Under consideration for publication in Knowledge and Information Systems

Inferring Lockstep Behavior from Connectivity Pattern in Large Graphs

Meng Jiang[1], Peng Cui[1], Alex Beutel[2], Christos Faloutsos[2] and Shiqiang Yang[1]

[1]Department of Computer Science and Technology, Tsinghua University, Beijing, China;
[2]Computer Science Department, Carnegie Mellon University, PA, USA

Abstract.

Given multimillion-node graphs such as "who-follows-whom", "patent-cites-patent", "user-likes-page" and "actor/director-makes-movie" networks, how can we find unexpected behaviors? When companies operate on the graphs with monetary incentives to sell Twitter "Followers" and Facebook page "Likes", the graphs show strange connectivity patterns. In this paper, we study a complete graph from a large Twitter-style social network, spanning up to 3.33 billion edges. We report strange deviations from typical patterns like smooth degree distributions. We find that such deviations are often due to "lockstep behavior" that large groups of followers connect to the same groups of followees. Our first contribution is that we study strange patterns on the adjacency matrix and in the spectral subspaces with respect to several flavors of lockstep. We discover that (a) the lockstep behaviors on the graph shape dense "block" in its adjacency matrix and creates "rays" in spectral subspaces, and (b) partially overlapping of the behaviors shape "staircase" in its adjacency matrix and creates "pearls" in spectral subspaces. The second contribution is that we provide a fast algorithm, using the discovery as a guide for practitioners, to detect users who offer the lockstep behaviors in undirected/directed/bipartite graphs. We carry out extensive experiments on both synthetic and real datasets, as well as public datasets from IMDb and US Patent. The results demonstrate the scalability and effectiveness of our proposed algorithm.

Keywords: Lockstep behavior; Connectivity pattern; Spectral subspace; Propagation method; Singular vector

Received Mar 22, 2014
Revised Sep 07, 2015
Accepted Sep 26, 2015

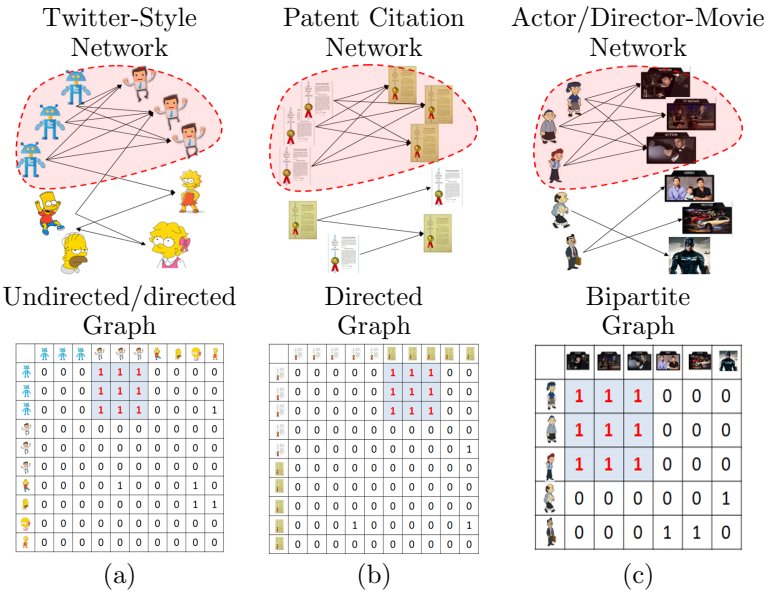


Fig. 1. The lockstep behavioral pattern is common in many graph-based applications such as social networks, patent citations and IMDb. The strange behaviors create lockstep block that is too big and too dense to occur in the undirected/directed/bipartite graphs.

1. Introduction

Given a large-scale graph from different kinds of applications such as social networks, patent citation networks and phone call networks, how can we catch strange user behaviors, and how can we find intriguing and unexpected connectivity patterns? While the strange behaviors have been documented across services ranging from telecommunication fraud (Becker et al, 2010) to deceptive Ebay's reviews (Chau et al, 2006) to ill-gotten Facebook's page-likes (Beutel et al, 2013), here we study the common strange behavior patterns and attempt developing a general, effective method to catch such behaviors.

Figure 1 shows three examples of lockstep behaviors. (a) In Facebook/Twitter-style social networks that can be represented with undirected/directed graphs, some companies set up thousands or even millions of botnets to act together, consistently connect to the same group of followees (customers) to increase their market value. Therefore, though the followees are not popular, they have a large number of followers that are probably paid to follow or created by a script. These lockstep behaviors create large, dense blocks in the adjacency matrix. (b) In patent citation networks, authors/owners who are working on the same topic/project often cite the relevant patents of their community. (c) In actor/director-movie networks (IMDb, MovieLens, Netflix, etc.), actors/actresses/directors often collaborate with their friends because close relationships bring easy communication for better understanding of the roles in movies. These networks can be represented with bipartite graphs. Note that lockstep behaviors that a group of actors and directors consistently join a group of movies create dense blocks in the adjacency matrix. The lockstep behavioral pattern is common in many graph-based applications, and hense an important and interesting question is *how to catch the near-full dense blocks, i.e., catch those lockstep behavioral links*?

In reality, the problem is not that easy. On huge social networks, there are

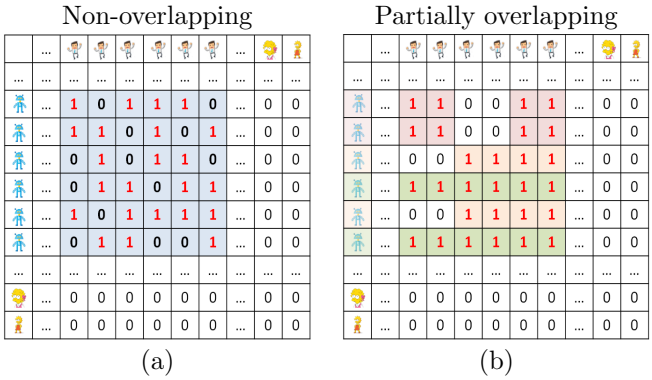


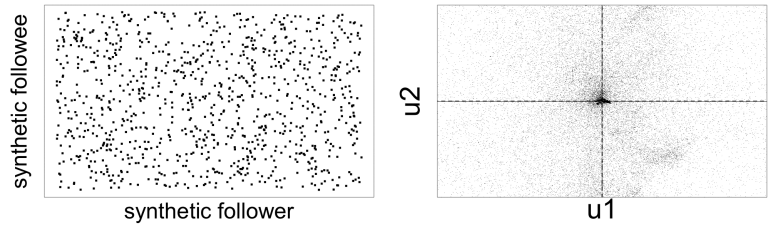
Fig. 2. How can we catch non-overlapping and partially overlapping lockstep behaviors? How can we catch dense but not 100% dense lockstep blocks in (a)? How can we catch the blocks when they have overlaps between each other? In (b), the customers get benefits (high follower number) from 3 groups of botnets but not fully connected to them.

