Under consideration for publication in Knowledge and Information Systems

Patterns and Anomalies in k-Cores of Real-World Graphs with Applications

Kijung Shin[1], Tina Eliassi-Rad[2], and Christos Faloutsos[1]

[1]Computer Science Department, Carnegie Mellon University, Pittsburgh PA, USA;
[2]Network Science Institute, Northeastern University, Boston MA, USA

Abstract. How do the k-core structures of real-world graphs look like? What are the common patterns and the anomalies? How can we exploit them for applications? A k-core is the maximal subgraph in which all vertices have degree at least k. This concept has been applied to such diverse areas as hierarchical structure analysis, graph visualization, and graph clustering. Here, we explore pervasive patterns related to k-cores and emerging in graphs from diverse domains.

Our discoveries are: (1) Mirror Pattern: coreness (i.e., maximum k such that each vertex belongs to the k-core) is strongly correlated to degree. (2) Core-Triangle Pattern: degeneracy (i.e., maximum k such that the k-core exists) obeys a *3-to-1* power law with respect to the count of triangles. (3) Structured Core Pattern: degeneracy-cores are not cliques but have non-trivial structures such as core-periphery and communities.

Our algorithmic contributions show the usefulness of these patterns. (1) Core-A, which measures the deviation from Mirror Pattern, successfully spots anomalies in real-world graphs, (2) Core-D, a single-pass streaming algorithm based on Core-Triangle Pattern, accurately estimates degeneracy up to **12× faster** than its competitor. (3) Core-S, inspired by Structured Core Pattern, identifies influential spreaders up to **17× faster** than its competitors with comparable accuracy.

Keywords: Graph, k-core, degeneracy, influential node, anomaly detection, k-truss

1. Introduction

Given an undirected graph G, the k-core is the maximal subgraph of G in which every vertex is adjacent to at least k vertices (Batagelj and Zaversnik, 2003). As

Received 16 Mar 2017
Revised 19 May 2017
Accepted 06 Jun 2017

discussed in Section 8, this concept has been used extensively in diverse applications, including hierarchical structure analysis (Alvarez-Hamelin et al, 2008), graph visualization (Alvarez-Hamelin et al, 2006), protein function prediction (Wuchty and Almaas, 2005), and graph clustering (Giatsidis et al, 2014). An equally useful and closely related concept is the *degeneracy* of G, that is, the maximum k such that the k-core exists in G. For example, a clique of 5 vertices itself is a 4-core and thus has degeneracy 4; a ring of 10 vertices has degeneracy 2; a star of 100 vertices has degeneracy 1. The simplest algorithm to compute k-cores is the so-called *"shaving"* method: repeatedly deleting vertices with degree less than k until no such vertex is left.

Despite the huge interest in k-cores and their applications, it is not known whether k-cores or degeneracy follow any patterns in real graphs. Our motivating questions are: (1) what are common patterns regarding k-cores or degeneracy occurring across graphs in diverse domains? (2) are there anomalies deviating from these patterns? (3) how can these patterns and anomalies be used for algorithm design?

To answer these questions, we present three empirical patterns that govern k-cores or degeneracy across a wide variety of real-world graphs, including social networks, web graphs, internet topologies, and citation networks. We also show the practical uses of these patterns.

Our first Mirror Pattern states that the *coreness* of a vertex (i.e., the maximum k such that the vertex belongs to the k-core) is strongly correlated to its degree, as seen in Figure 1(a). We also observe that anomalies (e.g., the CEO in Figure 1(a) and accounts using a 'follower booster' in Twitter) tend to deviate from this pattern. This observation leads to Core-A, our anomaly detection method based on the degree of deviation from Mirror Pattern. We show that Core-A is complementary to recent densest-subgraph based anomaly detection methods (Shin et al, 2016a; Hooi et al, 2016a), and their combination has the best of the two approaches.

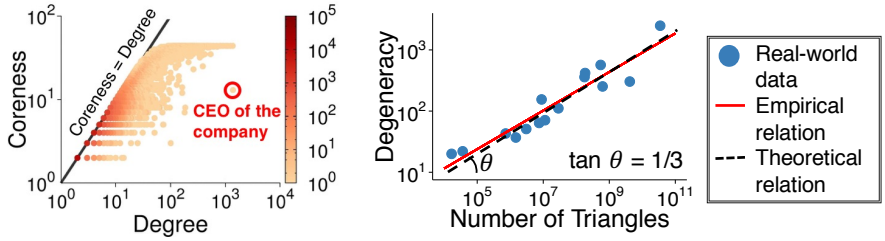
Our second discovery, Core-Triangle Pattern, states that, in real-world graphs, the degeneracy and the triangle-count obey a power-law with slope 1/3, as seen in Figure 1(b). This relation is theoretically analyzed in very realistic Kronecker graphs (Leskovec et al, 2005), and also utilized in Core-D, our single-pass streaming algorithm for estimating degeneracy. Core-D is up to *12*× *faster* than a recent multi-pass algorithm (Farach-Colton and Tsai, 2014), while providing comparable accuracy (see Figure 1(c)).

Our last discovery, Structured Core Pattern, states that degeneracy-cores in real-world graphs are not cliques but have non-trivial structures (core-periphery, communities, etc.), as seen in Figure 1(d). We also show that vertices central within degeneracy-cores are particularly good spreaders up to *2.6*× *more influential* than the average vertices in degeneracy-cores, which are already known as good spreaders (Kitsak et al, 2010). Those spreaders are spotted by Core-S, our influential spreader identification method, which is up to *17*× *faster* than its top competitors (Kitsak et al, 2010; Rossi et al, 2015; Macdonald et al, 2012) with similar accuracy.

In summary, the contributions of our work[1] are as follows:

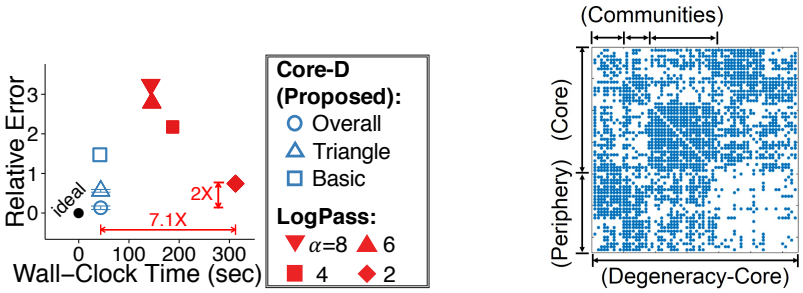
– **Patterns**: We discover three empirical patterns that hold across several real-world graphs from diverse domains.

[1] This paper is an extended version of (Shin et al, 2016b).



(a) P1: Mirror Pattern
A1: Anomaly Detection

(b) P2: Core-Triangle Pattern



(c) A2: Core-D Algorithm

(d) P3: Structured Core Pattern

Fig. 1. **Three patterns (P1-P3) discovered in real-world graphs, and their applications (A1-A3).** (a) **P1**: Coreness and degree are strongly correlated. **A1**: Anomalies deviate from this pattern. (b) **P2**: Degeneracy and the number of triangles in graphs obey a *3-to-1* power law, which is theoretically supported. (c) **A2**: Our Core-D algorithm (with Overall Model) estimates the degeneracy in a graph stream **7× faster** and **2× more accurately** than its competitor. (d) **P3**: As seen in the sparsity pattern of the adjacency matrix of a degeneracy-core, degeneracy-cores have structure, such as core-periphery and communities, which can be exploited for identifying influential spreaders (**A3**). See Appendix A for the interpretation of the sparsity pattern.

– **Anomalies**: We detect interesting anomalies (e.g., accounts using a 'follower-boosting' service in Twitter) from vertices deviating from the patterns.
– **Algorithms**: The patterns are practically used in our algorithms for detecting anomalies (Core-A), estimating degeneracy (Core-D), and identifying influential spreaders (Core-S). Our experiments show that our algorithms either complement or outperform state-of-the-art algorithms.

Reproducibility: Our open-sourced code and the data we used are available at `http://www.cs.cmu.edu/~kijungs/codes/kcore/`.

In Section 2, we give preliminaries on k-cores. In Section 3, we describe the datasets used in the paper. In Section 4, we present Mirror Pattern and its application to anomaly detection. In Section 5, we discuss Core-Triangle Pattern and Core-D, a streaming algorithm for estimating degeneracy. In Section 6, we describe Structured Core Pattern and its application to influential-spreader detection. In Section 7, we briefly discuss k-trusses, an extension of k-cores, in real-world graphs. After reviewing related work in Section 8, we draw conclusions in Section 9.

2. Preliminaries

In this section, we provide the definitions of k-cores and related concepts. We also discuss algorithms for computing k-cores and degeneracy.

2.1. Definitions and Notations

Let $G(V, E)$ be an undirected unweighted graph. We define $n = |V|$ and $m = |E|$. We denote the neighbors of a vertex $v \in V$ by $N(v) = \{u \in V | (u, v) \in E\}$ and its degree by $d(v) = |N(v)|$. Likewise, for a subgraph $G'(V', E')$ of G, we use $N_{G'}(v) = \{u \in V' | (u, v) \in E'\}$ and $d_{G'}(v) = |N_{G'}(v)|$.

The *k-core* or the *core of order k* (Batagelj and Zaversnik, 2003) is the maximal subgraph $G'(V', E')$ where $\forall v \in V'$, $d_{G'}(v) \geq k$. Notice that, for each k, there exists at most one k-core, and it is not necessarily a connected subgraph. Cores are nested, i.e., the k_1-core is a subgraph of the k_2-core if $k_1 \geq k_2$. The *coreness* or *core number* of a vertex v (Batagelj and Zaversnik, 2003), denoted by $c(v)$, is the order of the highest-order core that v belongs to. By definition, coreness is upper bounded by degree, i.e., $c(v) \leq d(v)$. The *degeneracy* of a graph G, defined as $k_{max} = \max_{v \in V} c(v)$, is the maximum coreness. The k_{max}-core is also called the *degeneracy-core*. We define the *density* of a subgraph as the ratio between the number of existing edges and the largest possible number of edges. If we let n_{max} and m_{max} be the numbers of the vertices and edges in the degeneracy-core, then the density of the degeneracy-core is $D_{max} = m_{max} / \binom{n_{max}}{2}$.

The k-truss is a concept closely related to the k-core. The *k-truss* or the *truss of order k* of G is the maximal subgraph of G where every edge is involved in at least $(k-2)$ triangles (i.e., cycles of length three) within the subgraph. As coreness is defined using k-cores, we define the *trussness* or *truss number* of each vertex $v \in V$ as the order of the highest-order truss that v belongs to, and we denote the trussness of v by $t(v)$.

Additionally, we denote the number of triangles in a graph G by $\#\Delta$. The eigenvalues of the adjacency matrix A of G are denoted by $(\lambda_1, ..., \lambda_n)$ where $\lambda_i \geq \lambda_j$ if $i < j$. Table 1 lists the symbols frequently used in the paper.

2.2. Algorithm for k-Cores and Degeneracy

The k-core remains if we remove vertices with degree less than k and edges incident to them recursively from G until no vertex has degree less than k. The $(k+1)$-core can be computed in the same way from the k-core since the $(k+1)$-core is a subgraph of the k-core. Likewise, by computing k-cores sequentially from $k = 1$ to $k = k_{max}$, we divide all vertices according to their coreness. This process, called *core decomposition*, runs in $O(n + m)$ (Batagelj and Zaversnik, 2003) if a graph fits in memory.

However, if a graph does not fit in memory, the computational cost grows. For example, in a graph stream, a recent method LogPass (Farach-Colton and Tsai, 2014) requires $O(\log_{\alpha/2}(n))$ passes of the entire graph for α-approximation of the degeneracy, for any real number α larger than 2. It requires $O(n)$ memory space, independent of α. In Section 5.3, however, we propose a single-pass algorithm for estimating degeneracy with similar memory requirements. Other algorithms for k-cores in large graphs are discussed in Section 8.

Table 1. Table of symbols frequently used in the paper.

Symbol	Definition
$G(V, E)$	undirected and unweighted graph
A	adjacency matrix of G
n	number of vertices in G
m	number of edges in G
k_{max}	degeneracy of G
n_{max}	number of vertices in the degeneracy-core
m_{max}	number of edges in the degeneracy-core
D_{max}	density of the degeneracy-core
d_{avg}	average degree of G
$c(v)$	coreness of vertex v
$t(v)$	trussness of vertex v
$d(v)$	degree of vertex v
r	Pearson correlation coefficient
ρ	Spearman's rank correlation coefficient
$dmp(v)$	vertex v's degree of deviation from Mirror Pattern
DSM	densest-subgraph based anomaly detection methods
$a\text{-}score(G')$	anomaly score of subgraph G'
$\#\Delta$	number of triangles in G
λ_i	i-th largest eigenvalue of A
$i(v)$	in-core centrality of vertex v
β	infection rate in the SIR Model

3. Datasets

In this section, we describe the datasets used in the following sections. Since the objective of this work is to find pervasive patterns emerging across graphs in diverse domains, our datasets include social networks, web graphs, internet topologies, and citation networks. The direction of edges is ignored in all the datasets because k-cores are defined only in undirected graphs. The datasets are summarized in Table 2 with the details below.

Social Networks. Hamster Dataset[2] is a friendship network between users of hamsterster.com, an online community for hamster owners. Catster Dataset[3] is a friendship network between users of catster.com, an online community for cat owners. YouTube Dataset (Mislove et al, 2007) is a friendship network between users of YouTube, a video-sharing web site. Flickr Dataset (Mislove et al, 2007) is a social network between users of Flickr, a photo sharing site. Orkut Dataset (Mislove et al, 2007) is a social network between users of Orkut, a social networking site. LiveJournal Dataset (Mislove et al, 2007) is a friendship network between users of Live Journal, an online blogging community. Friendster Dataset[4] is a friendship network between users of Friendster, a former social networking site. Twitter Dataset (Kwak et al, 2010) is a subscription network between users in Twitter, a microblogging service. Email Dataset (Klimt and Yang, 2004) is an email network between employees of Enron Corporation, an energy, commodities, and services company. This dataset also includes emails between the employees and people outside the company.

