CoreScope: Graph Mining Using k-Core Analysis - Patterns, Anomalies and Algorithms

Kijung Shin
Carnegie Mellon University
Pittsburgh, PA, USA
kijungs@cs.cmu.edu

Tina Eliassi-Rad
Northeastern University
Boston, MA, USA
eliassi@ccs.neu.edu

Christos Faloutsos
Carnegie Mellon University
Pittsburgh, PA, USA
christos@cs.cmu.edu

***Abstract*—How do the k-core structures of real-world graphs look like? What are the common patterns and the anomalies? How can we use them for algorithm design and applications? A k-core is the maximal subgraph where 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 that are related to k-cores and emerging in graphs from several diverse domains.**

Our discoveries are as follows: (1) MIRROR PATTERN: coreness of vertices (i.e., maximum k such that each vertex belongs to the k-core) is strongly correlated to their degree. (2) CORE-TRIANGLE PATTERN: degeneracy of a graph (i.e., maximum k such that the k-core exists in the graph) 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 finds anomalies in real-world graphs complementing densest-subgraph based anomaly detection methods. (2) CORE-D, a single-pass streaming algorithm based on CORE-TRIANGLE PATTERN, accurately estimates the degeneracy of billion-scale graphs up to 7× faster than a recent multi-pass algorithm. (3) CORE-S, inspired by STRUCTURED CORE PATTERN, identifies influential spreaders up to 17× faster than top competitors with comparable accuracy.

***Index Terms*—Graphs, k-cores, degeneracy, influential nodes, anomaly detection**

I. 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 [1]. As discussed in Section VI, this concept has been used extensively in diverse applications, including hierarchical structure analysis [2], graph visualization [3], protein function prediction [4], and graph clustering [5]. 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 node 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 better 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 use of these patterns.

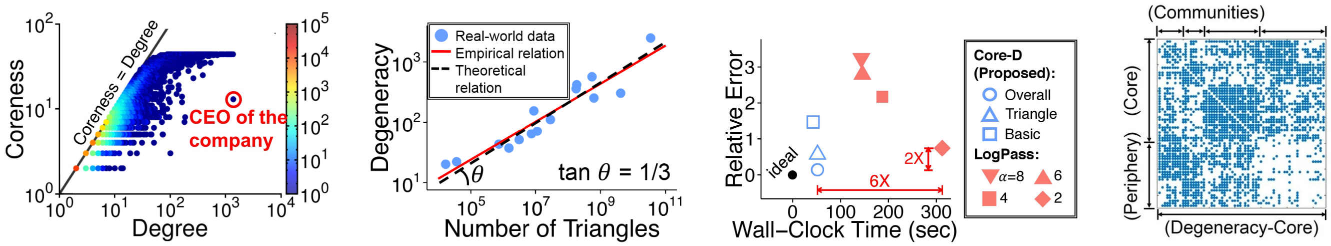
Our first MIRROR PATTERN states that the *coreness* of a vertex (i.e., the maximum k such that the vertex belongs to the k-core) is strongly correlated to its degree, as seen in Figure 1(a). We also observe that anomalies (e.g., the CEO in Figure 1(a) and accounts using 'follower-booster' in Twitter) 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 [6], [7], 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 [8], and also utilized in CORE-D, our single-pass streaming algorithm for estimating degeneracy. CORE-D is up to 7× *faster* than a recent multi-pass algorithm [9], 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 [10]. Those spreaders are spotted by CORE-S, our influential spreader identification method, which is up to *17× faster* than top competitors with similar accuracy.

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

- **Patterns**: We discover three empirical patterns that hold across several real-world graphs from diverse domains.
- **Anomalies**: We detect various interesting anomalies (e.g., accounts involved in a 'follower-boosting' service in Twitter) from vertices deviating from the patterns.



(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 **6× faster and 2× more accurately** than its state-of-the-art competitor. (d) **P3**: As seen in the sparsity pattern of the given degeneracy-core, degeneracy-cores have structure, such as core-periphery and communities, which can be exploited for identifying influential spreaders (**A3**).

- **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 complement or outperform state-of-the-art algorithms.

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

In Section II, we give preliminaries on k-cores. In Section III, we present MIRROR PATTERN and its application to anomaly detection. In Section IV, we describe CORE-TRIANGLE PATTERN and CORE-D, a streaming algorithm for estimating degeneracy. STRUCTURED CORE PATTERN and its application to influential spreader detection are presented in Section V. After discussing related work in Section VI, we make a conclusion in Section VII.

II. PRELIMINARIES

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

A. 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 [1] 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. In addition, cores are nested. 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 [1], denoted by $c(v)$, is the order of the highest-order core that v belongs to. A vertex v has coreness k iff v belongs to the k-core but not to the $(k+1)$-core. 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 *degeneracy-core*. If we let n_{max} and m_{max} be the number of vertices and that of edges in the degeneracy-core, the density of the degeneracy-core is defined as $D_{max}=m_{max}/\binom{n_{max}}{2}$.

TABLE I: Table of symbols.