many botnet services that build lockstep behaviors to increase their customers' follower number. Such behaviors distort the network structure and have been harmful to the ordinary users' social experience (causing distrust, frauds, spams, etc.). The botnet services have developed strategies to evade the detection. One is to make the blocks of less density. For example, in Figure 1a, a new difficult problem is *how to catch dense but not 100% dense lockstep blocks*. When is a block too large and dense enough to occur in the graph? Figure 1b presents how several groups of botnets connect to their customers. The botnet groups often share customer and thus their lockstep blocks have partially overlaps. *How can we catch the partially overlapping lockstep behaviors?*

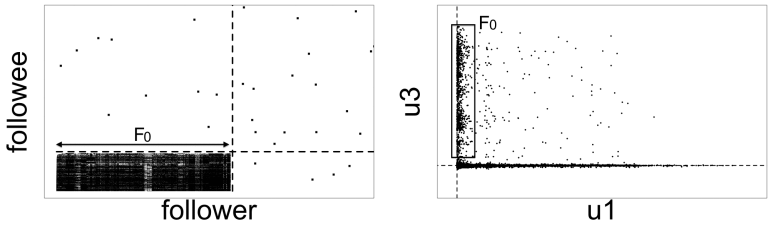
Several recent studies have used graph data to characterize connectivity patterns, with a focus on understanding the community structure (Leskovec et al, 2008; Fortunato et al, 2010; Chen et al, 2012) and the cluster property (Zha et al, 2001; Gunnemann et al, 2013). However, no analysis was presented to demonstrate what strange connectivity patterns we can infer strange behaviors from and how to catch the strange behaviors. In this paper, we work on a complete, directed graph from Tencent Weibo, one of the most popular microblogging services in China. The graph was crawled in January 2011 and it has more than *117 million* users and *3.33 billion* edges. We study users' following behaviors from connectivity patterns in Weibo graph. We discuss different flavors of "lockstep"such as *no* lockstep (Figure 3a and 3b), *non-overlapping* lockstep (Figure 3c and 3d) or *partially-overlapping* lockstep patterns (Figure 3e and 3f). We characterize connectivity patterns in the adjacency matrix of the graph, and examine the associated patterns in spectral subspaces.

Figure 3a, 3c and 3e plot the connectivities, i.e., non-zero entries in the adjacency matrix, in which a black point shows that the follower node on the X-axis connects to the followee node on the Y-axis. The dense, black "blocks" created by lockstep behaviors are highlighted at the bottom left corner by the dashed lines. Figure 3b, 3d and 3f plot every follower node by its values in a pair of left-singular vectors of the matrix. These figures visualize the spectral subspaces, called *spectral-subspace plot* and the dashed lines are X-axis and Y-axis.

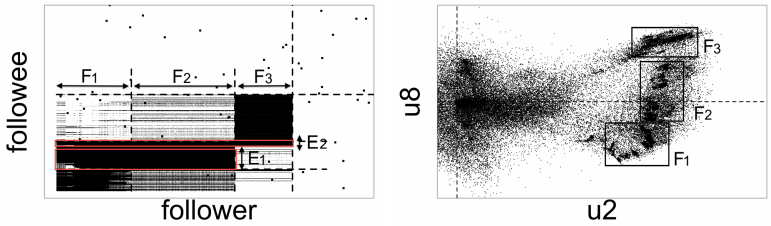
Definition 1.1 (Spectral-subspace plot). Spectral subspace is spanned by a pair of singular vectors of the adjacency matrix of a given graph. A spectral-



(a) synthetic random graph (b) around the origin in spectral subspace



(c) "block" in adjacency matrix (d) "rays" in spectral subspace



(e) "staircase" in adjacency matrix (f) "pearls" in spectral subspace

Fig. 3. Lockstep behavior shows specific connectivity patterns and strange patterns in spectral subspaces: On synthetic graph, followers are around the origin in all spectral subspaces. On Weibo, non-overlapping lockstep behaviors of followers in group F_0 shape a dense "block" in adjacency matrix and create "rays" in spectral subspace. Overlapping lockstep behaviors of followers in group F_1-F_3 create a "staircase" and "pearls".

Term	Description
"Staircase"	Multiple cases of lockstep, forming overlapping blocks in spectral subspaces
"Rays"	A set of points along a line that goes through the origin in the plots
"Pearls"	A set of points that form spherical-like clusters within roughly same radius

Table 1. Terms and their descriptions utilized in this paper.

subspace plot is a 2-dimensional plot where nodes in the graph are represented by dots in the spectral subspace.

For example, Figure 3f plots all the follower nodes in a spectral subspace spanned by the 2^{rd} and 8^{th} left-singular vectors of the adjacency matrix. Our discussion is as follows, while we use the terms in Table 1 to notate the patterns.

– *No lockstep behavior:* According to the Chung-Lu model (Chung et al, 2002), we generate a random power law graph where no lockstep behavior exists. The adjacency matrix in Figure 3a has no large, dense blocks. We study every 2-dimensional spectral subspace of this synthetic graph and observe that follower nodes are around the original point, as shown in Figure 3b.

– *Non-overlapping lockstep behavior:* On Weibo, there is a group of followers

Behavior pattern	Connectivity pattern	Spectral subspace
No lockstep behavior	Scatter	Around the origin
Non-overlapping lockstep behavior	"Blocks"	"Rays"
Partially overlapping lockstep behavior	"Staircase"	"Pearls"

Table 2. How to use spectral-subspace plots to infer strange behavior patterns: Lockstep behavior patterns show strange connectivity patterns on the graph. The spectral subspace of adjacency matrix presents strange shapes in the plots.

in F_0 connecting to the same group of followees. Thus, the adjacency matrix shows a large, dense block (83,208 followers, 81.3% dense) in Figure 3c. Figure 3d plots the spectral subspace formed by the 1^{st} and 3^{rd} left-singular vectors. The followers in group F_0 neatly align the Y-axis. We name this pattern "ray" according to the shape of the points.

– *Partially overlapping lockstep behavior:* A more surprising connectivity pattern we discover in the adjacency matrix is a "staircase" (10,052 followers, 43.1% dense), as shown in Figure 3e. Groups of followers in F_1-F_3 behave in lockstep, forming three more than 89% dense blocks. However, different from the non-overlapping case, F_1-F_2 have the same large group of followees E_1, and F_1-F_3 share a small group E_2. The overlapping lockstep behaviors of the followers create multiple micro-clusters of points that deviate from the origin and lines in the 2^{nd} and 8^{th} left-singular vector subspace. Figure 3f shows the spherical micro-clusters, roughly on a circle, so called "pearls" pattern.

A handbook (Table 2) shows how to use spectral-subspace plots to infer strange behavior patterns. The key idea is that lockstep behavior patterns show strange connectivity patterns on the graph, and the spectral subspace of adjacency matrix presents strange shapes in the plots. The later methodology section explains the handbook in details.