[2] http://konect.uni-koblenz.de/networks/petster-friendships-hamster
[3] http://konect.uni-koblenz.de/networks/petster-friendships-cat
[4] http://konect.uni-koblenz.de/networks/friendster

Table 2. Summary of the datasets used in the paper.

Category	Name	n	m	$\#\Delta$	k_{max}	n_{max}	D_{max}
Social Network	Hamster	1.86K	12.6K	16.8K	20	130	0.24
	Email	36.7K	184K	727K	43	275	0.26
	Catster	150K	5.45M	185M	419	1.28K	0.48
	YouTube	1.13M	2.99M	3.06M	51	845	0.10
	Flickr	1.72M	15.6M	548M	568	1.75K	0.49
	Orkut	3.07M	117M	628M	253	15.7K	0.03
	LiveJournal	4.00M	34.7M	178M	360	377	0.99
	Twitter	41.7M	1.20B	34.8B	2.49K	3.19K	0.90
	FriendSter	65.6M	1.81B	4.17B	304	24.5K	0.02
Web Graph	Stanford	282K	1.99M	11.3M	71	387	0.29
	NotreDame	326K	1.09M	8.91M	155	1.37K	0.12
Internet Topology	Caida	26.5K	53.4K	36.3K	22	64	0.53
	Skitter	1.70M	11.1M	28.8M	111	222	0.68
Citation Network	HepTh	27.8K	352K	1.48M	37	52	0.86
	Patent	3.77M	16.5M	7.52M	64	106	0.73

Web Graphs. NotreDame Dataset (Albert et al, 1999) and Stanford Dataset (Leskovec et al, 2009) are hyperlink networks between web pages from University of Notre Dame and Stanford University, respectively.

Internet Topologies. Caida Dataset[5] is an internet topology graph obtained from RouteViews Border Gateway Protocol routing tables. Skitter Dataset[6] is an internet topology graph obtained from traceroute data collected by Skitter, which is Caida's probing tool.

Citation Networks. Patent Dataset (Hall et al, 2001) is a citation network between patents registered with the United States Patent and Trademark Office. HepTh Dataset (Gehrke et al, 2003) is a citation network between papers submitted to arXiv High Energy Physics Theory Section.

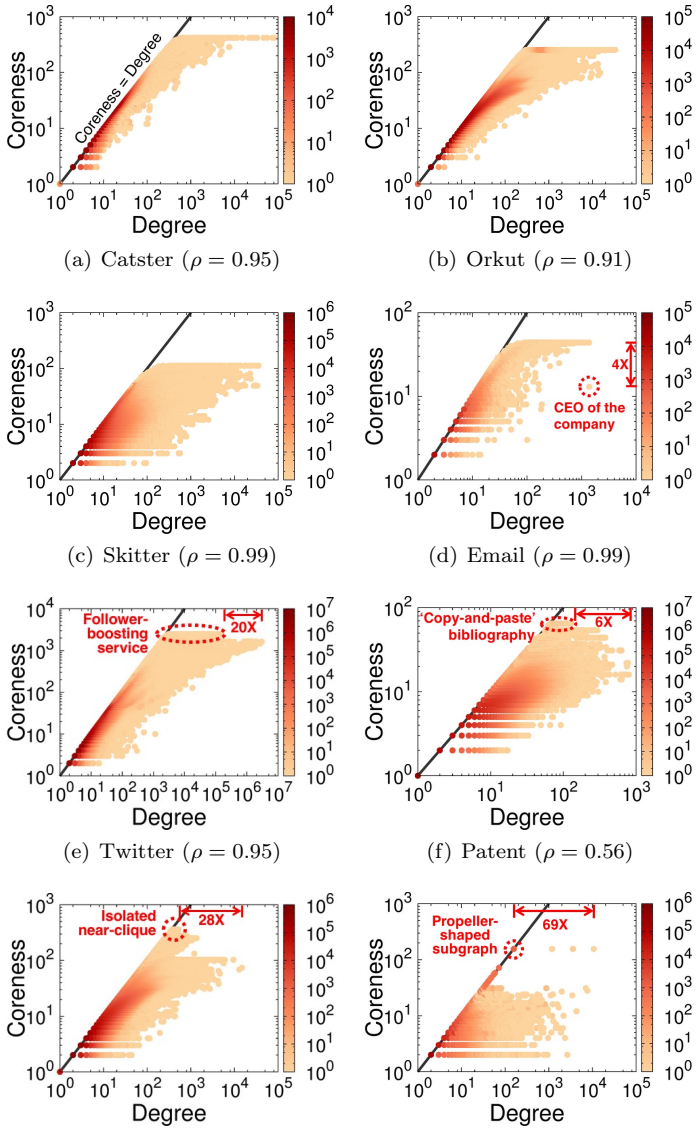
4. Pattern 1: "Mirror Pattern"

In this section, we discuss Mirror Pattern and its use for anomaly detection.

4.1. Observation: Pattern in Real-world Graphs

What are the key factors determining the coreness of the vertices in real-world graphs? We find out that a strong positive correlation exists between coreness and degree, which is an upper bound of coreness. Specifically, as seen in Figure 2, Spearman's rank correlation coefficient ρ [7] is significantly higher than 0 (no correlation) in all the considered graphs and close to 1 (perfect positive correlation) in many of them. This empirical pattern is described in Observation 4.1.

[5] http://www.caida.org/data/as-relationships
[6] http://www.caida.org/tools/measurements/skitter
[7] Spearman's rank correlation coefficient ρ (Spearman, 1904) is the standard (Pearson) correlation coefficient r of the ranks. Here, ρ is equivalent to r between the ranks of vertices in terms of degree and their ranks in terms of coreness. Using ρ is known to be robust to outlying values than simply using r. We ignored isolated vertices when computing ρ.



(a) Catster ($\rho = 0.95$) (b) Orkut ($\rho = 0.91$)

(c) Skitter ($\rho = 0.99$) (d) Email ($\rho = 0.99$)

(e) Twitter ($\rho = 0.95$) (f) Patent ($\rho = 0.56$)

(g) LiveJournal ($\rho = 0.93$) (h) NotreDame ($\rho = 0.99$)

Fig. 2. **Our Mirror Pattern is pervasive in real-world graphs; exceptions signal anomalies.** ρ ($\in [-1, 1]$) indicates Spearman's rank correlation coefficient; and colors are for heatmap of point density. Degree and coreness have strong positive correlation; exceptions (in red circles) are "strange": the vertex ranked first in terms of degree but relatively lower in terms of coreness corresponds to an email account of the company's CEO in (d); vertices ranked first in terms of coreness but relatively lower in terms of degree indicate accounts involved in a 'follower-boosting' service in (e), 'copy-and-paste' bibliography in (f), an isolated near-clique in (g), and a propeller-shaped subgraph in (h).

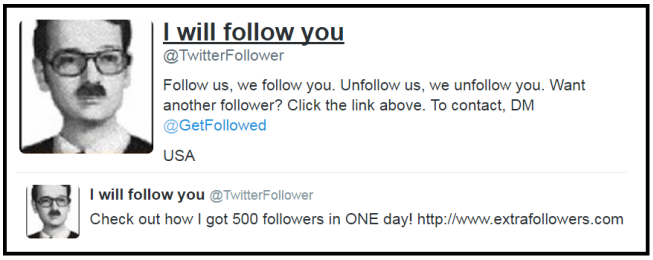


Fig. 3. **Vertices deviating from Mirror Pattern are involved in a 'Follower booster' in Twitter.** 78% of the vertices in the degeneracy-core were following the above Twitter account when the data were crawled. The account still exists without being suspended.

Observation 4.1. (Mirror Pattern) *In real-world graphs, coreness has a strong positive correlation with degree.*

4.2. Application: Anomaly Detection in Real-World Graphs

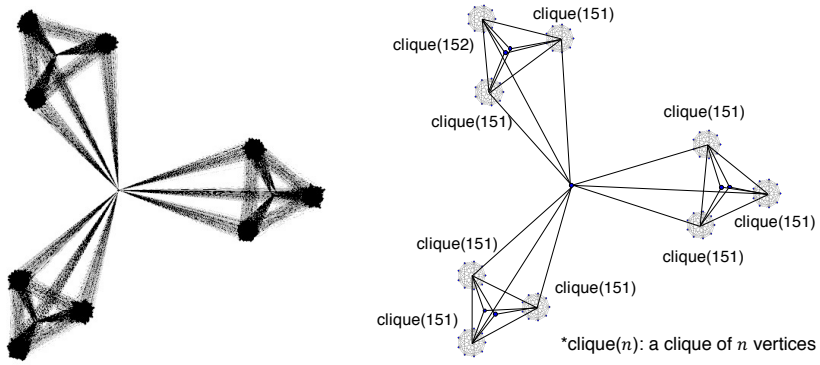
Mirror Pattern implies that vertices with high coreness have tendency to have high degree and vice versa. However, the degree-coreness plots in Figure 2 highlight some vertices deviating from the pattern, i.e., vertices ranked first in terms of degree but relatively lower in terms of coreness, and vice versa. In this section, we take a close look at these vertices and show that they indicate two different types of anomalies: *'loner-stars'* (i.e., vertices mostly connected to 'loners') or *'lockstep behavior'* (i.e., a group of similarly behaving vertices).

4.2.1. Second Email Account of the CEO (Loner-Star)

In the Email dataset, the vertex marked in Figure 2(d) has the highest degree 1,383 but relatively low coreness 12, deviating from Mirror Pattern. This vertex corresponds to the second email account of the former CEO of the company. This account was used only to receive emails, and not a single email was sent from this account. The former CEO used the other email account when sending emails. The 99.6% of the sources of the received emails are outside the company, while only 0.4% are inside. Since email accounts outside the company mostly have small coreness in the dataset (they are 'loners'), this anomalous email account has small coreness despite its high degree.

4.2.2. 'Follower-Boosting' Service in Twitter (Lockstep Behavior)

In Twitter, the vertices with the highest coreness, marked in Figure 2(e), have relatively low degrees, deviating from Mirror Pattern. We find out that at least 78% of the vertices with the highest coreness were directly involved in a 'Follower-Boosting' service (i.e., following '@TwitterFollower' in Figure 3) when the Twitter dataset was crawled. Since the accounts involved in the service are densely connected with each other ($D_{max} = 0.90$), to boost the number of followers, they have the highest coreness despite their relatively low degrees. Surprisingly, this misbehavior has been undetected by Twitter, and '@TwitterFollower' account has not been suspended or removed since the data was crawled in 2009.



(a) Degeneracy-core of the NotreDame web graph

(b) The structure of the degeneracy-core of the NotreDame web graph

Fig. 4. **Vertices deviating from Mirror Pattern form a propeller-shaped subgraph in the NotreDame dataset.** The degeneracy-core of NotreDame Dataset is composed by a set of cliques of the same size connected in a symmetric way. This subgraph is unlikely to occur naturally.

4.2.3. 'Copy-and-Paste' Bibliography (Lockstep Behavior)

As in Twitter, the vertices with the highest coreness in the Patent dataset have relatively low degrees, deviating from Mirror Pattern (see Figure 2(f)). We find out that 88% of these vertices are patents owned by the same pharmaceutical company, and bibliography in previous patents of the company has been reused repeatedly in a 'copy-and-paste' manner in later patents of the company. This results in a dense subgraph in the citation network, and the patents in the subgraph have the highest coreness despite their relatively low degrees.

4.2.4. Isolated Near-Clique in Live Journal (Lockstep Behavior)

Vertices with the highest coreness but relatively low degrees are also found in the LiveJournal dataset, as marked in Figure 2(g). Although we could not identify actual accounts corresponding to these 377 vertices, their abnormality was supported by the following facts: (1) the vertices form a near-clique with density 99.7%, unlikely to occur naturally, (2) the group formed by the vertices is isolated as judged from the fact that 88% of the neighbors of the vertices are also in the group, while only 12% are outside, and (3) the vertices have suspicious uniformity. Specifically, 127 vertices (one third of the considered vertices) have degrees between 387 and 391.

4.2.5. Propeller-Shaped Subgraph in Web (Lockstep Behavior)

As marked in Figure 2(h), in the NotreDame dataset, the vertices with the highest coreness have relatively low degree, deviating from Mirror Pattern. The structure of the degeneracy-core, the subgraph formed by these vertices, is shown in Figure 4. The degeneracy-core consists of 9 cliques of almost the same size (8 cliques of 151 vertices and a clique of 152 vertices). Moreover, the way these cliques are connected is surprisingly symmetric. That is, the cliques are

divided into 3 groups where the cliques in each group are connected to the same two vertices, and every clique is also connected to the center vertex. Although the actual web pages corresponding to the vertices composing this subgraph are unknown, this subgraph is unlikely to occur naturally. This subgraph seems to be artificially constructed for special purposes.

4.3. Core-A: Algorithm for Anomaly Detection

Inspired by the observations in the previous section, we propose CORE-A, an anomaly detection method based on the deviation from MIRROR PATTERN. We show that CORE-A is complementary to densest-subgraph based anomaly detection, and their combination has the best of the two methods.

4.3.1. Algorithm

In the previous section, we show that vertices deviating from MIRROR PATTERN are worth noticing, as they indicate the two types of anomalies: 'loner-stars' (e.g. the CEO in Figure 2(d)) and 'lockstep behavior' (e.g., an isolated near-clique in Figure 2(g)). What scoring function gives a high score, to both types of anomalies? *Deviation from* MIRROR PATTERN (*dmp*) in Definition 4.1 gives an answer. CORE-A, our proposed anomaly detection method, ranks vertices in decreasing order of *dmp*. The main idea behind our proposed *dmp* measure, is to use the *rank* of each vertex, and since we expect power-laws, the log of the rank. Specifically, we use $rank_d(v)$, the fractional rank[8] of vertex v in decreasing degree order, and similarly, $rank_c(v)$, in decreasing coreness order (in case of the same coreness, in decreasing degree order).