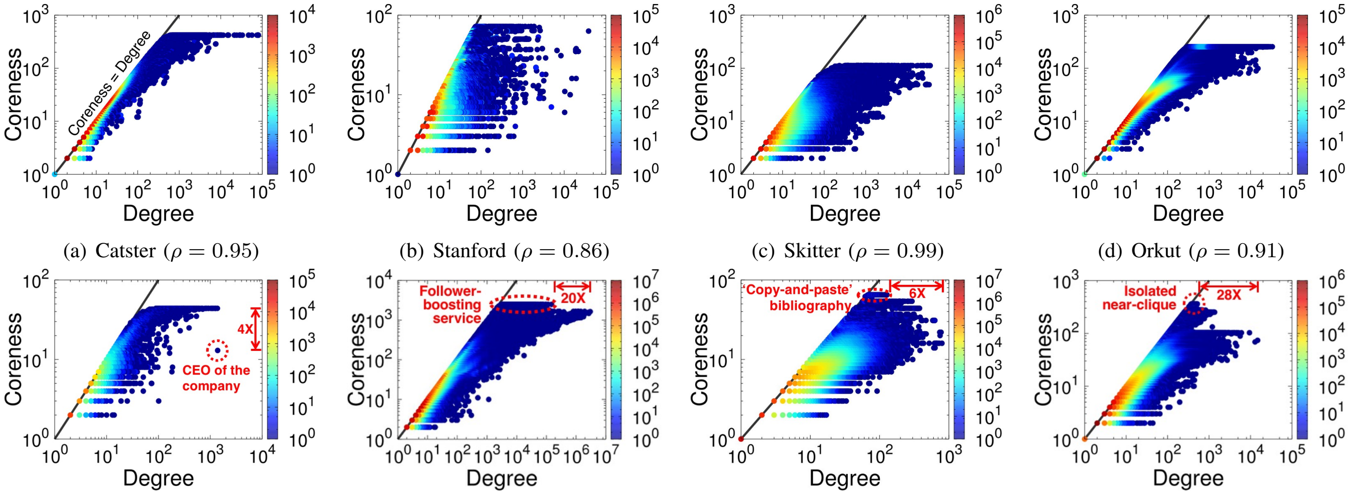
Symbol	**Definition**
$G(V,E)$	undirected and unweighted graph
A	adjacency matrix of G
n	number of vertices in G
m	number of edges in G
k_{max}	degeneracy of G
n_{max}	number of vertices in the degeneracy-core
m_{max}	number of edges in the degeneracy-core
D_{max}	density of the degeneracy-core
d_{avg}	average degree of G
$c(v)$	coreness of vertex v
$d(v)$	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

Additionally, we denote the number of triangles (i.e., complete subgraphs with three vertices) 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 I lists the symbols frequently used in the paper.

B. 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 $O(n+m)$ [1] 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 [9] requires $O(\log_{\alpha/2}(n))$ passes of the entire graph and n memory space for α-approximation of the degeneracy. In Section IV-C, however, we propose a single-pass algorithm for estimating degeneracy. Other k-core algorithms for large graphs are discussed in Section VI.



(a) Catster ($\rho = 0.95$) (b) Stanford ($\rho = 0.86$) (c) Skitter ($\rho = 0.99$) (d) Orkut ($\rho = 0.91$)

(e) Email ($\rho = 0.99$) (f) Twitter ($\rho = 0.95$) (g) Patent ($\rho = 0.56$) (h) LiveJournal ($\rho = 0.93$)

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 (e); vertices ranked first in terms of coreness but relatively lower in terms of degree indicate accounts involved in a 'follower-boosting' service in (f), 'copy-and-paste' bibliography in (g), and an isolated near-clique in (h).

TABLE II: Summary of the datasets used in the paper. All graphs are considered undirected and unweighted.

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

III. Pattern 1: "Mirror Pattern"

In this section, we describe Mirror Pattern and its application to anomaly detection. Table II lists the datasets we use in this work, with more details about them in Appendix A.

A. Observation: Pattern in Real-world Graphs

What are the key factors determining the coreness of the vertices in real graphs? We find out a strong positive correlation between coreness and degree, which is an upper bound of coreness. As seen in Figure 2, Spearman's rank correlation coefficient ρ is significantly higher than 0 (no correlation) in all the considered graphs and close to 1 (perfect correlation) in many of them. Isolated vertices are ignored when computing ρ. This empirical pattern is described in Observation 1.