Inspired by the observations, we propose a novel approach LockInfer, which includes effective and efficient techniques that can learn the connectivity patterns and infer lockstep behaviors. The contributions are as follows:

– *Insights:* We offer new insights into the fingerprints on the singular vectors left by different types of synthetic lockstep behaviors. This gives us a set of rules as diagnostic tools that data scientists and practitioners can use to discover strange connectivity patterns and strange user behaviors.

– *Algorithm:* We propose a fast, scalable algorithm that exploits the insights above, and automatically find the followers that behave in lockstep. Run time of the algorithm is linear to the number of edges in the graph. We demonstrate the effectiveness on both synthetic data and real graphs including a "who-follows-whom" graph from Tencent Weibo and other two public datasets from IMDb and US Patent.

The rest of the paper is organized as follows: Section 2 discusses related work. Section 3 provides insights from strange connectivity patterns and Section 4 introduces our algorithm inferring lockstep behaviors. We give experimental results in Section 5 and conclude in Section 6. A preliminary version could be found at (Jiang et al, 2014).

	(a.1) Detecting lockstep behavior forming "blocks"	(a.2) Detecting lockstep behavior forming "staircase"	(b) Selecting seeds ("rays" and "pearls")	(c) Scalability that complexity $\leq O(E)$
METIS	×	×	×	√
CROCHET	×	×	×	√
ADJCLUSTER	√	×	×	×
SPOKEN	√	×	×	√
COPYCATCH	√	×	×	√
LOCKINFER	√	√	√	√

Table 3. Can previous approaches solve the problem: (a) detecting lockstep behavior forming (a.1) dense "blocks", (a.2) "staircase"; (b) interpreting "rays" and "pearls"; (c) being scalable for large graphs? We show advantages of our proposed method LOCKINFER.

2. Related work

There is a significant body on research related to our problem, which we categorize into four groups: subgraph mining, graph partitioning, spectral clustering and community detection. Specifically, we survey graph mining algorithms with eigenvectors and spectral subspaces, and compare with our proposed LOCKINFER from the perspectives of effectiveness, interpretation and scalability in Table 3.

Subgraph mining: A lot of work devotes to techniques for mining subgraphs from a vast body of applications including hypertext data (Chakrabarti et al, 2002), computational biology and computer networking (Aggarwal et al, 2010). The graph-mining techniques like CROCHET find quasi-clique patterns (Pei et al, 2005; Jiang et al, 2009), frequent graph (Yan et al, 2003), period subgraph (Lahiri et al, 2010), and dense subgraph (Moonesinghe, 2008; Bahmani et al, 2012; Jiang et al, 2014; Jiang et al, 2015). Non-overlapping lockstep behavior forms dense bipartite subgraph instead of frequent graph and clique, but the subgraph formed by partially overlapping lockstep behavior is not dense enough to be detected by existing techniques.

Graph partitioning: The classical graph partitioning problem is to divide a graph into components, such that the components are of about the same size and there are few connections between the components, as shown in METIS (Karypis et al, 1998; Chakrabarti, 2004; Dhillon et al, 2007; Akoglu, 2010). However, when the lockstep behavior has overlapping parts, the graphs will have "no good cuts" (Leskovec et al, 2008). Recently there are works like COPYCATCH on bipartite core detection using belief propagation (Feng et al, 2012; Beutel et al, 2013; Jiang et al, 2014; Jiang et al, 2015). But how to appropriately choose the seeds remains challenging to all the related methods, while we are giving a handbook on seed selection.

Spectral clustering: Spectral clustering methods have been widely used in large graphs (Yan et al, 2009). A spectral clustering algorithm was presented by (Ng et al, 2002) using eigenvectors of matrices derived from the data. A spectral bi-partitioning algorithm was devised by (Huang et al, 2008) using the second eigenvector of the normalized Laplacian matrix. The properties of spectral subspaces have received much attention. SPOKEN (Prakash et al, 2010) shows that the singular vectors of mobile call graphs, when plotted against each other, have separate lines along specific axes, which is associated with the presence of tightly-knit communities. However, the lines formed by nodes in well-structured communities are not necessarily axes aligned (Ying et al, 2009). ADJCLUSTER (Wu et al, 2011) give theoretical studies to explain the existence of orthogonal

lines in the spectral subspaces. However, existing works have not fully explained the (tilting) "rays" and even not discussed about "pearls" in the plots.

Community detection: A great deal of work has been devoted to detecting communities (Newman et al, 2006; Satuluri et al, 2009). The idea of (Leskovec et al, 2008) captures the intuition of a cluster as set of users with better internal connectivity than external connectivity. Researchers also infer community structure from network topology by optimizing the modularity (Clauset et al, 2004; Wakita et al, 2007). It is desirable that user of a community have a dense internal links and small number of links connected to users of other communities. However, lockstep behavior patterns do not follow such hypothesis because the users connect to different components of targets.

In summary, none of the above approaches provided a guide for practitioners to understand real settings, namely, non-overlapping and partially overlapping lockstep behaviors, with an explanation for the strange spectral patterns we observe ("staircase" and "pearls"), and strange connectivity patterns.

3. Guide for lockstep behavior inference

In this section, we first give a definition of "Lockstep Blocks" with a theoretical bound of the density. Then we introduce how to plot spectral subspaces. By discussing different types of lockstep behavior and showing the Lockstep Blocks that the behavior brings, we give a list of rules on which type of lockstep behavior the spectral patterns and connectivity patterns represent.

3.1. Definition of Lockstep Block

Let (S, T) be the subgraph formed by the set of source nodes S and the set of target nodes T. After appropriate ordering of the nodes, this is a *block* in the adjacency matrix. Let $d(S, T)$ be the density of this block, i.e., fraction of non-zero entries. "Lockstep Blocks" are informally defined as the blocks with density higher than what the uniformity assumption would expect. Formally, the definition is as follows:

Definition 3.1 (Lockstep Block). In an adjacency matrix A ($M \times N$ and density D), a $m \times n$ block (S, T) is called a *Lockstep Block*, if and only if its density $d(S, T)$ is higher than the uniformity threshold $\hat{d}$: $d(S, T) \geq \hat{d}$, where $\hat{d}$ is the threshold density, defined in Lemma 3.1.