Definition 4.1 (Deviation from MIRROR PATTERN)**.** *A vertex v's degree of deviation from* MIRROR PATTERN *in graph G is defined as*

$$dmp(v) \equiv |\log(rank_d(v)) - \log(rank_c(v))|.$$

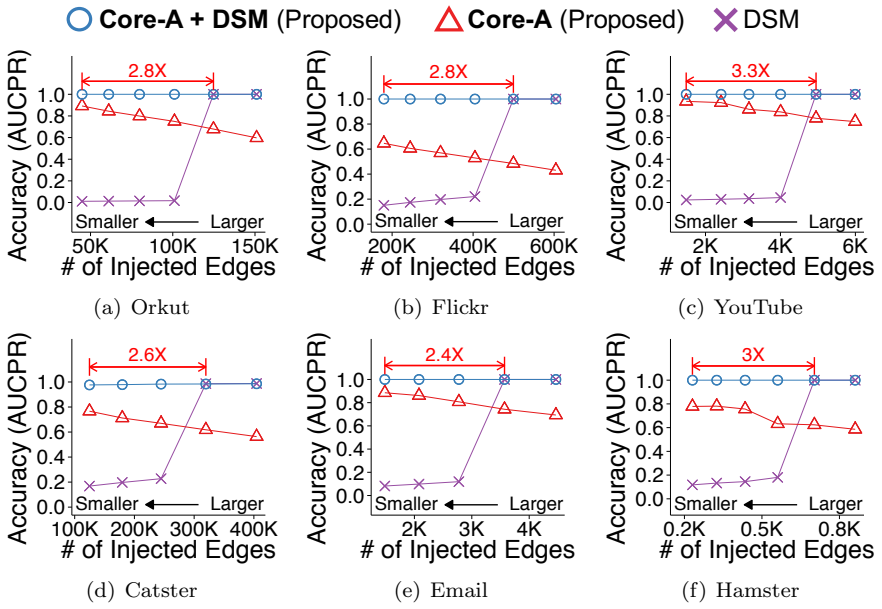
CORE-A has time complexity $O(n+m)$ since the *dmp* scores of all vertices can be computed in $O(n)$ using 'counting sort' once we compute core decomposition in $O(n+m)$ (Batagelj and Zaversnik, 2003).

4.3.2. Complementarity of CORE-A

Anomaly detection in graphs (especially in social networks) has been extensively researched (see Section 8), and many of them detect dense subgraphs since anomalies tend to form dense subgraphs, as we also show in Section 4.2. Especially, the latest methods (Shin et al, 2016a; Hooi et al, 2016a) are based on densest subgraphs (i.e., subgraphs with maximum average degree). We show that CORE-A and these densest-subgraph based methods (DSM) are complementary as they are good at detecting different-size dense subgraphs.

To demonstrate that CORE-A and DSM (specifically M-ZOOM (Shin et al, 2016a), which includes FRAUDAR (Hooi et al, 2016a) as a special case) are complementary, we compare their performances when different-size subgraphs

[8] The fractional rank of an item is one plus the number of items greater than it plus half the number of items equal to it.



(a) Orkut (b) Flickr (c) YouTube

(d) Catster (e) Email (f) Hamster

Fig. 5. **Core-A is complementary to DSM; their combination has the best of the two.** In social networks, our Core-A method accurately detects small injected subgraphs that cannot be detected accurately by DSM. The combination of Core-A and DSM successfully detects both small and large injected subgraphs. The combination detects up to **3.3× smaller** injected subgraphs than DSM with near-perfect accuracy.

are injected into social networks. We randomly choose k vertices and inject $\binom{k}{2}$ edges between them into each network. Then, we compare how precisely and exhaustively each method detects the k chosen vertices using Area Under the Precision-Recall Curve (AUCPR) (Davis and Goadrich, 2006).

As seen in Figure 5, DSM cannot detect small dense subgraphs accurately, while it detects large ones with near-perfect accuracy. In contrast, Core-A is more accurate for smaller subgraphs that cannot be detected by DSM. This is explained by the fact that the k chosen vertices have degree and coreness at least $k-1$. If $k \approx c_{max}$ but $k \ll d_{max}$, the vertices tend to have high *dmp* scores since they have small $rank_c$ but are likely to have large $rank_d$. However, if $k \approx d_{max}$, the vertices have low *dmp* scores since they have small $rank_d$ as well as small $rank_c$.

4.3.3. Combination with DSM

We can have the best of Core-A and DSM by combining them. Specifically, we propose to define the *anomaly score* (*a-score*) of a subgraph $G'(V', E')$ in a graph G based on *dmp* scores in G as well as the average degree as follows:

$$a\text{-}score(G') = \frac{|E'|}{|V'|} + w \sum_{v \in V'} \frac{dmp(v)}{|V'|} \tag{1}$$

where $w > 0$ is a parameter for balancing the two factors: $\frac{|E'|}{|V'|}$ and $\sum_{v \in V'} \frac{dmp(v)}{|V'|}$. We set w to the ratio of the maximum values of the factors in the given graph $G(V, E)$. The maximum value of $\frac{|E'|}{|V'|}$ is close (within a factor of 2) to $\frac{|E^*|}{|V^*|}$, where $G^*(V^*, E^*)$ is the densest subgraph detected by DSM; and the maximum value of $\sum_{v \in V'} \frac{dmp(v)}{|V'|}$ is $\max_{v \in V} dmp(v)$. We set w to their ratio, i.e.,

$$w = \frac{|E^*|}{|V^*|} \times \frac{1}{\max_{v \in V} dmp(v)}. \tag{2}$$

Once we set w, we use (Shin et al, 2016a) to identify the subgraph maximizing *a-score* (Eq (1)). We classify the vertices in the subgraph into anomalies. This entire process takes $O(m \log n)$, as DSM does (Shin et al, 2016a; Hooi et al, 2016a).

Figure 5 illustrates the success of our proposal to combine the scores (Eq. (1)): our combination successfully detects both small and large dense subgraphs injected into social networks, outperforming both its component methods (i.e., CORE-A and DSM), Especially, the combination detects up to **3.3× smaller** dense subgraphs than DSM, with near-perfect accuracy.

In Section 7, we extend CORE-A to TRUSS-A for better accuracy at the expense of more computational cost.

5. Pattern 2: "Core-Triangle Pattern"

In this section, we present CORE-TRIANGLE PATTERN (C-T PATTERN) in real-world graphs and provide mathematical analysis of the pattern. Then, we propose an one-pass streaming algorithm for estimating degeneracy, based on the pattern.

5.1. Observation: Pattern in Real-world Graphs

What are the major factors determining degeneracy, the maximum coreness, in real-world graphs? We investigate the relation between degeneracy and various graph measures in real-world graphs. As seen in Figure 6, the number of triangles has a particularly strong correlation ($r = 0.94$) with degeneracy in log scale, compared to the vertex-count ($r = 0.75$) and the edge-count ($r = 0.83$), which have a weaker correlation with degeneracy. Moreover, the slope is 0.32, which is very close to 1/3. This leads to Observation 5.1.

Observation 5.1. (CORE-TRIANGLE PATTERN or C-T PATTERN) *In real-world graphs, the triangle count and the degeneracy obey a* 3*-to-*1 *power law. That is,*

$$k_{max} \propto (\#\Delta)^{\frac{1}{3}}.$$

5.2. Analysis in Kronecker and ER Models

Why do real-world graphs obey C-T PATTERN? Here we show that C-T PATTERN holds in the so-called 'Kronecker Model' (Definition 5.1), which is considered as a very realistic graph model obeying common patterns in real-world graphs (Leskovec et al, 2005).

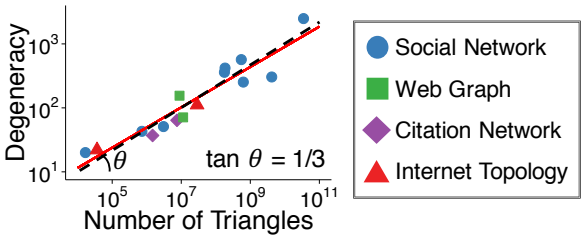


Fig. 6. **Core-Triangle Pattern: triangle count and degeneracy obey a 3-to-1 power law.** Each point corresponds to a graph dataset in Table 2. The count of triangles has a strong correlation ($r = 0.94$) with degeneracy in log scale. Moreover, the slope is very close to the theoretical slope 1/3 (dashed line).

Definition 5.1 (Kronecker Graph (Leskovec et al, 2005))**.** *Let G_q be the q-th power Kronecker graph of a seed graph G_1. If we denote the adjacency matrix of G_q by A_q, then A_q is defined as:*

$$A_q = A_{q-1} \otimes A_1 = \underbrace{A_1 \otimes A_1 \otimes ... \otimes A_1}_{q \text{ times}},$$

where $\otimes$ denotes Kronecker Product.

C-T Pattern in the model is defined formally in Definition 5.2, where we ignore constant factors for ease of analysis.

Definition 5.2. (C-T Pattern in Kronecker Model). *A Kronecker model with seed graph G_1 follows* C-T Pattern *if Eq.* (3) *holds in $\{G_q\}_{q \geq 1}$, graphs generated by the model.*

$$k_{max} = \Theta(\#\Delta^{\frac{1}{3}}) \text{ or equivalently } \#\Delta = \Theta(k_{max}^3). \tag{3}$$

To prove C-T Pattern in Kronecker Model, we use Lemmas 5.1 and 5.2, which give upper and lower bounds of degeneracy.

Lemma 5.1 (Lower Bound of Degeneracy (Erdös, 1963))**.** *The half of the average degree lower bounds the degeneracy. That is, if we let d_{avg} be the average degree, then $k_{max} \geq d_{avg}/2$.*

Lemma 5.2 (Upper Bound of Degeneracy)**.** *The largest eigenvalue upper bounds the degeneracy. That is, if we let λ_1 be the largest eigenvalue of the adjacency matrix, then $k_{max} \leq \lambda_1$.*

Proof. Let H be the degeneracy-core (i.e., k_{max}-core) of G and $d_{min}(H)$ be its minimum degree. By the definition of the k-core and the degeneracy, $d_{min}(H) = k_{max}(G)$. Since the largest eigenvalue is lower bounded by minimum degree (Brouwer and Haemers, 2001), $k_{max}(G) = d_{min}(H) \leq \lambda_1(H)$. The largest eigenvalue of a graph is also lower bounded by that of its induced subgraph (Brouwer and Haemers, 2001). Since the degeneracy-core is an induced subgraph due to its maximality, $k_{max}(G) \leq \lambda_1(H) \leq \lambda_1(G) = \lambda_1$. ■

Lemma 5.3 states that the graph measures used for upper and lower bounding degeneracy in Lemmas 5.1 and 5.2 increase exponentially with q, the power of Kronecker products, in Kronecker Model.

Lemma 5.3 (Graph Measures Increasing Exponentially in Kronecker Graphs)**.** *The average degree, the degeneracy, and the largest eigenvalue increase exponentially with q in $\{G_q\}_{q\geq 1}$, graphs generated by Kronecker Model. Specifically,*

$$d_{avg}(G_q) = (d_{avg}(G_1))^q,\ \forall q \geq 1. \tag{4}$$
$$k_{max}(G_q) \geq (k_{max}(G_1))^q,\ \forall q \geq 1. \tag{5}$$
$$\lambda_1(G_q) = (\lambda_1(G_1))^q,\ \forall q \geq 1. \tag{6}$$

Proof. Let $n(G)$ be the number of vertices and $nz(G)$ be the number of non-zero entries in the adjacency matrix.

We first show Eq. (4). From $d_{avg}(G) = nz(G)/n(G)$, $n(G_q) = (n(G_1))^q$, and $nz(G_q) = (nz(G_1))^q$, we have

$$d_{avg}(G_q) = \frac{nz(G_q)}{n(G_q)} = \frac{(nz(G_1))^q}{(n(G_1))^q} = \left(\frac{nz(G_1)}{n(G_1)}\right)^q = (d_{avg}(G_1))^q, \forall q \geq 1.$$

We prove Eq. (5) by induction. For seed graph G_1, $k_{max}(G_1) \geq (k_{max}(G_1))^1$. Assume $k_{max}(G_i) \geq (k_{max}(G_1))^i$. Each vertex in G_{i+1} can be represented as an ordered pair (v_i, v_1) where v_i is a vertex of G_i and v_1 is a vertex of G_1. Two vertices, (v_i, v_1) and (v_i', v_1'), in G_{i+1} are adjacent if and only if v_i and v_i' are adjacent in G_i and v_1 and v_1' are adjacent in G_1 (Leskovec et al, 2005). Let $G_i'(V_i', E_i')$ be the degeneracy-core of $G_i(V_i, E_i)$ where $V_i' = \{v_i \in V_i | c(v_i) = k_{max}(G_i)\}$. Then, each vertex (v_i, v_1) in $S = \{(v_i, v_1) \in V_{i+1} | v_i \in V_i', v_1 \in V_1'\}$ are adjacent to $d_{G_i'}(v_i) \times d_{G_1'}(v_1) (\geq k_{max}(G_i) \times k_{max}(G_1))$ vertices in S. Therefore, $k_{max}(G_{i+1}) \geq k_{max}(G_i) \times k_{max}(G_1) \geq k_{max}(G_1)^{(i+1)}$. By induction, $k_{max}(G_q) \geq (k_{max}(G_1))^q$, $\forall q \geq 1$.