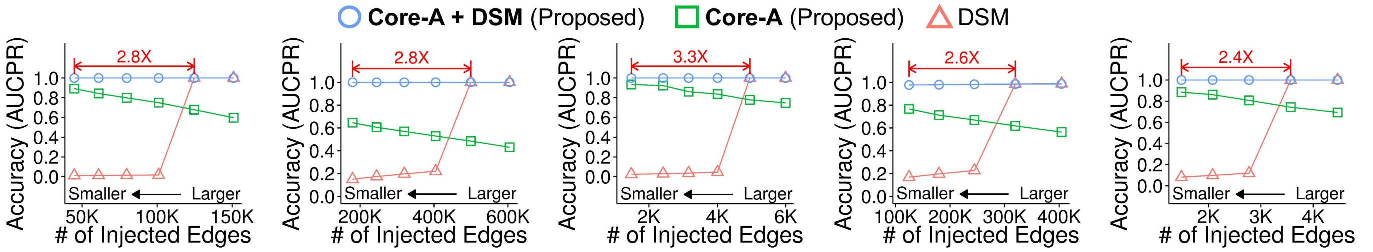
Observation 1. (Mirror Pattern) *In real-world graphs, coreness has a strong positive correlation with degree.*

B. 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).

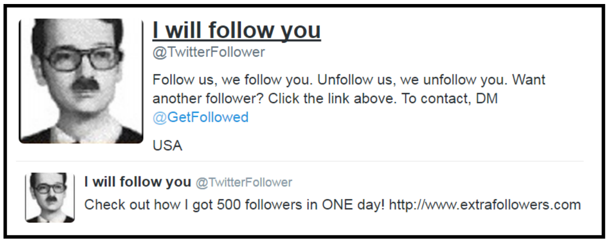
1) Second Email Account of the CEO (Loner-Star): In the Email dataset, the vertex marked in Figure 2(e) has the highest degree 1,383 but relatively low coreness 12, deviating from Mirror Pattern. This vertex corresponds to the second email account of the former CEO of the company. This account was used 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.

2) 'Follower-Boosting' Service in Twitter (Lockstep Behavior): In Twitter, the vertices with the highest coreness, marked in Figure 2(f), 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 4) 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 followers,



(a) Orkut (b) Flickr (c) YouTube (d) Catster (e) Email

Fig. 3: **Core-A is complementary to DSM; their combination has the best of the two.** In social networks, our Core-A method accurately detects small dense subgraphs that cannot be detected accurately by DSM. The combination of Core-A and DSM successfully detects both small and large subgraphs. The combination detects up to **3.3× smaller** subgraphs than DSM with near-perfect accuracy.



I will follow you
@TwitterFollower

Follow us, we follow you. Unfollow us, we unfollow you. Want another follower? Click the link above. To contact, DM @GetFollowed

USA

I will follow you @TwitterFollower
Check out how I got 500 followers in ONE day! http://www.extrafollowers.com

Fig. 4: **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.

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.

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(g)). 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) 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(h). 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. (3) The vertices have suspicious uniformity. Specifically, 127 vertices (one third of the considered vertices) have degrees between 387 and 391.

C. 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.

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(e)) 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 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 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 1 (Deviation from Mirror Pattern). *A vertex v's degree of deviation from* Mirror Pattern *in graph G is*

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

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 $O(n+m)$ [1].

2) Complementarity of Core-A*:* Anomaly detection in graphs (especially in social networks) has been extensively researched (see Section VI), and many of them detect dense subgraphs since anomalies tend to form dense subgraphs, as we also show in Section III-B. Especially, the latest methods [6], [7] 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 [6], which includes Fraudar [7] as a special case) are complementary, we compare their performances when different-size subgraphs are injected into social networks. 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) [11].

As seen in Figure 3, 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 $rank_c$.

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 follows:

$$a\text{-}score(G') = |E'|/|V'| + w\sum\nolimits_{v\in V'} dmp(v)/|V'| \quad (1)$$

where $w > 0$ is a parameter for balancing the two factors: $|E'|/|V'|$ and $\sum_{v\in V'} dmp(v)/|V'|$. We set w to the ratio of the maximum values of the factors in the given graph $G(V,E)$. The maximum value of $|E'|/|V'|$ is close (within a factor of 2) to $|E^*|/|V^*|$, where $G^*(V^*,E^*)$ is the densest subgraph detected by DSM; and the maximum value of $\sum_{v\in V'} dmp(v)/|V'|$ is $\max_{v\in V} dmp(v)$. We set w to their ratio, i.e., $w = (|E^*|/|V^*|)/\max_{v\in V} dmp(v)$. Once we set w, we use [6] to identify the subgraph maximizing *a-score* (Eq (1)). The vertices in the subgraph are classified as anomalies. This entire process takes $O(m \log n)$, as DSM does [6], [7].

Figure 3 illustrates the success of our proposal to combine the scores (Eq (1)): our combination successfully detects both small and large subgraphs injected into social networks, outperforming both its component methods (CORE-A and DSM), and it detects up to **3.3× smaller** subgraphs than DSM, with near-perfect accuracy.

IV. 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 a single-pass streaming algorithm for estimating degeneracy, based on the pattern.

A. 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 5, the number of triangles has a particularly strong correlation ($r = 0.94$) with degeneracy in log scale, compared with the node-count ($r = 0.75$) and the edge-count ($r = 0.83$). Moreover, the slope is 0.32, which is very close to 1/3. This leads to Observation 2.