The intuition is that large dense blocks represent lockstep behavior and thus they look strange. The threshold density $\hat{d}$ is estimated as follows:

Lemma 3.1. The threshold density $\hat{d}$ ensures that a *Lockstep Block* is highly unlikely, i.e., on the average, it appears less than once, in the sparse matrix. Thus, the threshold density is

$$\hat{d} = \frac{1}{log(D)} \left(\frac{1}{n} log \frac{m}{M} + \frac{1}{m} log \frac{n}{N} \right).$$

Proof. If A is the adjacency matrix of a $M \times N$ (density D) Erdös-Rényi graph,

$A_{i,j}$ is an independent Bernoulli random variable, each having probability D. The set of Lockstep Blocks ($m{\times}n$ and density $d \geq \hat{d}$) on A is

$$X(A,m,n,\hat{d}) = \{(S,T) : \sum_{i\in S}\sum_{j\in T} A_{i,j} \geq mn\hat{d}, S = \{i_1,...i_m\}, T = \{j_1,...j_n\},$$
$$1 \leq i_1 < \cdots < i_m \leq M, 1 \leq j_1 < \cdots < j_m \leq N\}$$

Let $Y{=}\sum_{i\in S}\sum_{j\in T} A_{i,j}$ and its expectation $\mu{=}E[Y]{=}mnD$, thus the expected count of such Lockstep Blocks is

$$E[\mid X(A,m,n,\hat{d}) \mid] = \binom{M}{m}\binom{N}{n} Pr[Y \geq mn\hat{d}].$$

According to Chernoff bound and Stirling's approximation for large factorials,

$$Pr[Y \geq mn\hat{d}] \leq \left(\left(\frac{D}{\hat{d}}\right)^{\hat{d}}\left(\frac{1-D}{1-\hat{d}}\right)^{1-\hat{d}}\right)^{mn},$$
$$\binom{M}{m}\binom{N}{n} \sim \frac{1}{2\pi\sqrt{mn}}\left(\frac{M}{m}\right)^{m}\left(\frac{N}{n}\right)^{n}.$$

Therefore, the logarithm of expected count is

$$\begin{aligned}
&logE[\mid X(A,m,n,\hat{d}) \mid] \\
&\leq -mn\left(\hat{d}\cdot log\frac{\hat{d}}{D} + (1-\hat{d})\cdot log\frac{1-\hat{d}}{1-D}\right) - m\cdot log\frac{m}{M} - n\cdot log\frac{n}{N} - log(2\pi\sqrt{mn}) \\
&\leq -mn\hat{d}\cdot log\frac{\hat{d}}{D} - m\cdot log\frac{m}{M} - n\cdot log\frac{n}{N} \\
&= -mn\hat{d}\cdot(log\hat{d} - logD) - m\cdot log\frac{m}{M} - n\cdot log\frac{n}{N} \\
&\approx mn\hat{d}\cdot logD - m\cdot log\frac{m}{M} - n\cdot log\frac{n}{N}
\end{aligned}$$

where the block's density $\hat{d}$ is often much higher than the data's density D, i.e., $\hat{d} \gg D$.

So the threshold density is

$$\hat{d} = \frac{1}{log(D)}\left(\frac{1}{n}log\frac{m}{M} + \frac{1}{m}log\frac{n}{N}\right).$$

For any block in the graph with a density higher than $\hat{d}$, the expected count is below one, which is the reason we name it as "Lockstep Block".

For example, Figure 4 plots the expected count of blocks with a given scale on a $1M{\times}1M$ ($3M$ edges) graph. A $100{\times}100$ block is a "Lockstep Block" if d is higher than $\hat{d} = 2\%$. □

3.2. Spectral-subspace plot

The concept of "spectral-subspace plot" is fundamental. The intuition behind it is that it is a visualization tool to help us see strange patterns. Let A be the

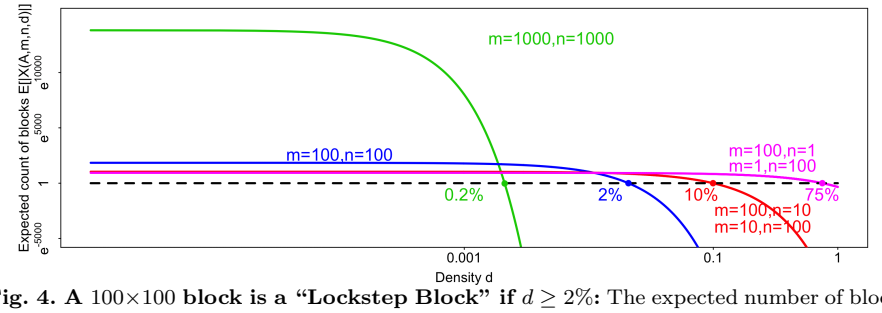


Fig. 4. A 100×100 **block is a "Lockstep Block" if** $d \geq 2\%$**:** The expected number of blocks on a $1M\times1M$ random graph decreases with the density increasing.

$N \times N$ adjacency matrix of our social graph. Each user can be envisioned as an N-dimensional point; a spectral-subspace plot is a projection of those points in N dimensions, into a suitable 2-dimensional subspace. Specifically, the subspace is spanned by two singular vectors.

More formally, the k-truncated singular value decomposition (SVD) is a factorization of the form $A = U\Sigma V^T$, where Σ is a $k \times k$ diagonal matrix with the first k singular values, and U and V are orthonormal matrices of dimensions $N \times k$. U and V contain as their columns the left- and right- singular vectors, respectively. Let $u_{n,i}$ be the (n,i) entry of matrix U, and similarly, $v_{n,i}$ is the entry of matrix V. The score $u_{n,i}$ is the coordinate of n-th follower on the i-th left-singular vector. Thus, we define (i,j)-left-spectral-subspace plot as the scatter plot of the points $(u_{n,i}, u_{n,j})$, for $n = 1, \ldots, N$. This plot is exactly the projection of all N followers on the i-th and j-th left-singular vectors. We have the symmetric definition for the N users as followees: (i,j)-right-spectral-subspace plot is the scatter plot of the points $(v_{n,i}, v_{n,j})$, for $n = 1, \ldots, N$. Clearly, it is easy to visualize such 2-dimensional plots; if used carefully, the plots can reveal a lot of information about the adjacency matrix, as we will show shortly.

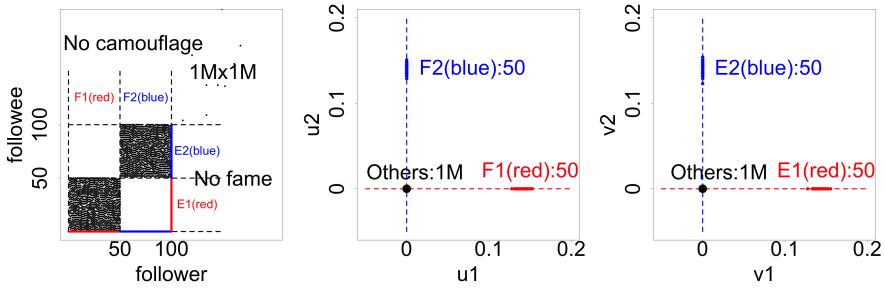
As we had shown in Figure 3(a,b), normally, given a random power law graph, we would expect to find a cloud of points around the origin in all the spectral subspaces. However, we find strange shapes ("ray" and "pearl") in some left-spectral-subspace plots of Weibo data. The question we want to answer here is: *What kind of user behavior could cause "rays" and "pearls" in spectral subspaces?*

The short answer is different types of lockstep behavior. We explain below in more detail what type of lockstep behavior generates such the odd patterns.