Finally, to show Eq. (6), let $\lambda(G) = (\lambda_1, ..., \lambda_n)$ be the eigenvalues of the adjacency matrix of G, and $\lambda_1(G)$ be the largest eigenvalue. Then, $\lambda(G_q) = sort(\lambda(G_{q-1}) \otimes \lambda(G_1))$ (Van Loan, 2000). As $\lambda_1(G_q) = \lambda_1(G_{q-1}) \times \lambda_1(G_1)$, $\lambda_1(G_q) = (\lambda_1(G_1))^q$, $\forall q \geq 1$ holds. ■

Lemmas 5.4 and 5.5 state how rapidly degeneracy and triangle count increase in Kronecker Model. Both of them increase exponentially with q, the power of Kronecker products, and the base numbers depend on seed graphs. For Lemma 5.5, we have to deal with self-loops in Kronecker graphs which happen naturally. We add one to the degree for each self-loop and define *a triangle in Kronecker graphs* as an unordered vertex triplet, which can contain multiple instances of the same vertex, where every instance is connected to the others either by self-loops or other edges. For example, (v_1, v_1, v_2) is a triangle in Kronecker graphs if v_1 has a self-loop and v_1 and v_2 are adjacent. Note that Lemma 5.5 and Theorem 5.1 hold equally, with the original definitions of degree and a triangle, in Kronecker graphs without self-loops.

Lemma 5.4. (Degeneracy in Kronecker Model). *Degeneracy in $\{G_q\}_{q\geq 1}$ increases exponentially with q. Let d_{avg} be the average degree and λ_1 be the largest eigenvalue of the adjacency matrix. Then,*

$$k_{max}(G_q) = \Omega(max\{(d_{avg}(G_1))^q, (k_{max}(G_1))^q\}). \tag{7}$$
$$k_{max}(G_q) = O((\lambda_1(G_1))^q). \tag{8}$$

Proof. Lemma 5.4 is immediate from Lemmas 5.1, 5.2, and 5.3. ■

Lemma 5.5. (Triangles in Kronecker Model). *The number of triangles in $\{G_q\}_{q\geq 1}$*

increases exponentially with q. That is, if we let $\lambda(G_1) = (\lambda_1, ..., \lambda_n)$ be the eigenvalues of the adjacency matrix of the seed graph G_1, then

$$\#\Delta(G_q) = \Theta\left(\left(\sum_{i=1}^{n} \lambda_i^3\right)^q\right). \tag{9}$$

Proof. Let $\lambda(G_i) = (\lambda_1(G_i), ..., \lambda_{n^i}(G_i))$ be the eigenvalues of the adjacency matrix of G_i. The number of walks of length 3 in G_i that begin and end on the same vertex is $\sum_{j=1}^{n^i}(\lambda_j(G_i))^3$ (Tsourakakis, 2008) and linearly related to the number of triangles, i.e., $\#\Delta(G_i) = \Theta(\sum_{j=1}^{n^i}(\lambda_j(G_i))^3)$. For seed graph G_1, $\sum_{j=1}^{n}(\lambda_j(G_1))^3 = (\sum_{j=1}^{n}(\lambda_j(G_1))^3)^1$. Assume $\sum_{j=1}^{n^i}(\lambda_j(G_i))^3 = (\sum_{j=1}^{n}(\lambda_j(G_1))^3)^i$. As $\lambda(G_{i+1}) = sort(\lambda(G_i) \otimes \lambda(G_1))$ (Van Loan, 2000),

$$\begin{aligned}\sum_{j=1}^{n^{(i+1)}} (\lambda_j(G_{i+1}))^3 &= \sum_{r=1}^{n^i}\sum_{s=1}^{n}(\lambda_r(G_i))^3(\lambda_s(G_1))^3 \\ &= \left(\sum_{r=1}^{n^i}(\lambda_r(G_i))^3\right)\left(\sum_{s=1}^{n}(\lambda_s(G_1))^3\right) = \left(\sum_{s=1}^{n}(\lambda_s(G_1))^3\right)^{(i+1)}.\end{aligned}$$

By induction, $\sum_{j=1}^{n^q}(\lambda_j(G_q))^3 = (\sum_{j=1}^{n}(\lambda_j(G_1))^3)^q$, $\forall q \geq 1$. Hence, $\#\Delta(G_q) = \Theta(\sum_{j=1}^{n^q}(\lambda_j(G_q))^3) = \Theta((\sum_{j=1}^{n}(\lambda_j(G_1))^3)^q)$, $\forall q \geq 1$. ■

Based on the speed of increase of degeneracy and triangle count given in Lemmas 5.4 and 5.5, Theorem 5.1 states a sufficient and a necessary condition for C-T Pattern to hold in Kronecker Model. Note that $\sum_{i=1}^{n} \lambda_i^3 = \lambda_1^3$ in Eq. (10) and $\sum_{i=1}^{n} \lambda_i^3 \leq \lambda_1^3$ in Eq. (11) can hold since the eigenvalues can be negative.

Theorem 5.1. (C-T Pattern in Kronecker Model). *In a Kronecker model with a seed graph G,*

(1) *A sufficient condition for* C-T Pattern *to hold is, in the seed graph G,*

$$\max(d_{avg}^3, k_{max}^3) = \sum_{i=1}^{n} \lambda_i^3 = \lambda_1^3. \tag{10}$$

(2) *A seed graph satisfying the sufficient condition exists.*

(3) *A necessary condition for* C-T Pattern *to hold is, in the seed graph G,*

$$\max(d_{avg}^3, k_{max}^3) \leq \sum_{i=1}^{n} \lambda_i^3 \leq \lambda_1^3. \tag{11}$$

Proof. Assume that the sufficient condition holds, and $c = \max(d_{avg}^3, k_{max}^3) = \sum_{i=1}^{n} \lambda_i^3 = \lambda_1^3$. Then, $(k_{max}(G_q))^3 = \Theta(c^q)$ by Lemma 5.4, and $\#\Delta(G_q) = \Theta(c^q)$ by Lemma 5.5. Therefore, $\#\Delta(G_q) = \Theta((k_{max}(G_q))^3)$, and C-T Pattern holds. The Mediator seed graph in Table 3 satisfies this sufficient condition.

Assume that the necessary condition is not met. By Lemmas 5.4 and 5.5, $(k_{max}(G_q))^3$ increases faster than $\#\Delta(G_q)$ if $\sum_{i=1}^{n} \lambda_i^3 < \max(d_{avg}^3, k_{max}^3)$. Instead, $\#\Delta(G_q)$ increases faster than $(k_{max}(G_q))^3$ if $\lambda_1^3 < \sum_{i=1}^{n} \lambda_i^3$. Hence, $\#\Delta(G_q) \neq \Theta((k_{max}(G_q))^3)$, and C-T Pattern does not hold. ■

Table 3. Sample seed graphs for Kronecker Model. All graphs satisfy the necessary condition for C-T Pattern, and Mediator satisfies also the sufficient condition. When computing k_{max} and d_{avg}, we add one to the degree for each self-loop if self-loops exist.

	Core-Periphery	**Mediator**	**Triangle**	**Star**
Shape				
k^3_{max}	1	8	8	1
d^3_{avg}	3.38	8	18.96	5.36
$\sum_{i=1}^{n} \lambda_i^3$	4	8	20	10
λ_1^3	4.24	8	20.39	12.21

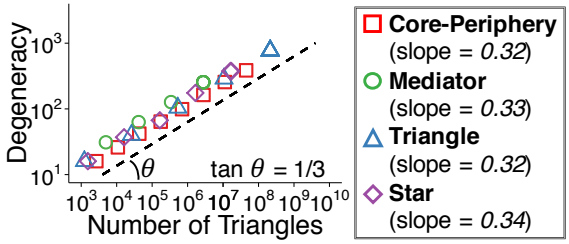


Fig. 7. **Core-Triangle Pattern in Kronecker Model.** Points represent graphs generated by Kronecker Model with different seed graphs. The slopes between the triangle count and the degeneracy are close to 1/3 (dashed line) in log scale regardless of seed graphs.

Many realistic seed graphs satisfy the necessary condition for C-T Pattern, as listed in Table 3. Especially, Mediator satisfies also the sufficient condition. Even seed graphs that do not satisfy the sufficient condition empirically follow C-T Pattern, as seen in Figure 7. The slope of the regression line between the number of triangles and degeneracy is close to 1/3 in log scale with all the seed graphs considered.

In addition to Kronecker Model, C-T Pattern is proved also in Erdős-Rényi (ER) Model, another mathematically tractable graph generation model where each of possible $\binom{n}{2}$ edges occurs independently with probability p, as formalized in Theorem 5.2. Figure 8 shows C-T Pattern in ER random graphs generated with different p values. The slopes of the regression lines are close to 1/3 in log scale, regardless of p values.

Theorem 5.2. (C-T Pattern in ER Model). *Graphs generated by ER Model with probability p follow* C-T Pattern *in terms of expected values if $p = \Omega(\log n/n)$. That is,*

$$\mathbb{E}[\#\Delta] = \Theta(\mathbb{E}[k_{max}]^3).$$

Proof. From $p = \Omega(\log n/n)$, there exists $c > 0$ such that $p \geq c \log n/n$. Let

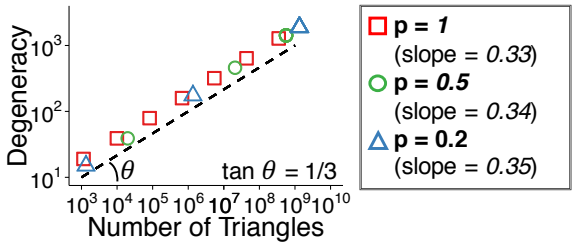


Fig. 8. **Core-Triangle Pattern in ER Model.** Points represent graphs generated by ER Model with different p values. The slopes between the triangle count and the degeneracy are close to 1/3 (dashed line) in log scale regardless of p.

$\epsilon = \max(2, 12/c)$ (> 1). Then,

$$\begin{aligned}
&\mathbb{P}[\exists v \in V \text{ s.t. } d(v) > (1+\epsilon)(n-1)p] \\
&\quad \le n\mathbb{P}[d(v) > (1+\epsilon)(n-1)p] && \text{(Boole's inequality)} \\
&\quad \le n\exp\{-(n-1)p\epsilon/3\} && \text{(Chernoff bound)} \\
&\quad \le n\exp\{-c\log(n)(n-1)\epsilon/3n\} && (p \ge c\log(n)/n) \\
&\quad \le n\exp\{-4\log(n)(n-1)/n\} && (\epsilon \ge 12/c) \\
&\quad \le n\exp\{-2\log n\} = n^{-1}.
\end{aligned}$$

Let $q = \mathbb{P}[\exists v \in V \text{ s.t. } d(v) > (1+\epsilon)(n-1)p]$. Then,

$$\begin{aligned}
\mathbb{E}[k_{max}] &\le \mathbb{E}[d_{max}] \le (1-q)(1+\epsilon)(n-1)p + q(n-1) \\
&\le (1+\epsilon)(n-1)p + (n-1)/n = O(np)
\end{aligned}$$

Hence, $\mathbb{E}[k_{max}] = O(np)$. As $\mathbb{E}[k_{max}] \ge \mathbb{E}[d_{avg}/2] = \Omega(np)$ by Lemma 5.1, $\mathbb{E}[k_{max}] = \Theta(np)$ holds.

On the other hand, the expected number of triangles is the sum of probabilities that each three vertices form a traingle:

$$\mathbb{E}[\#\Delta] = \frac{n(n-1)(n-2)}{6}p^3.$$

Therefore, $\mathbb{E}[\#\Delta] = \Theta(n^3p^3) = \Theta(\mathbb{E}[k_{max}]^3)$ holds. ■

5.3. Core-D: Streaming Algorithm for Degeneracy

Based on C-T Pattern, we propose Core-D, a single-pass streaming algorithm for estimating degeneracy. We empirically show that Core-D gives a better trade-off between speed and accuracy than a state-of-the-art method.

5.3.1. Algorithm

Computing degeneracy in a graph stream not fitting in memory remains as a challenge. As explained in Section 2.2, a recent approximate method, namely LogPass, needs $O(\log_{\alpha/2}(n))$ passes of a graph stream for α-approximation of its degeneracy, for any real number α greater than 2, with $O(n)$ memory

Table 4. Models of CORE-D. *: p value ≤ 0.05, ****: p value ≤ 0.0001. OVERALL MODEL fits the data best (i.e., has the highest adjusted R^2), and **only the log of triangle-count is statistically significant** with p value < 0.001.

Model	Variable	Coefficient Estimate	Coefficient Std.Err.	p-value
Basic ($R^2_{adj} = 0.72$)	1	-0.03	0.43	0.94
	$\log(n)$	-0.35	0.28	0.24
	$\log(m)$	0.62	0.24	0.02 *
Triangle ($R^2_{adj} = 0.89$)	1	-0.20	0.23	0.40
	$\log(\#\Delta)$	0.32	0.03	1.3e-07 ****
Overall ($R^2_{adj} = 0.95$)	1	0.03	0.20	0.88
	$\log(n)$	0.18	0.15	0.26
	$\log(m)$	-0.50	0.20	0.03 *
	$\log(\#\Delta)$	0.59	0.09	3.3e-05 ****

requirements (regardless of α). However, multiple passes of graph streams are time-consuming and not even possible in many real-world settings.

In contrast, the number of triangles can be estimated accurately even in a single pass (Tsourakakis et al, 2009; Lim and Kang, 2015; De Stefani et al, 2016). Simply sampling each edge with probability p in a graph stream and estimating the number of triangles in the whole graph from that in the sampled graph (Tsourakakis et al, 2009) also can be thought as a single-pass streaming algorithm if the sampled graph fits in memory and needs not be streamed again. This sampling method, which our CORE-D method uses, estimates triangle-count accurately even with less than n sampled edges.

CORE-TRIANGLE PATTERN (Observation 5.1), a high correlation between degeneracy and the number of triangles, enables using the accurately estimated triangle-count for estimating degeneracy. Specifically, we consider the following models, whose coefficients are denoted by w, relating the number of triangles and degeneracy:

– **Basic Model** (Baseline): $\log(\hat{k}_{max}) = w_{0,0} + w_{0,1}\log(n) + w_{0,2}\log(m)$
– **Triangle Model**: $\log(\hat{k}_{max}) = w_{1,0} + w_{1,1}\log(\#\Delta)$
– **Overall Model**: $\log(\hat{k}_{max}) = w_{2,0} + w_{2,1}\log(n) + w_{2,2}\log(m) + w_{2,3}\log(\#\Delta)$

Table 4 summarizes the estimates of the coefficients obtained by linear regression on the real-world graphs listed in Table 2. The OVERALL MODEL has the highest adjusted R-squared (0.95) among all possible linear models, and the log triangle-count is statistically significant with p-value < 0.001, proving the effectiveness of using triangle-count for estimating degeneracy.