Observation 2. (CORE-TRIANGLE PATTERN) (C-T PATTERN in short). *In real-world graphs, the triangle count and the degeneracy obey a* 3*-to-*1 *power law. That is,*

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

B. Analysis in Kronecker Model

Why do real graphs obey C-T PATTERN? Here we show that C-T PATTERN holds for the so-called 'Kronecker Model' (Definition 2), which is considered as a very realistic graph model obeying common patterns in real-world networks [8].

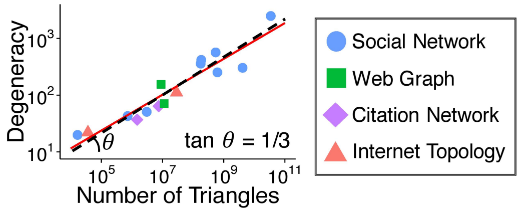


Fig. 5: **CORE-TRIANGLE PATTERN: triangle count and degeneracy obey a 3-to-1 power law.** Each point corresponds to a graph dataset in Table II. The count of triangles has a strong correlation ($r = 0.94$) with degeneracy in log scale. Moreover, the slope is very close to the theoretical slope $1/3$ (dashed line).

Definition 2 (Kronecker Graph [8]). *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, A_q is defined as:*

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

where $\otimes$ denotes Kronecker Product.

C-T PATTERN in the model is defined formally in Definition 3, where we ignore constant factors for ease of analysis.

Definition 3. (C-T PATTERN in Kronecker Model). *A Kronecker model with seed graph G_1 follows* C-T PATTERN *if* (2) *holds in $\{G_q\}_{q\geq 1}$, graphs generated by the model.*

$$k_{max} = \Theta(\#\Delta^{\frac{1}{3}}) \text{ or equivalently } \#\Delta = \Theta(k_{max}^3). \quad (2)$$

Lemmas 1 and 2 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.

Lemma 1. (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,*
1) *$k_{max}(G_q) = \Omega(max\{(d_{avg}(G_1))^q, (k_{max}(G_1))^q\})$.*
2) *$k_{max}(G_q) = O((\lambda_1(G_1))^q)$.*

Proof. See the supplementary document [12]. ■

Lemma 2. (Triangles of Kronecker Model). *The number of triangles in $\{G_q\}_{q\geq 1}$ increases exponentially with q. Let $\lambda(G_1) = (\lambda_1, ..., \lambda_n)$ be the eigenvalues of the adjacency matrix of the seed graph G_1. Then, $\#\Delta(G_q) = \Theta((\sum_{i=1}^n \lambda_i^3)^q)$.*

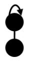
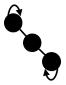
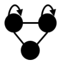
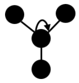
Proof. See the supplementary document [12]. ■

Based on the speed of increase in degeneracy and triangle count given in Lemmas 1 and 2, Theorem 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 (3) and $\sum_{i=1}^n \lambda_i^3 \leq \lambda_1^3$ in (4) can hold since the eigenvalues can be negative.

Theorem 1. (C-T PATTERN in Kronecker Model). *In Kronecker graphs with a seed graph G,*
1) *A sufficient condition for* C-T PATTERN *to hold is*

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

TABLE III: 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_{max}^3	1	8	8	1
d_{avg}^3	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

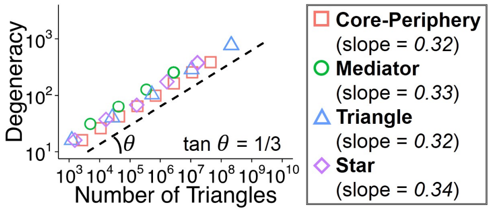


Fig. 6: **CORE-TRIANGLE PATTERN holds in Kronecker Model.** Points represent graphs generated by Kronecker Model with different seed graphs. The slopes between the triangle count and degeneracy are close to $1/3$ (dashed line) in log scale regardless of seed graphs.

2) *A seed graph satisfying the sufficient condition exists.*
3) *A necessary condition for* C-T PATTERN *to hold is*

$$\max(d_{avg}^3, k_{max}^3) \leq \sum_{i=1}^{n}\lambda_i^3 \leq \lambda_1^3. \quad (4)$$

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 1, and $\#\Delta(G_q) = \Theta(c^q)$ by Lemma 2. Therefore, $\#\Delta(G_q) = \Theta((k_{max}(G_q))^3)$, and C-T PATTERN holds. The Mediator seed graph in Table III satisfies this sufficient condition.

Assume that the necessary condition is not met. By Lemmas 1 and 2, $(k_{max}(G_q))^3$ 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. ■

Many realistic seed graphs satisfy the necessary condition for C-T PATTERN, as listed in Table III. 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 6. The slope of regression line between the number of triangles and degeneracy is very 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 (Theorem 2), another mathematically tractable graph generation model where each of possible $\binom{n}{2}$ edges occurs independently with probability p.

Theorem 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,*

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

Proof. See the supplementary document [12]. ■

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

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

C. 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 significantly better trade-off between speed and accuracy than a state-of-the-art method.