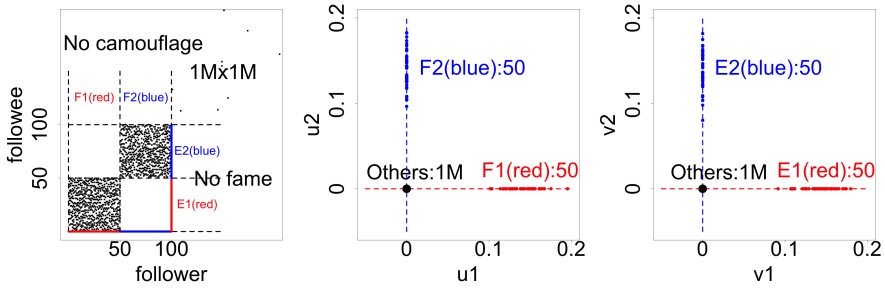
3.3. "Ray" for non-overlapping lockstep behavior

In order to enumerate all the types of lockstep behavior, we introduce concepts of "camouflage" and "fame". If a group of followers F had monetary incentives to follow the same group of followees E in lockstep, they could follow additional followees who are not in E, which is called "camouflage" that helps look normal. Similarly, the group of followees E could have additional followers who are not in F, which we succinctly call "fame".

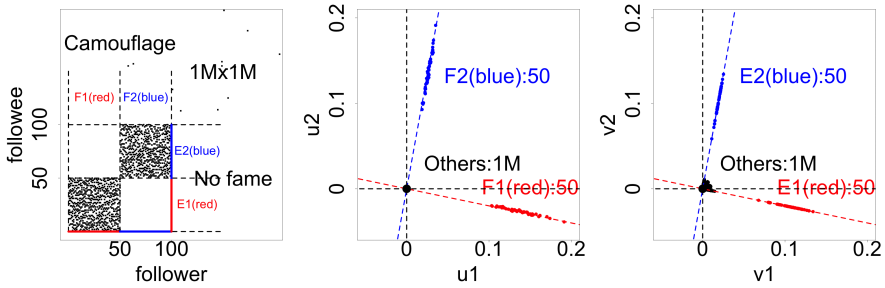
With these concepts, we can now study users' lockstep behavior with synthetic datasets. We first generate a $1M \times 1M$ random power law graph and then inject two groups of followers that separately operate in lockstep. In detail, we create 50 new followers in group F_1 to consistently follow 50 followees in group E_1. Similarly, we create another new follower group F_2 to follow a followee group



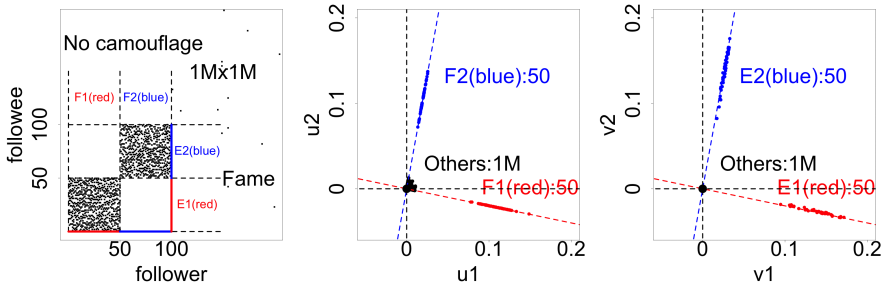
Rule 1 (short "rays"): two blocks,
high density (90%), no "camouflage", no "fame"



Rule 2 (long "rays"): two blocks,
low density (50%), no "camouflage", no "fame"



Rule 3 (tilting "rays"): two blocks,
with "camouflage", no "fame"



Rule 3 (tilting "rays"): two blocks,
no "camouflage", with "fame"

Fig. 5. Rule 1-3 ("rays"): non-overlapping blocks in adjacency matrix.

E_2. Thus, if we plot black dots for non-zero entries in the adjacency matrix in the left side of Figure 5, we spot two 50×50 non-overlapping, dense blocks. Properties of the non-overlapping lockstep behavior are discussed as follows:

- *Density:* High, if a new follower connects to 90% of the related followee group; low, if the ratio is as small as 50%.
- *Camouflage:* With camouflage, if the follower connects to 0.1% of other followees; no camouflage, if he follows only the new followees and no one else.
- *Fame:* With fame, if a new followee is also followed by 0.1% of other followers; no fame, if the followee is followed by no one else.

The spectral subspaces formed by left- and right-singular vectors are plotted in the middle and right of Figure 5, respectively. We spot footprints left in these plots by the different types of non-overlapping lockstep behavior and summarize the following rules:

- *Rule 1 (short "rays"):* If the lockstep behavior of followers is compact on the graph, the adjacency matrix contains one or more non-overlapping blocks of high density like 90%. The spectral-subspace plots show short rays: a set of points that densely fall along a line that goes through the origin.
- *Rule 2 (long "rays"):* If a group of followers and a group of followees are consistently but loosely connected, the adjacency matrix contains blocks of low density like 50%. The plots show long rays: the rays stretch into lines aligned with the axes and elongate towards the origin.
- *Rule 3 (tilting "rays"):* If the follower group has "camouflage" or the followee group has "fame", the adjacency matrix shows sparse external connections outside the blocks. Different from Rule 1-2, a more messy set of rays come out of the origin at different angles, called tilting rays.

Besides the practical proof, we use matrix decomposition theory (Kalman, 1996) to prove the "rays" shape. Suppose that in random matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$, we have two dense blocks $\mathbf{A_1} \in \mathbb{R}^{n_1 \times n_1}$ and $\mathbf{A_2} \in \mathbb{R}^{n_2 \times n_2}$. We denote the rest of the random matrix by $\mathbf{A_0} \in \mathbb{R}^{(N-n_1-n_2)\times(N-n_1-n_2)}$. Without essential difference, we focus on the left-singular vectors and denote the 1^{st} and the 2^{nd} vector by $\mathbf{u_1} = [\mathbf{u_{11}}, \mathbf{u_{12}}, \mathbf{u_{10}}]^{\mathrm{T}}$ and $\mathbf{u_2} = [\mathbf{u_{21}}, \mathbf{u_{22}}, \mathbf{u_{20}}]^{\mathrm{T}}$. If the two blocks have no overlap, we have

$$\begin{bmatrix} \mathbf{A}_1\mathbf{A}_1{}^{\mathrm{T}} & & \\ & \mathbf{A}_2\mathbf{A}_2{}^{\mathrm{T}} & \\ & & \mathbf{A}_0\mathbf{A}_0{}^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} \mathbf{u_{11}} \\ \mathbf{u_{12}} \\ \mathbf{u_{10}} \end{bmatrix} = \lambda_1 \begin{bmatrix} \mathbf{u_{11}} \\ \mathbf{u_{12}} \\ \mathbf{u_{10}} \end{bmatrix}$$