Given a new graph stream, we estimate the vertex-count, the edge-count, and the triangle-count in the graph in a single pass. Then, by plugging these statistics into one of the models, whose coefficients are given as input parameters, we obtain an estimate of degeneracy. Algorithm 1 describes the details of CORE-D with TRIANGLE MODEL. For estimating the triangle-count, CORE-D requires $O(mp)$ memory space on average to store sampled edges. The memory requirement becomes $O(n)$ if we set sampling probability p to $O(n/m)$.

We also need n and m for BASIC MODEL and OVERALL MODEL. We obtain m by simply counting edges in the graph stream. In many real-world settings, n is available or is easily computed from the difference between maximum and

Require: Graph stream: G, Sampling probability: p, Coefficients in triangle model: $(w_{1,0}, w_{1,1})$
Ensure: Estimated degeneracy: $\hat{k}_{max}$

1: $G_{Sample} = \emptyset$
2: **for each** edge e in G **do**
3: add e to G_{Sample} with probability p
4: **end for**
5: $\#\Delta_{Sample} \leftarrow$ InMemoryTriangleCounting(G_{Sample}) (Schank, 2007)
6: $\hat{\#\Delta} \leftarrow \#\Delta_{Sample} * (1/p)^3$
7: $\hat{k}_{max} \leftarrow exp(w_{1,0} + w_{1,1} \log(\hat{\#\Delta}))$
8: **return** $\hat{k}_{max}$

Algorithm 1: Core-D with Triangle Model

minimum vertex ids. Otherwise, we obtain n by counting distinct vertex ids with $O(n)$ space. Even when n and m are needed, Core-D still requires only one pass because both edge sampling (lines 2-4 of Algorithm 1) and computing n and m can be conducted at the same time within one pass.

5.3.2. Experiments

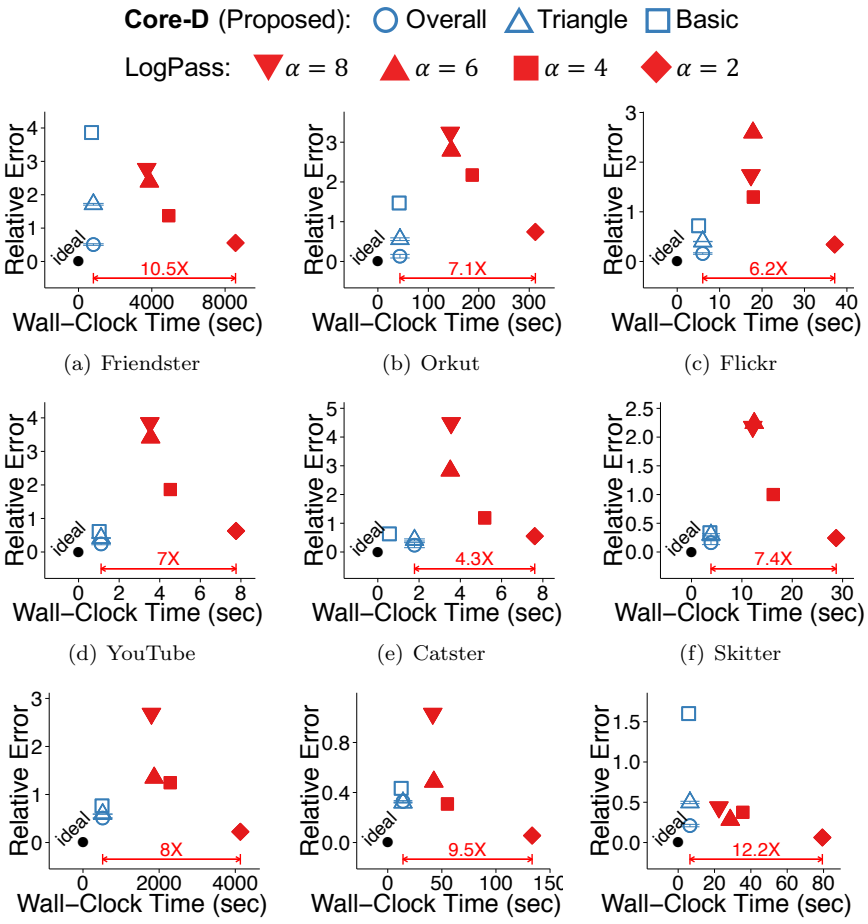
We compare the speed, accuracy, and memory efficiency of Core-D and LogPass. We used a desktop with a 3.6GHz cpu and 16GB memory space, and graphs (see Table 2) were streamed from disk whose speed is 192MB/sec for sequential read. We assumed that n is known or is computed easily from vertex ids in all methods. We set sampling probability p to $n/(5m)$. With this value of p, Core-D estimated degeneracy reliably and accurately, while using similar amount of memory space to LogPass. The effect of p on the accuracy of Core-D is explored in Appendix B. For the coefficients of the models (e.g., $w_{1,0}$ and $w_{1,1}$ in Algorithm 1), we used the values estimated from the real-world graphs listed in Table 2. Specifically, we used $\log(n)$, $\log(m)$ and $\log(\#\Delta)$ in all the datasets except the one being tested as training data, and learned the coefficients using linear regression. A graph being tested was excluded from training data for fairly evaluating accuracy in a new (unseen) graph. To measure the accuracy of the considered algorithms, we used *relative error* defined as:

$$relative\_error(k_{max}, \hat{k}_{max}) \equiv |k_{max} - \hat{k}_{max}|/k_{max}.$$

For randomized algorithms, we reported the average over ten runs.

As seen in Figure 9, Core-D provided a significantly better trade-off between accuracy and speed than LogPass. Specifically, Core-D (with Overall Model) was up to **12× faster** than LogPass ($\alpha = 2$) with similar accuracy. Noteworthy, Core-D with Overall Model was more accurate than LogPass in all the datasets except the ones whose degeneracies are known to be affected by anomalies (see Section 4.2). Among the models of Core-D, Overall Model consistently yielded the best performance in all the datasets. Basic Model, solely based on the numbers of vertices and edges, showed the lowest accuracy especially in the Friendster dataset and the Patent dataset. This supports the effectiveness of using the number of triangles for estimating degeneracy.

We experimentally compare the memory requirements of Core-D and Log-



(a) Friendster (b) Orkut (c) Flickr

(d) YouTube (e) Catster (f) Skitter

(g) Twitter* (h) LiveJournal* (i) Patent*

* Graphs whose degeneracies are known to be affected by anomalies (see Section 4.2)

Fig. 9. **Core-D achieves both speed and accuracy.** Points in each plot represent the performances of different methods with different parameters. For randomized algorithms, error bars show ± one standard deviation over ten runs. Lower-left region indicates better performance. Our proposed Core-D algorithm provided a better trade-off between speed and accuracy than LogPass. Specifically, Core-D (with Overall Model) was up to **12× faster** than LogPass ($\alpha = 2$), while still providing comparable accuracy. Among the models of Core-D, Overall Model yielded the best performance in most datasets.

Pass, whose memory requirement does not depend on α. The memory requirement of Core-D with Overall Model or Triangle Model was similar to that of LogPass, as seen in Figure 10. Specifically, Core-D with Overall Model or Triangle Model required 63-124% of the memory space required by LogPass. Core-D with Basic Model, which does not have to sample edges for estimating the triangle count, required the least memory space.

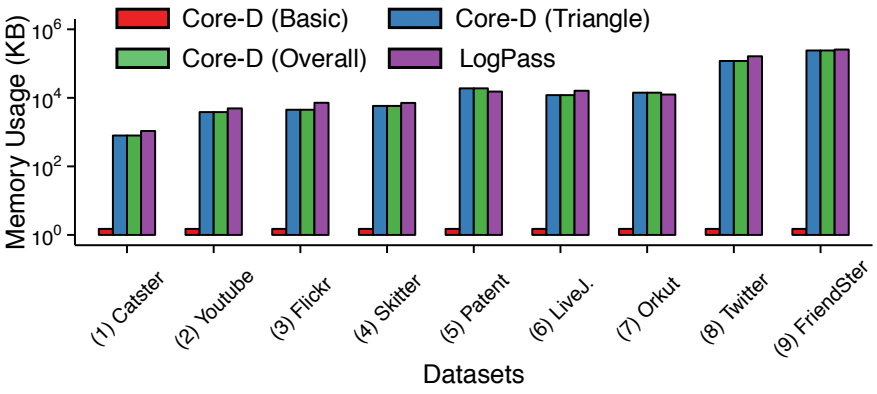


Fig. 10. **Core-D is comparable with LogPass in terms of memory requirements.** Core-D with Overall Model or Triangle Model has similar memory requirements to LogPass, and Core-D with Basic Model has the smallest memory requirements.

6. Pattern 3: "Structured Core Pattern"

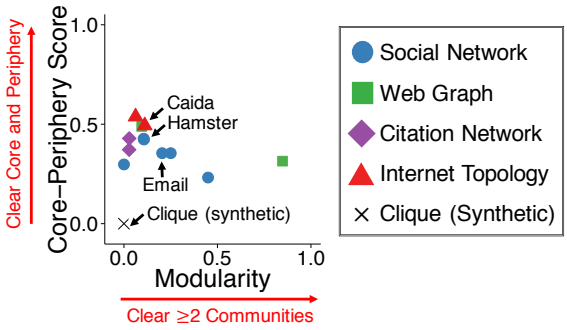
In this section, we describe Structured Core Pattern and discuss its application to influential spreader identification.

6.1. Observation: Pattern in Real-world Graphs

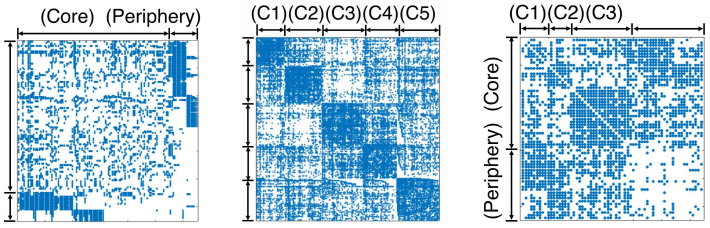
How do the degeneracy-cores in real-world graphs look like? Are they cliques? Our observation shows that degeneracy-cores in real-world graphs are not cliques but have structural patterns such as core-periphery (Borgatti and Everett, 2000) (i.e., have a cohesive core and a loosely connected periphery) and communities (Newman, 2006) (i.e., consist of groups of vertices with dense connections internally and sparser connections between groups). This leads to Observation 6.1, which is supported by the following facts:

- As shown in Table 2, degeneracy-cores have density much less than one in all the datasets (e.g., 0.02 in Friendster and 0.03 in Orkut) except LiveJournal and Twitter, whose degeneracy-cores include anomalies (see Section 4.2).
- In all the datasets, degeneracy-cores have significantly higher core-periphery score [9] (e.g., 0.54 in Skitter and 0.49 in Stanford) than cliques, as shown in Figure 11(a).
- Figure 11(a) also indicates that many datasets have significantly higher modularity [10] than cliques (e.g., 0.85 in NotreDame and 0.47 in Orkut).
- The sparsity patterns of the adjacency matrices of degeneracy-cores reveal

[9] Strength of core-periphery structure. The correlation between the adjacency matrix of the measured graph and that of a graph with perfect core-periphery structure. See (Borgatti and Everett, 2000) for details.
[10] Strength of community structure. The fraction of the edges within communities minus such fraction expected in a randomly connected graph. See (Newman, 2006) for details.



(a) Structural Property of Real-world Graphs



(b) Core-Periphery (Hamster) (c) Communities (Email) (d) Core-Periphery with Communities (Caida)

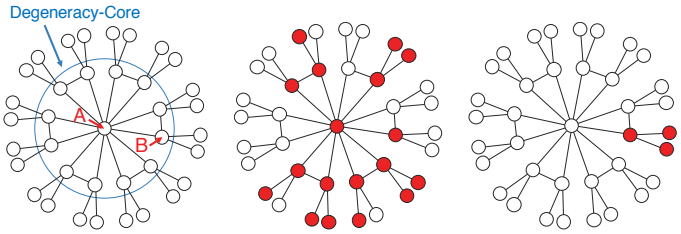
Fig. 11. **Degeneracy-cores of real-world graphs are not cliques but have structural patterns** such as core-periphery and communities. (a) Core-periphery score ($\in[0,1]$) and modularity ($\in[-0.5,1]$) measure the strength of core-periphery and community structure, resp., in graphs. (b), (c), and (d) show the sparsity patterns of the adjacency matrices of degeneracy-cores. Ci denotes the i-th community. See Appendix A for how to interpret sparsity patterns.

structural patterns such as core-periphery and communities. We explain sparsity patterns and the way to interpret them in Appendix A. Figures 11(b)-11(d) show the sparsity patterns of some real-world degeneracy-cores, where vertices are reordered as proposed in (Hooi et al, 2016b). In Figure 11(b), vertices in the degeneracy-core are clearly divided into the core and the periphery. In Figure 11(c), vertices are divided into five communities. In Figure 11(d), vertices are clearly divided into the core and the periphery, and the vertices in the core are again divided into three communities.

Observation 6.1 (Structured Core Pattern)**.** *In real-world graphs, degeneracy-cores have structural patterns such as core-periphery and communities.*

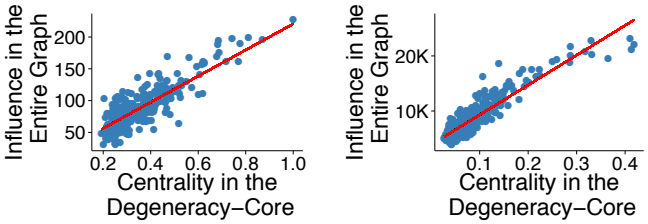
6.2. Application: Finding Influential Spreaders

The problem of identifying influential spreaders in social networks has gained considerable attention due to its wide applications, including information spreading, viral marketing, and epidemic disease control (see Section 8 for related work). For the problem of finding individual spreaders (instead of a set of spreaders,



(a) Example Graph (b) Influence of A (c) Influence of B

Fig. 12. **Intuition behind Core-S.** In (a), although both vertex A and vertex B have the highest coreness, vertex A, which is in the core of the degeneracy-core, has higher influence than vertex B, which is in the periphery of the degeneracy-core. (b) shows the infected vertices (colored red) when vertex A is used as the seed of SIR simulation, explained in Appendix C, with $\beta = 0.5$. (c) shows the same result when B is used as the seed.