1) Algorithm: Computing degeneracy in a graph stream not fitting in memory remains as a challenge. As explained in Section II-B, a recent approximate method, LOGPASS, needs $O(\log_{\alpha/2}(n))$ passes and n memory space for given α (≥ 2). However, multiple passes of graph streams are time-consuming and not even available in many real-world settings.

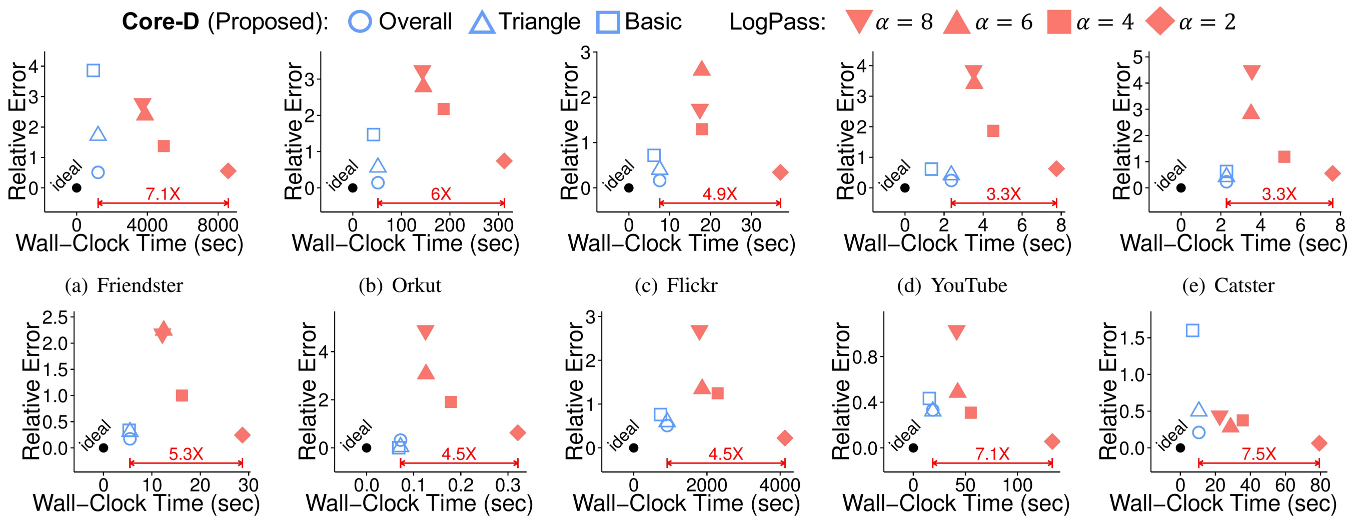
In contrast, the number of triangles can be estimated accurately even in a single pass [13], [14]. Simply sampling each edge with probability p from a graph stream and estimating the number of triangles in the whole graph from that in the sampled graph [13] also can be thought as a single-pass streaming 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 2), 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 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 IV summarizes the estimates of the coefficients obtained by linear regression on the real-world graphs in Table II. 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, 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



(a) Friendster (b) Orkut (c) Flickr (d) YouTube (e) Catster

(f) Skitter (g) Email (h) Twitter* (i) LiveJournal* (j) Patent*

* Graphs whose degeneracies are known to be affected by anomalies (see Section III-B)

Fig. 7: **CORE-D achieves both speed and accuracy.** Points in each plot represent the performances of different methods with different parameters. 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 **7× 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.

Algorithm 1: CORE-D with TRIANGLE MODEL

```
Input: Graph stream: $G$, Sampling probability : $p$
Output: Estimated degeneracy: $\hat{k}_{max}$
$G_{Sample} = \emptyset$
for each edge $e$ in $G$ do
    add $e$ to $G_{Sample}$ with probability $p$
end for
$\#\Delta_{Sample} \leftarrow$ InMemoryTriangleCounting($G_{Sample}$) [13]
$\hat{\#\Delta} \leftarrow \#\Delta_{Sample} * (1/p)^3$
$\hat{k}_{max} \leftarrow exp(w_{1,0} + w_{1,1} \log(\hat{\#\Delta}))$
return $\hat{k}_{max}$
```

on average to store sampled edges. The memory requirement becomes $O(n)$ if we set sampling probability $p = n/m$.

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 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 graph sampling (in Algorithm 1) and computing n and m can be conducted at the same time within one pass.

2) Experiments: We compare the performances of CORE-D and LOGPASS. We used a desktop with a 3.6GHz cpu and 16GB memory space, and graphs (see Table II) 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 $p = n/m$ so that on average n memory space is required in CORE-D[1], as in LOGPASS. We compared accuracy using *relative error* defined as:

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

Note that, in order to fairly evaluate accuracy in a new graph, we excluded the graph being tested from training sets when estimating coefficients of the models.

Experimental results in the largest datasets are presented in Figure 7, where CORE-D provided a significantly better trade-off between accuracy and speed than LOGPASS. Specifically, CORE-D (with OVERALL MODEL) was up to **7× 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 III-B).