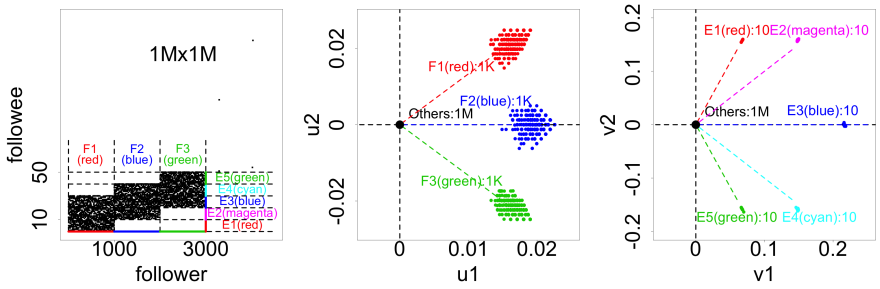
$$\begin{bmatrix} \mathbf{A}_1\mathbf{A}_1{}^{\mathrm{T}} & & \\ & \mathbf{A}_2\mathbf{A}_2{}^{\mathrm{T}} & \\ & & \mathbf{A}_0\mathbf{A}_0{}^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} \mathbf{u_{21}} \\ \mathbf{u_{22}} \\ \mathbf{u_{20}} \end{bmatrix} = \lambda_2 \begin{bmatrix} \mathbf{u_{21}} \\ \mathbf{u_{22}} \\ \mathbf{u_{20}} \end{bmatrix}$$

$$\mathbf{u_{11}}{}^{\mathrm{T}}\mathbf{u_{21}} + \mathbf{u_{12}}{}^{\mathrm{T}}\mathbf{u_{22}} + \mathbf{u_{10}}{}^{\mathrm{T}}\mathbf{u_{20}} = 0$$

$$trace(\mathbf{A_1}\mathbf{A_1}{}^{\mathrm{T}}) > trace(\mathbf{A_2}\mathbf{A_2}{}^{\mathrm{T}}) \gg trace(\mathbf{A_0}\mathbf{A_0}{}^{\mathrm{T}})$$

$$||\mathbf{u_{11}}||^2 + ||\mathbf{u_{12}}||^2 + ||\mathbf{u_{10}}||^2 = 1$$

$$||\mathbf{u_{21}}||^2 + ||\mathbf{u_{22}}||^2 + ||\mathbf{u_{20}}||^2 = 1$$



(a) adjacency matrix (b) left-spectral subspace (c) right-spectral subspace

Fig. 6. Rule 4 ("pearls"): a "staircase" of three partially overlapping blocks.

So, we assume that $\mathbf{u_{10}} = \mathbf{u_{20}} = \mathbf{0}$. Then, we have

$$\begin{cases} \mathbf{A_1}\mathbf{A_1}^{\mathrm{T}}\mathbf{u_{11}} = \lambda_1\mathbf{u_{11}} \\ \mathbf{A_2}\mathbf{A_2}^{\mathrm{T}}\mathbf{u_{12}} = \lambda_1\mathbf{u_{12}} \\ \mathbf{A_1}\mathbf{A_1}^{\mathrm{T}}\mathbf{u_{21}} = \lambda_2\mathbf{u_{21}} \\ \mathbf{A_2}\mathbf{A_2}^{\mathrm{T}}\mathbf{u_{22}} = \lambda_2\mathbf{u_{22}} \\ \mathbf{u_{11}}^{\mathrm{T}}\mathbf{u_{21}} + \mathbf{u_{12}}^{\mathrm{T}}\mathbf{u_{22}} = 0 \\ ||\mathbf{u_{11}}||^2 + ||\mathbf{u_{12}}||^2 = 1 \\ ||\mathbf{u_{21}}||^2 + ||\mathbf{u_{22}}||^2 = 1 \end{cases}$$

The answer to solving these functions is to let

$$\mathbf{u_{12}} = \mathbf{u_{21}} = \mathbf{0},$$

which indicates that the first two left-singular vectors are

$$\mathbf{u_1} = \begin{bmatrix} \mathbf{u_{11}} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}; \mathbf{u_2} = \begin{bmatrix} \mathbf{0} \\ \mathbf{u_{22}} \\ \mathbf{0} \end{bmatrix}$$

where $||\mathbf{u_{11}}||^2 = 1$ and $||\mathbf{u_{22}}||^2 = 1$.

It's easy to see, when we plot the $\mathbf{u_1}$ vs $\mathbf{u_2}$ subspace plot, we can spot the "rays" aligned with the axes. The nodes in the 1^{st} block have big values in $\mathbf{u_1}$ and zero in $\mathbf{u_2}$. The nodes in the 2^{nd} block have big values in $\mathbf{u_2}$ and zero in $\mathbf{u_1}$.

In summary, we find that non-overlapping lockstep behavior creates rays on the spectral-subspace plots: as the density decreases, the rays elongate; as the followers add camouflage or the followees add fame, the rays tilt.

3.4. "Pearl" for partially overlapping lockstep behavior

If a group of followers consistently follows their related group of followees, and partially connect to other groups of followees, we say they have partially overlapping lockstep behavior.

Here we inject the random power law graph with three follower groups F_i, for $i = 1, \ldots, 3$, and five followee groups E_i, for $i = 1, \ldots, 5$. Each follower group

has 1,000 fans and each followee group has 10 idols. Followers in F_1 connect to followees in E_1-E_3; followers in F_2 connect to followees in E_2-E_4; and followers in F_3 connect to followees in E_3-E_5; Figure 6a plots the adjacency matrix and Figure 6b plots the left- and right-spectral subspaces. We summarize a new rule.

– *Rule 4 ("pearls"):* Overlapping lockstep behavior creates "staircase" in the matrix, that is, multiple dense blocks that are overlapping due to followers from each block connecting to some followees in some other blocks. The spectral-subspace plots show "pearls" as a set of points that form spherical-like high density regions within roughly a same radius from the origin, reminiscent of pearls in a necklace.

In our case, Figure 6b shows "pearls" of three clusters, each having 1,000 followers in groups from F_1 to F_3. Figure 6c shows five clusters, each having 10 followees in E_1 to E_5. If the follower groups share some followees, or followee groups have the same followers, their clusters are close on these plots.