(a) Email ($r = 0.84$) (b) Orkut ($r = 0.92$)

Fig. 13. **Vertices central in degeneracy-cores are influential in entire graphs.** 300 vertices randomly picked in the degeneracy-core of each graph are plotted. r denotes the Pearson correlation coefficient. Influence is measured using SIR Model simulation (see Appendix C), and in-core centrality (Definition 6.1) is used for measuring centrality.

which is another well-studied problem, called the influence maximization problem (Kempe et al, 2003)), it is shown in (Kitsak et al, 2010) that the ability of vertices to spread information to the large portion of a network is more closely related to their coreness rather than other centrality measures such as degree and betweenness centrality. This implies that the vertices in the degeneracy-core tend to be good spreaders,

Our Structured Core Pattern reveals that even vertices belonging to the degeneracy-core can be further divided into those in core and those in periphery; or those connecting communities and those inside a community. As suggested in the example in Figure 12, we observe that this position of a vertex within the degeneracy-core is highly related to its ability to spread information not just in the degeneracy-core but in the entire graph. Specifically, we find out a strong correlation between influence (see Appendix C for the measurement method) and in-core centrality, which we define in Definition 6.1, as shown in Figure 13.

Definition 6.1 (In-Core Centrality)**.** *Let $G'(V', E')$ be the degeneracy-core of*

Require: Graph: G, Number of spreaders: k ($\leq n_{max}$)
Ensure: k influential spreaders

1: run the core decomposition of G
2: extract the degeneracy-core $G'(V', E')$ from G
3: compute the in-core centrality of the vertices in V' by power iteration in G'
4: **return** top-k vertices with the highest in-core centralities

Algorithm 2: CORE-S for top-k spreaders

graph G, Then, for each vertex v in V', v's in-core centrality in G is defined as

$$i(v) \equiv v\text{'s eigenvector centrality in } G'.$$

Among many centrality measures, eigenvector centrality (i.e., entries of the eigenvector corresponding to the largest real eigenvalue) is used since it is computationally efficient and is known to be effective in identifying influential spreaders (Macdonald et al, 2012).

This observation is used to further refine influential spreaders in the degeneracy-core in the following section.

6.3. Core-S: Algorithm for Influential Spreader Identification

Inspired by STRUCTURED CORE PATTERN, we propose CORE-S, a top-k influential-spreader identification algorithm based on in-core centrality. We show that CORE-S gives a better trade-off between speed and accuracy than its top competitors.

6.3.1. Algorithm

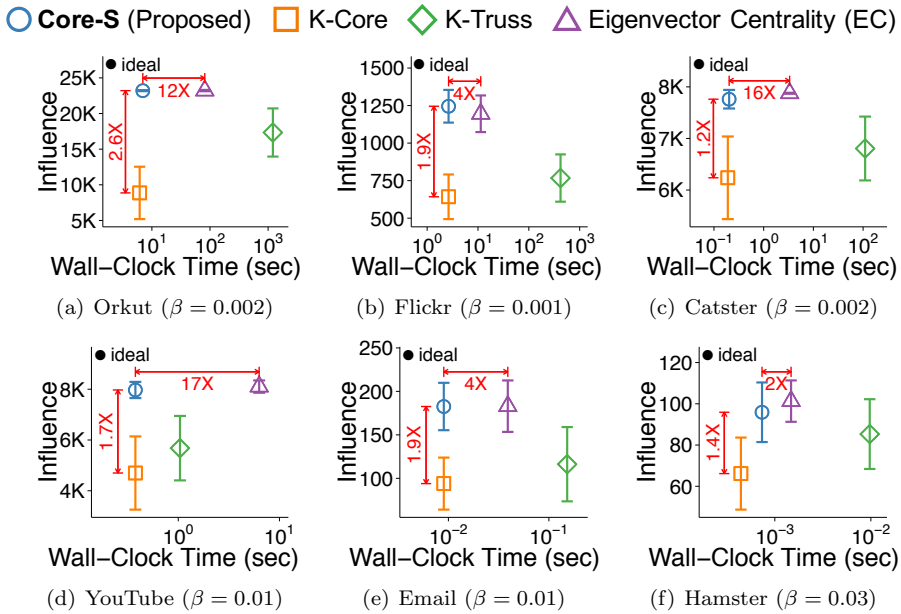
As outlined in Algorithm 2, CORE-S first runs core decomposition and extracts the degeneracy-core $G'(V', E')$. Then, the in-core centralities of the vertices in V' are computed using power iteration. As the last step, CORE-S returns the top-k vertices with the highest in-core centralities. The time complexity of CORE-S is $O(n + m + Tm_{max} + n_{max} \log k)$, where $(n + m)$ is for core decomposition, Tm_{max} is for power iteration, and $n_{max} \log k$ is for top-k selection. T denotes the number of iterations in the power iteration.

6.3.2. Experiments

The experimental settings were the same with those in Section 5.3.2. We compared the average influence of ten vertices chosen by CORE-S with that of the vertices chosen by the following methods:

- K-CORE (Kitsak et al, 2010): all vertices with the highest coreness.
- K-TRUSS (Rossi et al, 2015): all vertices with the highest trussness (defined in Section 2.1).
- Eigenvector Centrality (EC) (Macdonald et al, 2012): top-ten vertices with the highest eigenvector centralities in the entire graph.

The influence of each vertex was measured using SIR simulation (see Appendix C for details). We also compared the time taken for choosing influential vertices in each method.



(a) Orkut ($\beta = 0.002$) (b) Flickr ($\beta = 0.001$) (c) Catster ($\beta = 0.002$)

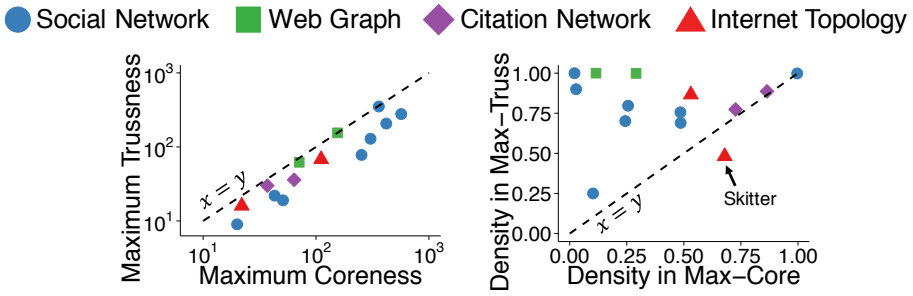
(d) YouTube ($\beta = 0.01$) (e) Email ($\beta = 0.01$) (f) Hamster ($\beta = 0.03$)

Fig. 14. **Core-S achieves both speed and accuracy.** β denotes the infection rate in SIR Model. Points in each plot represent the performances of different methods. Upper-left region indicates better performance. CORE-S provided the best trade-off between speed and accuracy. Specifically, it found up to **2.6× more influential** vertices than K-CORE with similar speed. Compared with EC, CORE-S was up to **17× faster**, while still finding vertices with comparable (95-104%) influence.

As seen in Figure 14, CORE-S provided the best trade-off between speed and accuracy in social networks. Specifically, the average influence of the vertices chosen by CORE-S was up to **2.6× higher** than that of all the vertices in the degeneracy-core (K-CORE). However, additional time taken in CORE-A for further refining vertices in degeneracy-cores was at most 12% of the time taken for the core decomposition of entire graphs in all the considered social networks except the smallest Hamster dataset. Besides, CORE-S was up to **17× faster**, than EC, which computes the eigenvector centrality in entire graphs (instead of only in degeneracy-cores). However, the average influence of the vertices chosen by CORE-S was comparable (95-104%) with that of the vertices found by EC.

7. Beyond k-Cores: Extension to k-Trusses

As discussed in the previous sections, k-cores are easy-to-compute and useful in many applications. However, finding k-cores has its limits as a way of detecting dense subgraphs since even degeneracy-cores are often far from cliques, as STRUCTURED CORE PATTERN suggests. Due to this limitation, the extensions of k-cores with more rigid definitions have been considered, although finding them requires more computation (see Section 8 for such extensions).



(a) Comparsion of maximum trusness and maximum trusness

(b) Comparison of highest-order cores and highest-order trusses

Fig. 15. Comparison between k-trusses and k-cores in real-world graphs. (a) In most graphs, the maximum trussness is much lower than the maximum coreness (i.e., degeneracy). (b) In most graphs, the highest-order truss is much more dense than the highest-order core (i.e., degeneracy-core)

.

In this section, we discuss k-trusses, a well-studied extension of k-cores. Especially, we use k-trusses to enhance CORE-A, our anomaly detection algorithm proposed in Section 4. Before that, we first show that MIRROR PATTERN, which motivates CORE-A, also exists in k-trusses in real-world graphs.

7.1. Comparison of k-Trusses and k-Cores

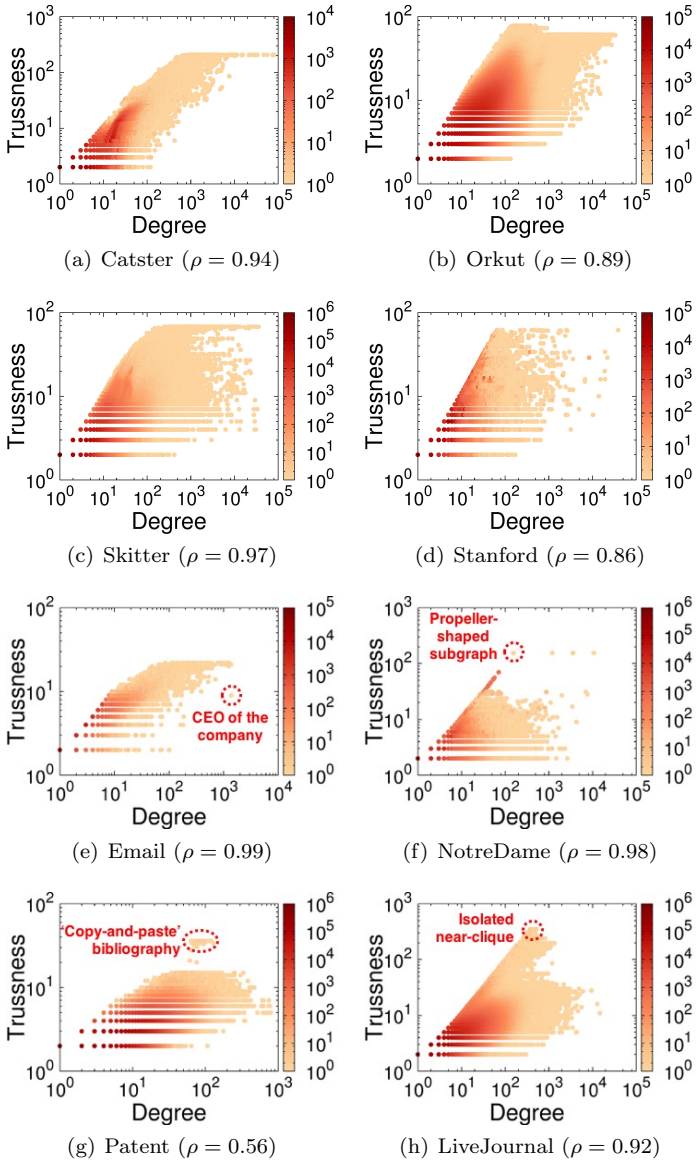
Note that the definition of the $(k+1)$-truss, given in Section 2.1, is similar to but more rigid than that of the k-core. This is because the fact that every edge is involved in at least $(k-1)$ triangles within the subgraph implies that every vertex has degree at least k within the subgraph, while the latter does not imply the former. Due to this rigidity, in most real-world graphs, the maximum trussness is much lower than the maximum coreness (i.e., degeneracy), and the highest-order truss is much denser than the highest-order core (i.e., degeneracy-core), as shown in Figure 15.

7.2. Mirror Pattern and Anomalies in k-Trusses

We discover a strong positive correlation between trussnesses of vertices and their degrees in real-world graphs. Specifically, as shown in Figure 16, Spearman's rank correlation coefficient ρ [11] is significantly higher than 0 (no correlation) in all the considered graphs and close to 1 (perfect positive correlation) in many of them. Especially, these correlation coefficients are similar to those between coreness and degree (see Figure 2 in Section 4). This empirical pattern is described in Observation 7.1.

Observation 7.1 (TRUSS MIRROR PATTERN)**.** *In real-world graphs, trussnesses of vertices have a strong positive correlation with their degrees.*

[11] Isolated vertices are ignored when we compute Spearman's rank correlation coefficient ρ.



(a) Catster ($\rho = 0.94$) (b) Orkut ($\rho = 0.89$)

(c) Skitter ($\rho = 0.97$) (d) Stanford ($\rho = 0.86$)

(e) Email ($\rho = 0.99$) (f) NotreDame ($\rho = 0.98$)

(g) Patent ($\rho = 0.56$) (h) LiveJournal ($\rho = 0.92$)

Fig. 16. **Truss Mirror Pattern in real-world graphs and exceptions indicating anomalies.** ρ ($\in [-1, 1]$) indicates Spearman's rank correlation coefficient; and colors are for heatmap of point density. Degree and trussness have strong positive correlation; exceptions in red circles indicate anomalies: the vertex ranked first in terms of degree but relatively lower in terms of trussness corresponds to an email account of the company's CEO in (e); vertices ranked first in terms of trussness but relatively lower in terms of degree indicate a propeller-shaped subgraph in (f), 'copy-and-paste' bibliography in (g), and an isolated near-clique in (h).