Among the models of CORE-D, OVERALL MODEL consistently yielded the best performance in all the datasets except the Email dataset. 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, based on CORE-TRIANGLE PATTERN.

V. PATTERN 3: "STRUCTURED CORE PATTERN"

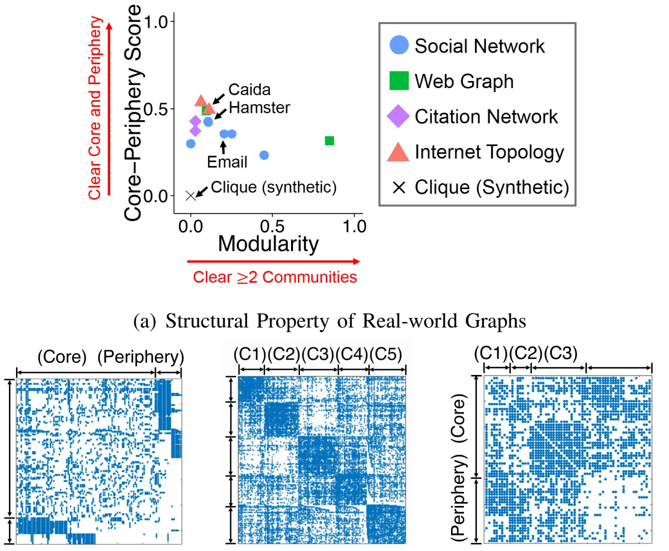
In this section, we describe STRUCTURED CORE PATTERN and discuss its application to influential spreader identification.

A. Observation: Pattern in Real-world Graphs

How do the degeneracy-cores in real-world graphs look like? Are they cliques? Our observation indicates that degeneracy-cores in real-world graphs are not cliques but have structural patterns such as core-periphery [15] (i.e., have a cohesive core and a loosely connected periphery) and communities [16] (i.e., consist of groups of vertices with dense connections internally and sparser connections between groups). This leads to Observation 3, which is supported by the following facts:

- As shown in Table II, degeneracy-cores have density much less than one in all datasets (e.g., 0.02 in Friendster

[1] Figure 1 in the supplementary documenet [12] shows that CORE-D works reliably even with smaller number of samples.



(a) Structural Property of Real-world Graphs

(b) Core-Periphery (Hamster)

(c) Communities (Email)

(d) Core-Periphery with Communities (Caida)

Fig. 8: **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 three degeneracy-cores. Ci denotes the i-th community.

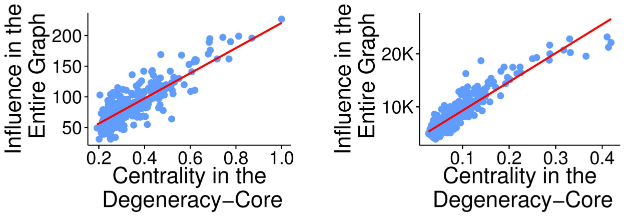
and 0.03 in Orkut) except LiveJournal and Twitter, whose degeneracy-cores include anomalies (see Section III-B).

- In all datasets, degeneracy-cores have significantly higher core-periphery score[2] (e.g., 0.54 in Skitter and 0.49 in Stanford) than cliques, as shown in Figure 8(a).
- Figure 8(a) also indicates that many datasets have significantly higher modularity[3] than cliques (e.g., 0.85 in NotreDame and 0.47 in Orkut).
- Reordering vertices in the sparsity patterns of degeneracy-cores reveals structural patterns such as core-periphery and communities. Figures 8(b)-8(d) show some examples.

Observation 3 (STRUCTURED CORE PATTERN)**.** *In real-world graphs, degeneracy cores have structural patterns such as core-periphery and communities.*

B. 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 VI for related work). For the problem of finding individual spreaders (instead of a set of spreaders, which is another well-studied problem), [10] showed 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,



(a) Email ($r = 0.84$)

(b) Orkut ($r = 0.92$)

Fig. 9: **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 correlation coefficient. Influence is measured using SIR Model simulation (see Appendix B), and in-core centrality (Definition 4) is used for centrality.

Algorithm 2: CORE-S for top-k spreaders

```
Input: Graph: $G$, Number of spreaders: $k$ ($\leq n_{max}$)
Output: $k$ 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 $V'$ using
   power iteration in $G'$
4: return top-$k$ vertices with the highest in-core centralities
```

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. 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 B for the measurement method) and in-core centrality, which we define in Definition 4, as shown in Figure 9.

Definition 4 (In-Core Centrality)**.** *Let $G'(V', E')$ be the degeneracy-core of 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 (see Section VI).