Let's continue denoting the matrices by what we did in the "ray" proof. Now suppose the overlapping block is $\mathbf{A_{12}} \in \mathbb{R}^{n_1 \times n_2}$, we have

$$\begin{bmatrix} \mathbf{A}_1\mathbf{A}_1{}^{\mathrm{T}} & \mathbf{A}_{12}\mathbf{A}_{12}{}^{\mathrm{T}} & \\ & \mathbf{A}_2\mathbf{A}_2{}^{\mathrm{T}} & \\ & & \mathbf{A}_0\mathbf{A}_0{}^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} \mathbf{u_{11}} \\ \mathbf{u_{12}} \\ \mathbf{u_{10}} \end{bmatrix} = \lambda_1 \begin{bmatrix} \mathbf{u_{11}} \\ \mathbf{u_{12}} \\ \mathbf{u_{10}} \end{bmatrix}$$

$$\begin{bmatrix} \mathbf{A}_1\mathbf{A}_1{}^{\mathrm{T}} & \mathbf{A}_{12}\mathbf{A}_{12}{}^{\mathrm{T}} & \\ & \mathbf{A}_2\mathbf{A}_2{}^{\mathrm{T}} & \\ & & \mathbf{A}_0\mathbf{A}_0{}^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} \mathbf{u_{21}} \\ \mathbf{u_{22}} \\ \mathbf{u_{20}} \end{bmatrix} = \lambda_2 \begin{bmatrix} \mathbf{u_{21}} \\ \mathbf{u_{22}} \\ \mathbf{u_{20}} \end{bmatrix}$$

$$\mathbf{u_{11}}{}^{\mathrm{T}}\mathbf{u_{21}} + \mathbf{u_{12}}{}^{\mathrm{T}}\mathbf{u_{22}} + \mathbf{u_{10}}{}^{\mathrm{T}}\mathbf{u_{20}} = 0$$

$$trace(\mathbf{A_1}\mathbf{A_1}{}^{\mathrm{T}}) > trace(\mathbf{A_2}\mathbf{A_2}{}^{\mathrm{T}}) > trace(\mathbf{A_{12}}\mathbf{A_{12}}{}^{\mathrm{T}}) \gg trace(\mathbf{A_0}\mathbf{A_0}{}^{\mathrm{T}})$$

$$||\mathbf{u_{11}}||^2 + ||\mathbf{u_{12}}||^2 + ||\mathbf{u_{10}}||^2 = 1$$

$$||\mathbf{u_{21}}||^2 + ||\mathbf{u_{22}}||^2 + ||\mathbf{u_{20}}||^2 = 1$$

We still assume that $\mathbf{u_{10}} = \mathbf{u_{20}} = \mathbf{0}$. Then, we have

$$\begin{cases} \mathbf{A_1}\mathbf{A_1}{}^{\mathrm{T}}\mathbf{u_{11}} + \mathbf{A_{12}}\mathbf{A_{12}}{}^{\mathrm{T}}\mathbf{u_{12}} = \lambda_1\mathbf{u_{11}} \\ \mathbf{A_2}\mathbf{A_2}{}^{\mathrm{T}}\mathbf{u_{12}} = \lambda_1\mathbf{u_{12}} \\ \mathbf{A_1}\mathbf{A_1}{}^{\mathrm{T}}\mathbf{u_{21}} + \mathbf{A_{12}}\mathbf{A_{12}}{}^{\mathrm{T}}\mathbf{u_{22}} = \lambda_2\mathbf{u_{21}} \\ \mathbf{A_2}\mathbf{A_2}{}^{\mathrm{T}}\mathbf{u_{22}} = \lambda_2\mathbf{u_{22}} \\ \mathbf{u_{11}}{}^{\mathrm{T}}\mathbf{u_{21}} + \mathbf{u_{12}}{}^{\mathrm{T}}\mathbf{u_{22}} = 0 \\ ||\mathbf{u_{11}}||^2 + ||\mathbf{u_{12}}||^2 = 1 \\ ||\mathbf{u_{21}}||^2 + ||\mathbf{u_{22}}||^2 = 1 \end{cases}$$

The first two left-singular vectors are

$$\mathbf{u_1} = \begin{bmatrix} \mathbf{u_{11}} \\ \mathbf{u_{12}} \\ \mathbf{0} \end{bmatrix}; \mathbf{u_2} = \begin{bmatrix} \mathbf{u_{21}} \\ \mathbf{u_{22}} \\ \mathbf{0} \end{bmatrix}$$

where $\mathbf{u_{12}}$ and $\mathbf{u_{22}}$ are two orthogonal, eigenvectors of the matrix $\mathbf{A_2}\mathbf{A_2}{}^{\mathrm{T}}$, and

the vectors $\mathbf{u_{11}}$ and $\mathbf{u_{21}}$ satisfy

$$\begin{cases} \mathbf{u_{11}}^{\mathrm{T}}(\mathbf{A_1}\mathbf{A_1}^{\mathrm{T}} - \lambda_1\mathbf{I})\mathbf{u_{22}} = 0 \\ \mathbf{u_{21}}^{\mathrm{T}}(\mathbf{A_1}\mathbf{A_1}^{\mathrm{T}} - \lambda_2\mathbf{I})\mathbf{u_{12}} = 0 \end{cases}$$

Thus, we have

$$(\mathbf{A_1}\mathbf{A_1}^{\mathrm{T}} - \lambda_1\mathbf{I})\mathbf{u_{11}}\mathbf{u_{21}}^{\mathrm{T}}(\mathbf{A_1}\mathbf{A_1}^{\mathrm{T}} - \lambda_2\mathbf{I}) = 0$$

If we plot the dots of vector $\mathbf{u_1}$ and $\mathbf{u_2}$, the two clusters of dots including (1) vector $\mathbf{u_{11}}$ vs $\mathbf{u_{21}}$ (2) vector $\mathbf{u_{12}}$ vs $\mathbf{u_{22}}$ are distributed in the subspace as "pearls".

Both "Ray" and "Pearl" patterns reflect dense blocks in adjacency matrix that are caused by lockstep behavioral phenomenon. However, they have intrinsic differences: "Ray" pattern catches non-overlapping dense blocks (non-overlapping lockstep behaviors), while "Pearl" pattern catches partially overlapping dense blocks (partially overlapping lockstep behaviors). Note that the overlapping part is still very dense - as dense as the major part of the dense block. It is different from "camouflage" and "fame" which are very sparse. Later in the experimental section we show that the density of "camouflage" or "fame" is as small as 1% or 0.1% of the density of blocks. Furthermore, "Pearl" means collaborations between fake accounts and "Ray" means NO collaboration. This is another big difference showing that seperating the two concepts is necessary. They indicate two differnet behavioral patterns of fake users and cover ALL the settings of lockstep behaviors.

With the insights into patterns on spectral-subspace plots (Rule 1-4), it is now easy for a practitioner to predict connectivity patterns in the adjacency matrix and infer different types of lockstep behavior.

4. Lockstep behavior inference algorithm

Our lockstep behavior inference algorithm LockInfer has two steps:

- *Seed selection:* Following Rule 1-4 in Section 3, select nodes as seeds of followers that behave in lockstep, simiply called "lockstep" followers.
- *"Lockstep" propagation:* Propagate "lockstep" score between followers and followees, and thus catch the lockstep behaviors. We scoop all the followers in lockstep, exonerate false alarms, and also spot followees in lockstep. We suggest to determine the threshold density $\hat{d}$ according to Section 3.1.

Next we describe each step in more detail. The overview of LockInfer is given below in Algorithm 1.

4.1. Seed selection

The LockInfer algorithm can start with any kind of seeds, even randomly selected ones. However, careful selection of seeds obviously accelerates the response time. We have a few heuristics to find good seeds:

- Select fans whose out-degree is at the spikes. The drawback is that the majority are actually innocent.