However, exceptions deviating Truss Mirror Pattern also exist. These exceptions indicate either *'loner-stars'* (i.e., vertices mostly connected to 'loners') or *'lockstep behavior'* (i.e., a group of similarly behaving vertices), which are the exceptions also deviating Mirror Pattern in Section 4.2.

7.3. Truss-A: Enhancing Core-A with Trussness

Motivated by Truss Mirror Pattern, we propose Truss-A, an anomaly detection algorithm enhancing Core-A with trussness. Truss-A shows higher accuracy at the expense of more computational cost than Core-A. The differences between Truss-A and Core-A are explained below.

Truss-A detects anomalies by ranking vertices in decreasing order of their deviation from Truss Mirror Pattern, which implies that high-degree vertices tend to have high trussness and vice versa. The way how Truss-A measures deviation from Truss Mirror Pattern (*dmp_truss*) is similar to the way how Core-A measures deviation from Mirror Pattern in Section 4.3. Truss-A first computes the rank of each vertex in terms of degree and the rank of each vertex in terms of trussness. Then, Truss-A computes the absolute difference of the log of the ranks, as formulated in Definition 7.1.

Definition 7.1 (Deviation from Truss Mirror Pattern)**.** *A vertex v's degree of deviation from* Truss Mirror Pattern *in graph G is defined as*

$$dmp\_truss(v) \equiv |\log(rank_d(v)) - \log(rank_t(v))|,$$

where $rank_d(v)$ is the fractional rank of v in decreasing order of degree and $rank_t(v)$ is that in decreasing order of trussness (in case of the same trussness, in decreasing order of degree).

Truss-A has time complexity $O(n + m^{1.5})$ since the *dmp_truss* scores of all vertices can be computed in $O(n)$ using 'counting sort' once we compute the degree and trussness of all vertices in $O(n + m^{1.5})$ (Wang and Cheng, 2012).

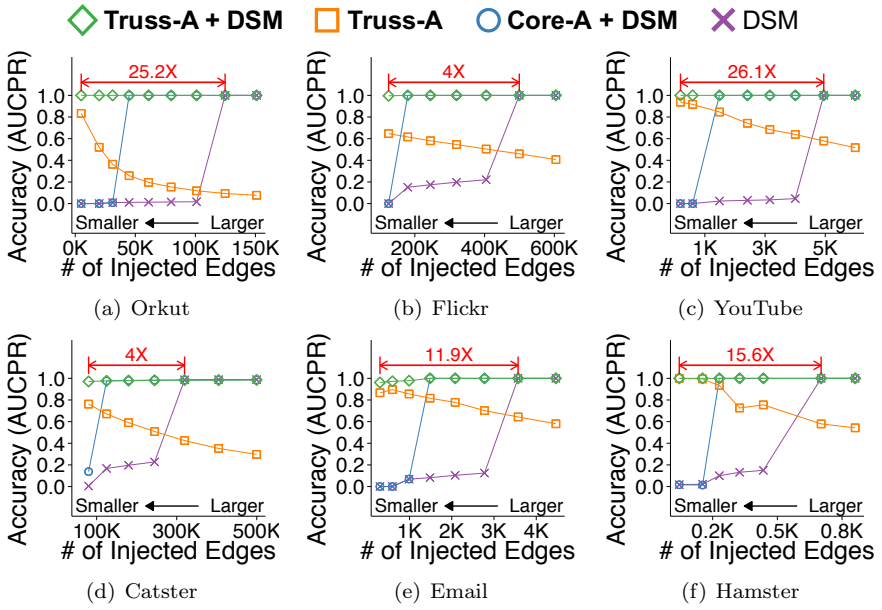
As we combine Core-A and DSM in Section 4.3, we combine Truss-A and DSM, which results in the best of Truss-A and DSM. Specifically, we define the *anomaly score with trussness* (*a-score-truss*) of a subgraph $G'(V', E')$ in a graph G using *dmp_truss* scores in G as well as the average degree as follows:

$$a\text{-}score\text{-}truss(G') = \frac{|E'|}{|V'|} + w \sum\nolimits_{v \in V'} \frac{dmp\_truss(v)}{|V'|}, \tag{12}$$

where the parameter $w > 0$ balances $\frac{|E'|}{|V'|}$ and $\sum_{v \in V'} \frac{dmp\_truss(v)}{|V'|}$. We set w to

$$w = \frac{|E^*|}{|V^*|} \times \frac{1.2}{\max_{v \in V} dmp\_truss(v)},$$

where $G^*(V^*, E^*)$ is the densest subgraph detected by DSM. Compared to Eq. (2), we put more emphasis on Truss-A than DSM, and this resulted in higher accuracy than giving them the same weight by setting $w = (|E^*|/|V^*|)$ $/(\max_{v \in V} dmp\_truss(v))$, in our experiments. Once w is set, Truss-A uses (Shin et al, 2016a) to identify the subgraph maximizing *a-score-truss* (Eq. (12)) and classifies the vertices in the subgraph into anomalies. This entire process takes $O(n + m^{1.5} + m \log n)$, where $O(n + m^{1.5})$ is the time complexity of Truss-A and $O(m \log n)$ is that of DSM (Shin et al, 2016a; Hooi et al, 2016a).



(a) Orkut (b) Flickr (c) YouTube

(d) Catster (e) Email (f) Hamster

Fig. 17. **Truss-A improves Core-A in terms of accuracy.** In social networks, the combination of Truss-A and DSM detects both small and large injected subgraphs with near-perfect accuracy, outperforming the other methods. The combination detects up to **9**× smaller injected subgraphs than the combination of Core-A and DSM and up to **26**× smaller injected subgraphs than DSM.

To demonstrate that Truss-A improves Core-A, we compare their accuracy in social networks where anomalies are injected. Specifically, we randomly choose k vertices and inject $\binom{k}{2}$ edges among them into each network. Then, we compare how precisely and exhaustively each method detects the k chosen vertices using Area Under the Precision-Recall Curve (AUCPR) (Davis and Goadrich, 2006). As seen in Figure 17, the combination of Truss-A and DSM detects both small and large dense subgraphs, clearly outperforming the other methods including individual Truss-A and DSM. Specifically, the combination detects up to **9**× smaller injected subgraphs than the combination of Core-A and DSM and up to **26**× smaller injected subgraphs than DSM, with near-perfect accuracy.

Although Truss-A is more accurate than Core-A, Truss-A requires more computation than Core-A. The time complexity of Truss-A is $O(n + m^{1.5})$, while that of Core-A is $O(n+m)$. The running times of different anomaly detection methods in social networks are compared in Figure 18 [12]. The combination of Truss-A and DSM took 9-1800× longer than the combination of Core-A and DSM; and took 17-3500× longer than individual Core-A or DSM.

Core-A and Truss-A provide a trade-off between speed and accuracy. That is, Core-A is faster than Truss-A, while Truss-A is more accurate than Core-A, when each of them is combined with DSM. A rule of thumb is to use Core-A for billion-scale or larger graphs and to use Truss-A for smaller graphs. An

[12] We used a machine with 2.67 GHz Intel Xeon E7-8837 CPUs and 1TB RAM.

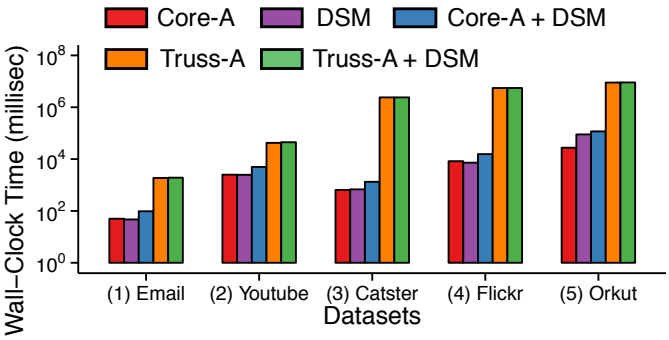


Fig. 18. **Core-A is faster than Truss-A.** Core-A (and its combination with DSM) is significantly faster than Truss-A (and its combination with DSM) although Truss-A is more accurate than Core-A.

interesting direction for future research is to design an algorithm achieving both speed and accuracy.

8. Related Work

Related work forms the following groups: applications of k-core analysis, algorithms for k-core analysis, dense subgraphs, graph-based anomaly detection, and influential spreader identification.

Applications of k-core Analysis. The concept of a k-core (Seidman, 1983) has been applied to hierarchical structure analysis (Alvarez-Hamelin et al, 2008), graph visualization (Alvarez-Hamelin et al, 2006), densest subgraph detection (Charikar, 2000) (a special case of DSM in Section 4.3.2), important protein identification (Wuchty and Almaas, 2005), influential spreader detection (Kitsak et al, 2010) (K-Core method in Section 6.3.2), and graph clustering (Giatsidis et al, 2014). Degeneracy also has been used as a graph-sparsity measure in many domains such as AI (Freuder, 1982) and Bioinformatics (Bader and Hogue, 2003).

Algorithms for k-core Analysis. Core decomposition can be computed in $O(n+m)$ by repeatedly removing vertices with the smallest degree (Batagelj and Zaversnik, 2003). (Saríyüce et al, 2013) proposed an incremental algorithm, while (Cheng et al, 2011) proposed an external memory algorithm, which requires $O(k_{max})$ scans of graphs. For degeneracy, (Farach-Colton and Tsai, 2014) proposed a streaming algorithm requiring $O(\log_{\alpha/2}(n))$ passes of a graph and n memory space for $\alpha(>2)$-approximation. Our Core-D, however, requires only one pass of a graph and n memory space for accurately estimating degeneracy.

Dense Subgraphs. In addition to k-cores, many notions of dense subgraphs have been proposed. The most strict one is a maximal clique (Bron and Kerbosch, 1973) (i.e, a complete subgraph not included in any other complete subgraphs). Since the definition of a clique is too rigid for many purposes, many relaxed forms have been proposed including n-cliques (Luce, 1950), k-plexes (Seidman and Foster, 1978), n-clans (Mokken, 1979), n-clubs (Mokken, 1979), and quasi-cliques (Abello et al, 2002). However, the computation of these dense subgraphs is NP-hard, while finding k-cores (i.e., core decomposition) runs in $O(n+m)$ (Batagelj and Zaversnik, 2003). The notion of a k-core also has been generalized

(Cohen, 2008; Saríyüce et al, 2015). A well studied one is a k-truss (Cohen, 2008), which is the maximal subgraph where every edge is contained in $(k-2)$ triangles within the subgraph. Finding all k-trusses, called truss decomposition, runs in $O(n+m^{1.5})$ (Wang and Cheng, 2012). Recently, k-trusses in probabilistic graphs were also explored (Huang et al, 2016).

Graph-based Anomaly Detection. There have been diverse approaches (belief propagation (Pandit et al, 2007), egonet features (Akoglu et al, 2010), spectral methods (Prakash et al, 2010), etc.) for anomaly detection in graphs (see (Akoglu et al, 2015) for a survey). Recently, many methods focus on dense subgraphs, which anomalies tend to form (Shin et al, 2016a; Shin et al, 2017a; Shin et al, 2017b; Hooi et al, 2016a; Beutel et al, 2013; Jiang et al, 2015), and their hierarchies (Zhang et al, 2017). Especially, the latest methods (Shin et al, 2016a; Shin et al, 2017a; Shin et al, 2017b; Hooi et al, 2016a) are based on densest subgraphs (i.e., subgraphs with maximum average degree). We show that our Core-A and Truss-A, which detect smaller dense subgraphs consisting of low-degree vertices, are complementary to these densest-subgraph based methods, and the combination of both approaches has the best of both approaches.

Influential Spreader Identification. The problem of identifying influential spreader is sub-categorized into (1) finding a group of spreaders, which is called the influence maximization problem (Kempe et al, 2003), and (2) finding individual influential spreaders. For the second problem, on which we focus, vertices with high coreness (Kitsak et al, 2010), truss number (Rossi et al, 2015), and eigenvector centrality (Macdonald et al, 2012) are known as good spreaders. Our Core-S combines these measures so that only the advantages of each measure (i.e., low computational cost of coreness and high accuracy of eigenvector centrality) are taken.

9. Conclusion

We discover three empirical patterns in real-world graphs related to k-cores, and utilize them for several applications.

Mirror Pattern and Core-A (Section 4): We observe a strong correlation between the coreness and the degree of vertices. Core-A, which measures the deviation from this trend, successfully detects anomalies in real-world graphs and complements a state-of-the-art anomaly detection method.

Core-Triangle Pattern and Core-D (Section 5): We discover a 3-to-1 power law between degeneracy and triangle count. Our Core-D method uses this pattern for accurately estimating degeneracy in only one pass of a graph stream and up to *12*× *faster* than a recent multi-pass method.

Structured Core Pattern and Core-S (Section 6): We observe that degeneracy-cores have non-trivial structures (core-periphery, communities, etc). Core-S, which finds vertices central within degeneracy-cores, identifies influential spreaders up to *17*× *faster* than methods with similar accuracy.

Acknowledgements. This material is based upon work supported by the National Science Foundation under Grant No. CNS-1314632 and IIS-1408924. Research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-09-2-0053. Kijung Shin was supported by KFAS Scholarship. Tina Eliassi-Rad was supported by NSF CNS-1314603 and by DTRA HDTRA1-10-1-0120. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the

views of the National Science Foundation, or other funding parties. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

Abello J, Resende MG, Sudarsky S (2002) Massive quasi-clique detection. InLatin American Symposium on Theoretical Informatics, Springer, pp 598–612

Akoglu L, McGlohon M, Faloutsos C (2010) Oddball: Spotting anomalies in weighted graphs. Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer, pp 410–421

Akoglu L, Tong H, Koutra D (2015) Graph based anomaly detection and description: a survey. Data Mining and Knowledge Discovery 29(3):626–688

Albert R, Jeong H, Barabsi AL (1999) Internet: Diameter of the world-wide web. nature 401(6749):130–1.