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

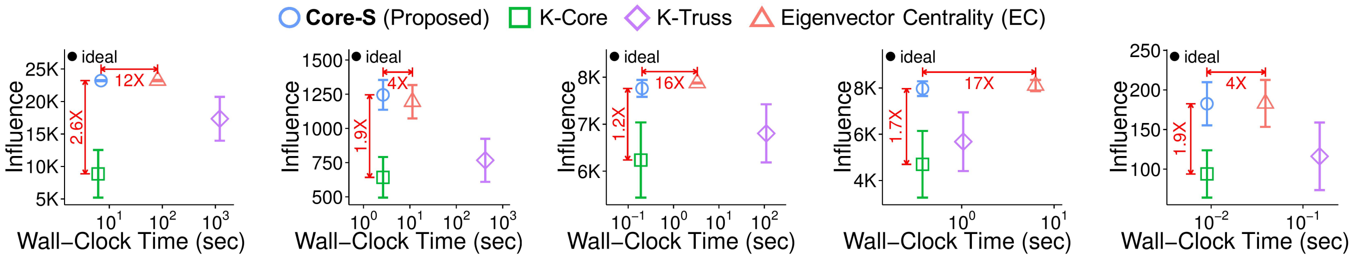
C. CORE-S*: Algorithm for Influential Spreader Identification*

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 top competitors.

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

[2] 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 [15] for details.

[3] Strength of community structure. The fraction of the edges within communities minus such fraction expected in a randomly connected graph. See [16] for details.



(a) Orkut ($\beta = 0.002$) (b) Flickr ($\beta = 0.001$) (c) Catster ($\beta = 0.002$) (d) YouTube ($\beta = 0.01$) (e) Email ($\beta = 0.01$)

Fig. 10: **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 (98-104%) influence.

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.

2) Experiments: The experimental settings were the same with those in Section IV-C2. We compared the average influence of ten vertices (see the supplementary document [12] for results with different numbers of spreaders) chosen by CORE-S with that of the vertices chosen by the following methods:

- K-CORE [10]: all vertices with the highest coreness.
- K-TRUSS [17]: all vertices with the highest truss number.
- Eigenvector Centrality (EC) [18]: top-ten vertices with the highest eigenvector centralities in the entire graph.

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

As seen in Figure 10, 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. Besides, CORE-S was up to **17× faster**, than EC, which has to compute 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 (98-104%) with that of the vertices found by EC.

VI. RELATED WORK

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

Applications of k-core Analysis. The concept of k-core has been applied to hierarchical structure analysis [2], graph visualization [3], densest subgraph detection [19] (a special case of DSM in Section III-C2), important protein identification [4], influential spreader detection [10] (K-CORE method in Section V-C2), and graph clustering [5]. Degeneracy also has been used as a graph-sparsity measure in many domains such as AI [20] and Bioinformatics [21].

Algorithms for k-core Analysis. Core decomposition can be computed in $O(n + m)$ by repeatedly removing vertices with the smallest degree [1]. [22] proposed an incremental algorithm, while [23] proposed an external memory algorithm, which requires $O(k_{max})$ scans of graphs. For degeneracy, [9] 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.

Graph-based Anomaly Detection. There have been diverse approaches (belief propagation [24], egonet features [25], spectral methods [26], etc.) for anomaly detection in graphs (see [27] for a survey). Recently, many methods focus on dense subgraphs, which anomalies tend to form [6], [7], [28], [29]. Especially, the latest methods [6], [7] are based on densest subgraphs (i.e., subgraphs with maximum average degree). We show that our CORE-A, which detects smaller dense subgraphs consisting of low-degree vertices, is complementary to these densest-subgraph based methods, and their combination 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 (see [30]) and (2) finding individual influential spreaders. For the second problem, which we focus, vertices with high coreness [10], truss number [17], and eigenvector centrality [18] 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.

VII. 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 III): 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 IV): 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 7× *faster* than a recent multi-pass method.

STRUCTURED CORE PATTERN and CORE-S (Section V): 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.

Reproducibility. Our source code and data are publicly available at **http://www.cs.cmu.edu/~kijungs/codes/kcore/**.

Acknowledgments[4]. 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. Eliassi-Rad was supported by NSF CNS-1314603 and by DTRA HDTRA1-10-1-0120.

REFERENCES

[1] V. Batagelj and M. Zaversnik, "An o(m) algorithm for cores decomposition of networks," *arXiv preprint cs/0310049*, 2003.

[2] J. I. Alvarez-Hamelin, L. Dall'Asta, A. Barrat, and A. Vespignani, "K-core decomposition of internet graphs: hierarchies, self-similarity and measurement biases," *NHM*, vol. 3, no. 2, pp. 371–393, 2008.

[3] ——, "Large scale networks fingerprinting and visualization using the k-core decomposition," in *NIPS*, 2005.

[4] S. Wuchty and E. Almaas, "Peeling the yeast protein network," *Proteomics*, vol. 5, no. 2, pp. 444–449, 2005.

[5] C. Giatsidis, F. Malliaros, D. M. Thilikos, and M. Vazirgiannis, "Corecluster: A degeneracy based graph clustering framework," in *AAAI*, 2014.

[6] K. Shin, B. Hooi, and C. Faloutsos, "M-zoom: Fast dense-block detection in tensors with quality guarantees," in *ECML/PKDD*, 2016.

[7] B. Hooi, H. A. Song, A. Beutel, N. Shah, K. Shin, and C. Faloutsos, "Fraudar: Bounding graph fraud in the face of camouflage," in *KDD*, 2016.