Alvarez-Hamelin JI, Dall'Asta L, Barrat A, Vespignani A (2006) Large scale networks fingerprinting and visualization using the k-core decomposition. Advances in neural information processing systems, Curran Associates, Inc., pp 41–50

Alvarez-Hamelin JI, Dall'Asta L, Barrat A, Vespignani A (2008) K-core decomposition of Internet graphs: hierarchies, self-similarity and measurement biases. Networks and Heterogeneous Media 3:371

Bader GD, Hogue CW (2003) An automated method for finding molecular complexes in large protein interaction networks. BMC bioinformatics 4(1): 2

Batagelj V, Zaversnik M (2003) An o(m) algorithm for cores decomposition of networks. arXiv preprint cs.DS/0310049

Beutel A, Xu W, Guruswami V, Palow C, Faloutsos C (2013) Copycatch: stopping group attacks by spotting lockstep behavior in social networks. Proceedings of the 22nd international conference on World Wide Web, ACM, pp 119–130

Borgatti SP, Everett MG (2000) Models of core/periphery structures. Social networks 21(4):375–395

Bron C, Kerbosch J (1973) Algorithm 457: finding all cliques of an undirected graph. Communications of the ACM 16(9):575–7.

Brouwer AE, Haemers WH (2001) Spectra of graphs. Springer Science & Business Media

Charikar M (2000) Greedy approximation algorithms for finding dense components in a graph. International Workshop on Approximation Algorithms for Combinatorial Optimization, Springer, pp 84–95

Cheng J, Ke Y, Chu S, Özsu MT (2011) Efficient core decomposition in massive networks. 2011 IEEE 27th International Conference on Data Engineering, IEEE, pp 51–62

Cohen J (2008) Trusses: Cohesive subgraphs for social network analysis. National Security Agency Technical Report, p 16

Davis J, Goadrich M (2006) The relationship between precision-recall and roc curves. Proceedings of the 23rd international conference on Machine learning, ACM, pp 233–240

De Stefani L, Epasto A, Riondato M, Upfal E (2016) TRIÈST: Counting Local and Global Triangles in Fully-Dynamic Streams with Fixed Memory Size. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, pp 825–834

Erdös P (1963) On the structure of linear graphs. Israel Journal of Mathematics 1(3):156–160

Farach-Colton M, Tsai MT (2014) Computing the degeneracy of large graphs. Latin American Symposium on Theoretical Informatics, Springer, pp 250–260

Freuder EC (1982) A sufficient condition for backtrack-free search. Journal of the ACM (JACM) 29(1):24–32

Gehrke J, Ginsparg P, Kleinberg J (2003) Overview of the 2003 KDD Cup, ACM SIGKDD Explorations Newsletter 5(2):149–51.

Giatsidis C, Malliaros F, Thilikos DM, Vazirgiannis M (2014) Corecluster: A degeneracy based graph clustering framework Twenty-Sixth Annual Conference on Innovative Applications of Artificial Intelligence, AAAI, pp 29–31

Hall BH, Jaffe AB, Trajtenberg M (2001) The NBER patent citation data file: Lessons, insights and methodological tools. National Bureau of Economic Research

Hooi B, Song HA, Beutel A, Shah N, Shin K, Faloutsos C (2016a) Fraudar: Bounding graph

fraud in the face of camouflage. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, pp 895–904

Hooi B, Song HA, Papalexakis E, Agrawal R, Faloutsos C (2016b) Matrices, Compression, Learning Curves: formulation, and the GROUPNTEACH algorithms. Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer, pp 376–387

Huang X, Lu W, Lakshmanan LV (2016) Truss decomposition of probabilistic graphs: Semantics and algorithms. Proceedings of the 2016 ACM SIGMOD international conference on Management of data, ACM, pp 77–90

Jiang M, Beutel A, Cui P, Hooi B, Yang S, Faloutsos C (2015) A general suspiciousness metric for dense blocks in multimodal data. 2015 IEEE International Conference on Data Mining, IEEE, pp 781–786

Kempe D, Kleinberg J, Tardos É (2003) Maximizing the spread of influence through a social network. Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, pp 137–146

Kitsak M, Gallos LK, Havlin S, Liljeros F, Muchnik L, Stanley HE, Makse HA (2010) Identification of influential spreaders in complex networks. Nature Physics 6(11):888–893

Klimt B, Yang Y (2004) The enron corpus: A new dataset for email classification research, European Conference on Machine Learning, Springer, pp 217–226

Kwak H, Lee C, Park H, Moon S (2010) What is Twitter, a social network or a news media?. Proceedings of the 19th international conference on World wide web, ACM, pp 591–600

Leskovec J, Chakrabarti D, Kleinberg J, Faloutsos C (2005) Realistic mathematically tractable graph generation and evolution, using kronecker multiplication. European Conference on Principles of Data Mining and Knowledge Discovery, Springer, pp 133–145

Leskovec J, Lang KJ, Dasgupta A, Mahoney MW (2009) Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. Internet Mathematics 6(1):29–123

Lim Y, Kang U (2015) Mascot: Memory-efficient and accurate sampling for counting local triangles in graph streams. Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, pp 685–694

Luce RD (1950) Connectivity and generalized cliques in sociometric group structure. Psychometrika 15(2):169–90

Macdonald B, Shakarian P, Howard N, Moores G (2012) Spreaders in the network sir model: An empirical study. arXiv preprint arXiv:1208.4269

Mislove A, Marcon M, Gummadi KP, Druschel P, Bhattacharjee B (2007) Measurement and analysis of online social networks. Proceedings of the 7th ACM SIGCOMM conference on Internet measurement, ACM, pp 29–42

Mokken RJ (1979) Cliques, clubs and clans. Quality & Quantity 13(2):161–73

Newman ME (2006) Modularity and community structure in networks. Proceedings of the national academy of sciences 103(23):8577–8582

Pandit S, Chau DH, Wang S, Faloutsos C (2007) Netprobe: a fast and scalable system for fraud detection in online auction networks. Proceedings of the 16th international conference on World Wide Web, ACM, pp 201–210.

Prakash BA, Sridharan A, Seshadri M, Machiraju S, Faloutsos C (2010) Eigenspokes: Surprising patterns and scalable community chipping in large graphs. Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer, pp 435–448

Rossi MEG, Malliaros FD, Vazirgiannis M (2015) Spread it good, spread it fast: Identification of influential nodes in social networks. Proceedings of the 24th International Conference on World Wide Web (Companion Volume), ACM, pp 101–102

Saríyüce AE, Gedik B, Jacques-Silva G, Wu KL, Çatalyürek ÜV (2013) Streaming algorithms for k-core decomposition. Proceedings of the VLDB Endowment 6(6):433–444

Saríyüce AE, Seshadhri C, Pinar A, Catalyurek UV (2015) Finding the hierarchy of dense subgraphs using nucleus decompositions. Proceedings of the 24th International Conference on World Wide Web, ACM, pp 927-937

Schank T (2007) Algorithmic aspects of triangle-based network analysis. PhD thesis, Universitt Karlsruhe (TH), Fakultt fr Informatik

Seidman SB, Foster BL (1978) A graphtheoretic generalization of the clique concept. Journal of Mathematical sociology 6(1):139–54

Seidman SB (1983) Network structure and minimum degree. Social networks 5(3):269–87

Shin K, Hooi B, Faloutsos C (2016a) M-zoom: Fast dense-block detection in tensors with quality guarantees. Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, pp 264–280

(a) Sparsity Pattern (b) Core-Periphery in the Sparsity Pattern (c) Communities in the Sparsity Pattern

Fig. 19. **Example sparsity pattern and its interpretation.**

Shin K, Eliassi-Rad T, Faloutsos C (2016b) Corescope: Graph mining using k-core analysis - patterns, anomalies and algorithms. 2016 16th IEEE International Conference on Data Mining, IEEE, pp 469–478

Shin K, Hooi B, Jisu K, Faloutsos C (2017a) D-cube: Dense-block detection in terabyte-scale tensors. Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, ACM, pp 681–690

Shin K, Hooi B, Jisu K, Faloutsos C (2017b) Densealert: Incremental dense-subtensor detection in tensor streams. arXiv preprint arXiv:1706.03374

Spearman C (1904) The proof and measurement of association between two things. The American journal of psychology 15(1):72–101

Tsourakakis CE (2008) Fast counting of triangles in large real networks without counting: Algorithms and laws. 2008 Eighth IEEE International Conference on Data Mining, IEEE, pp 608–617

Tsourakakis CE, Kang U, Miller GL, Faloutsos C (2009) Doulion: counting triangles in massive graphs with a coin. Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, pp 837–846

Van Loan CF (2000) The ubiquitous kronecker product. Journal of computational and applied mathematics 123(1):85–100

Wang J, Cheng J (2012) Truss decomposition in massive networks. Proceedings of the VLDB Endowment 5(9):812–23

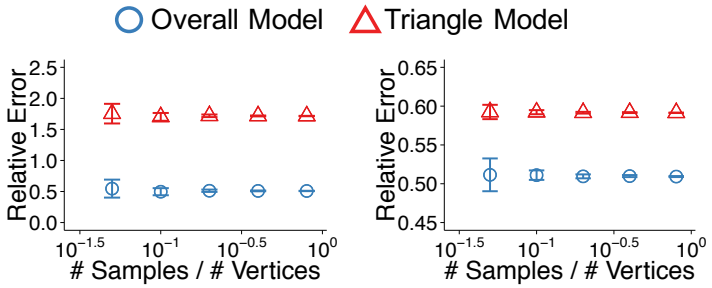
Wuchty S, Almaas E (2005) Peeling the yeast protein network. Proteomics 5(2):444–449

Zhang S, Zhou D, Yildirim MY, Alcorn S, He J, Davulcu H, Tong H (2017) HiDDen: Hierarchical Dense Subgraph Detection with Application to Financial Fraud Detection. Proceedings of the 2017 SIAM International Conference on Data Mining, SIAM, pp 570–578

A. Interpreting Sparsity Patterns

We explain sparsity patterns and how to interpret them. The sparsity pattern of a graph is a plot with the axes representing the rows and columns of the adjacency matrix. For each non-zero entry (i.e., edge in the graph), a point is plotted, thus displaying sparsity patterns in the adjacency matrix.

Figure 19(a) shows the sparsity pattern of the degeneracy-core of Caida Dataset. The rows in the plot indicate vertices, and they are divided into two ranges, which correspond to the core and the periphery. The vertices in the core are densely connected with each other, as seen in region A in Figure 19(b). The vertices in the periphery are well connected to the vertices in the core (regions B and C) but rarely connected to each other (region D). The vertices in the core are further divided into three communities, each of which corresponds to a range of the columns in Figure 19(a). The vertices in the same community are



(a) Friendster (b) Twitter

Fig. 20. **Core-D is nimble and accurate.** Points and error bars represent the average accuracy and ± one standard deviation over ten runs, respectively. Core-D reliably estimates degeneracy even with a small number of samples less than the number of vertices.

particularly well connected to each other, as seen in regions A1, A2, and A3 in Figure 19(c), which correspond to the sparsity patterns of the communities.

B. Core-D with a Small Number of Samples

Figure 20 presents the accuracy of Core-D with different sample sizes in the two largest datasets. Even with a small number of samples less than the number of vertices, Core-D, especially Overall Model, accurately and reliably estimated degeneracy. Thus, Core-D is still effective even when the amount of available memory space is less than n.

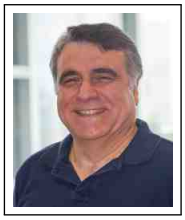
C. Measuring Influence using SIR Model Simulation

To evaluate influence as a spreader, we simulate spreading processes using SIR Model (Kitsak et al, 2010), a widely-used epidemic model. Initially, a vertex chosen as the seed is in the *infectious state* (I-state), while the others are in the *susceptible state* (S-state). Each vertex in the I-state infects each of its neighbors in the S-state with probability β (infection rate) and then enters the *recovered state* (R-state). This is repeated until no vertex is in the I-state. The influence of a seed, the initially infected vertex, can be quantified by the number of vertices infected at any time during the process. To reduce random effects, we repeat the whole process 100 times, and use the average number of infected vertices as the measure of influence. β is set close to the epidemic threshold λ_1^{-1}, as in previous work (Rossi et al, 2015).

Author Biographies

Kijung Shin is a Ph.D. student in the Computer Science Department of Carnegie Mellon University. He received B.S. in Computer Science and Engineering at Seoul National University. His research interests include graph mining and scalable machine learning.

Tina Eliassi-Rad is an Associate Professor of Computer Science at Northeastern University in Boston, MA. She is also on the faculty of Northeastern's Network Science Institute. Prior to joining Northeastern, Tina was an Associate Professor of Computer Science at Rutgers University; and before that she was a Member of Technical Staff and Principal Investigator at Lawrence Livermore National Laboratory. Tina earned her Ph.D. in Computer Sciences (with a minor in Mathematical Statistics) at the University of Wisconsin-Madison. Her research is rooted in data mining and machine learning; and spans theory, algorithms, and applications of massive data from networked representations of physical and social phenomena. Tina's work has been applied to personalized search on the World-Wide Web, statistical indices of large-scale scientific simulation data, fraud detection, mobile ad targeting, and cyber situational awareness.



Christos Faloutsos is a Professor at Carnegie Mellon University. He has received the Presidential Young Investigator Award by the National Science Foundation (1989), the Research Contributions Award in ICDM 2006, the SIGKDD Innovations Award (2010), 24 "best paper" awards (including 5 "test of time" awards), and four teaching awards. Six of his advisees have attracted KDD or SCS dissertation awards, He is an ACM Fellow, he has served as a member of the executive committee of SIGKDD; he has published over 350 refereed articles, 17 book chapters and two monographs. He holds seven patents (and 2 pending), and he has given over 40 tutorials and over 20 invited distinguished lectures. His research interests include large-scale data mining with emphasis on graphs and time sequences; anomaly detection, tensors, and fractals.

Correspondence and offprint requests to: Kijung Shin, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213, USA. Email: kijungs@cs.cmu.edu