[8] J. Leskovec, D. Chakrabarti, J. Kleinberg, and C. Faloutsos, "Realistic, mathematically tractable graph generation and evolution, using kronecker multiplication," in *PKDD*, 2005, pp. 133–145.

[9] M. Farach-Colton and M.-T. Tsai, "Computing the degeneracy of large graphs," in *LATIN*, 2014, pp. 250–260.

[10] M. Kitsak, L. K. Gallos, S. Havlin, F. Liljeros, L. Muchnik, H. E. Stanley, and H. A. Makse, "Identification of influential spreaders in complex networks," *Nature Physics*, vol. 6, no. 11, pp. 888–893, 2010.

[11] J. Davis and M. Goadrich, "The relationship between precision-recall and roc curves," in *ICML*, 2006, pp. 233–240.

[12] "Supplementary material (proofs and additional experiments)." [Online]. Available: http://www.cs.cmu.edu/~kijungs/codes/kcore/supple.pdf

[13] C. E. Tsourakakis, U. Kang, G. L. Miller, and C. Faloutsos, "Doulion: counting triangles in massive graphs with a coin," in *KDD*, 2009.

[14] Y. Lim and U. Kang, "Mascot: Memory-efficient and accurate sampling for counting local triangles in graph streams," in *KDD*, 2015.

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

[16] M. E. Newman, "Modularity and community structure in networks," *PNAS*, vol. 103, no. 23, pp. 8577–8582, 2006.

[17] M.-E. G. Rossi, F. D. Malliaros, and M. Vazirgiannis, "Spread it good, spread it fast: Identification of influential nodes in social networks," in *World Wide Web Companion*, 2015.

[18] B. Macdonald, P. Shakarian, N. Howard, and G. Moores, "Spreaders in the network sir model: An empirical study," *arXiv preprint arXiv:1208.4269*, 2012.

[19] M. Charikar, "Greedy approximation algorithms for finding dense components in a graph," in *APPROX*, 2000.

[20] E. C. Freuder, "A sufficient condition for backtrack-free search," *JACM*, vol. 29, no. 1, pp. 24–32, 1982.

[21] G. D. Bader and C. W. Hogue, "An automated method for finding molecular complexes in large protein interaction networks," *BMC bioinformatics*, vol. 4, p. 2, 2003.

[22] A. E. Sariyüce, B. Gedik, G. Jacques-Silva, K.-L. Wu, and Ü. V. Çatalyürek, "Streaming algorithms for k-core decomposition," *PVLDB*, vol. 6, no. 6, pp. 433–444, 2013.

[23] J. Cheng, Y. Ke, S. Chu, and M. T. Özsu, "Efficient core decomposition in massive networks," in *ICDE*, 2011.

[24] S. Pandit, D. H. Chau, S. Wang, and C. Faloutsos, "Netprobe: a fast and scalable system for fraud detection in online auction networks," in *WWW*, 2007.

[25] L. Akoglu, M. McGlohon, and C. Faloutsos, "Oddball: Spotting anomalies in weighted graphs," in *PAKDD*, 2010.

[26] B. A. Prakash, A. Sridharan, M. Seshadri, S. Machiraju, and C. Faloutsos, "Eigenspokes: Surprising patterns and scalable community chipping in large graphs," in *PAKDD*, 2010.

[27] L. Akoglu, H. Tong, and D. Koutra, "Graph based anomaly detection and description: a survey," *Data Min. Knowl. Discov.*, vol. 29, no. 3, pp. 626–688, 2015.

[28] A. Beutel, W. Xu, V. Guruswami, C. Palow, and C. Faloutsos, "Copycatch: stopping group attacks by spotting lockstep behavior in social networks," in *WWW*, 2013.

[29] M. Jiang, A. Beutel, P. Cui, B. Hooi, S. Yang, and C. Faloutsos, "A general suspiciousness metric for dense blocks in multimodal data," in *ICDM*, 2015.

[30] D. Kempe, J. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network," in *KDD*, 2003.

APPENDIX A
DESCRIPTION OF REAL-WORLD GRAPH DATASETS

Social Networks. Hamster, Catster, YouTube, Flickr, Orkut, LiveJournal, and Friendster are friendship networks of users in the corresponding online communities. Twitter is a subscription network among users in a microblogging service. Email is an email network among employees of Enron Corp. and between the employees and people outside the company.

Web Graphs. NotreDame and Stanford are hyperlink networks of web pages from each university.

Internet Topologies. Caida and Skitter are internet topologies obtained from routing tables and traceroute data.

Citation Networks. Patent is a citation network among U.S. patents. HepTh is a citation network of papers submitted to the HepTh section in arXiv.

All datasets are available at **http://www.cs.cmu.edu/~kijungs/codes/kcore/**.

APPENDIX B
MEASURING INFLUENCE USING SIR MODEL SIMULATION

To evaluate influence as a spreader, we simulate spreading processes using SIR Model [10], 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 [17].

[4] 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.