Input: Adjacency matrix A, the minimum size of block $m_{min} \times n_{min}$ and threshold density $\hat{d}$.
Output: Lockstep Blocks $LockB$.
$Seeds = SelectSeeds(A)$
$LockB = \{\}$
foreach *source node set* S_0 *in* $Seeds$ **do**
    `// Append a new, scooped Lockstep Block.`
    $LockB = LockB \cup Scoop(S_0, m_{min}, n_{min}, \hat{d})$
end
return $LockB$

Algorithm 1: LockInfer: infer lockstep blocks in a large graph

– Select fans with our insights in Section 3, and we demonstrate that it is more effective.

Of course, if we have side information, like IP address of the node, demographic info, etc., we can use that to select seeds (e.g., a large group of followers, all born on the first day of the same year, all males, all from the same city, would be suspicious). However, we show that, even without side information, LockInfer carefully chooses the seeds and it is still effective. Figure 7 shows how we conduct the seed selection. The "seed selection" algorithm has three steps (see Function *SelectSeeds*):

Input: The adjacency matrix A and the following values by default: the number of singular values k, number of bins K_r and K_θ for radius and angle marginals.
Output: Initial source node sets $Seeds$.
$Seeds = \{\}$
Calculate A's k-column singular vectors U and V.
foreach *eigen pair (i,j),* $1 \le i < j \le k$ **do**
    Step 1: Give spectral-subspace plot of U_i vs U_j.
    Step 2: Transfer into polar coordinates (radius r vs angle θ):
    for every user u_x, $x \le N$, $r_x = \sqrt{U_{i,x}^2 + U_{j,x}^2}$ and $\theta_x = \arctan \frac{U_{j,x}}{U_{i,x}}$.
    Step 3-1: Plot radius marginal (r vs frequency in histogram) and detect spikes with median filter:
    the frequency of r is $freq(r) = |\{x | r_x = r\}|$ and the frequency of θ is $freq(\theta) = |\{x | \theta_x = \theta\}|$.
    Step 3-2: Plot angle marginal (θ vs frequency in histogram) and detect spikes: using Median Filtering method, we can catch the red bars in Figure 7d.
    Step 3-3:Put the nodes at the spikes into $Seeds$.
end
return $Seeds$

Algorithm 2: $SelectSeeds(A)$

First, generate a range of spectral-subspace plots. We compute the top k left-singular vectors $u_1, \ldots, u_k$, and plot all the follower points in the subspace formed by each pair of the singular vectors, as shown in Figure 7a. The common spectral-

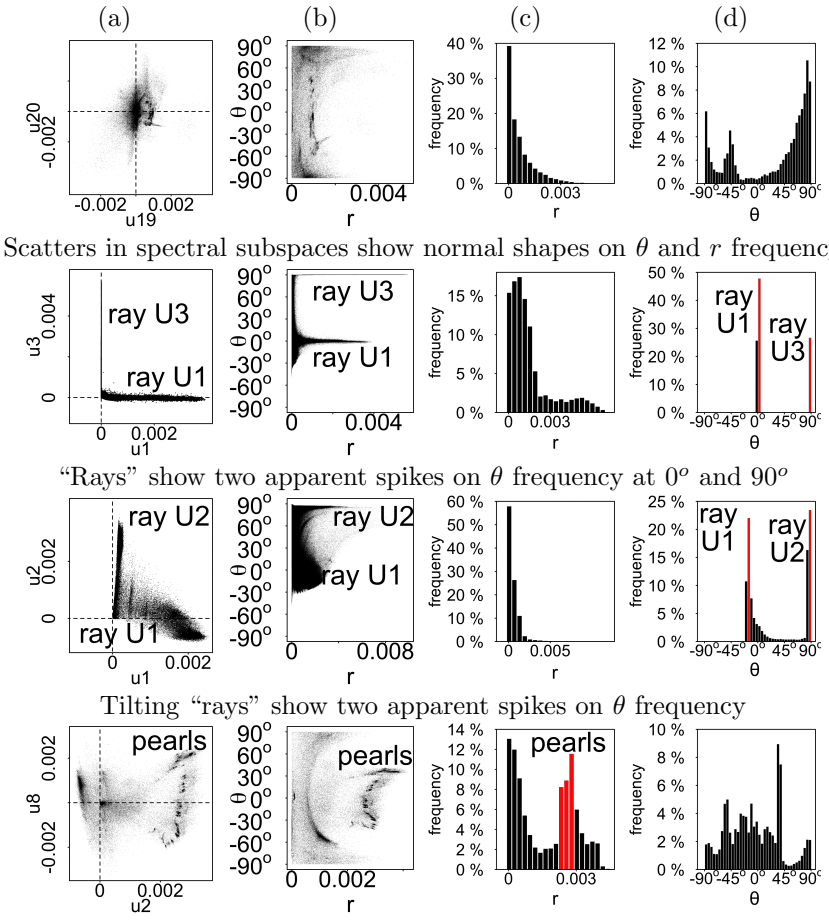


Fig. 7. Find (tilting) "rays" and "pearls": (a) spectral-subspace plot (b) hough transform (perpendicular distance r vs rotation angle θ) (c) bin plot (histogram) of distance r frequency (d) bin plot of angle θ frequency.

subspace plot has a cloud of points around the origin like what is happening for high-dimensional cases (e.g., U_{19} vs U_{20}). However, we spot strange shapes in some of the plots: "rays" with right angle (e.g., U_1 vs U_3), tilting "rays" (e.g., U_1 vs U_2) and "pearls" (e.g., U_2 vs U_8).

Second, use the points as input to Hough transform and plot them in polar coordinates (r,θ), where r is the perpendicular distance and θ is the rotation angle. As shown in Figure 7(b), for "rays", it shows two straight lines at $\theta = 0^o$ and $\theta = 90^o$; for "pearls", it shows a set of micro-clusters at some big r values.

Third, we divide radius (r) and angle (θ) axes into bins and thus plot radius and angle marginals: node frequencies of r and θ values. For "rays", the angle marginal (θ-bin plot) shows two clear spikes at 0^o and 90^o, while it shows nothing for other cases; for "pearls", the radius marginal (r-bin plot) shows a single spike, while the frequency decreases smoothly with the radius for other cases. With

median filter (Brownrigg et al, 1984), we detect the spikes and then put nodes at the spikes into seed set.

Notice that if there is no lockstep behavior, no dense block in the adjacency matrix, the spectral-subspace plots show a cloud of points around the origin, as shown in Figure 3(a-b). The node frequency of angle θ should be almost a constant, and the node frequency of distance r should decrease smoothly with the value increasing.

Parameter setting for Lockstep Block: We set the minimum size of Lockstep Block as $m_{min} \times n_{min}$ ($m_{min} = 100$, $n_{min} = 10$) and the minimum density $\hat{d}$ according to Lemma 3.1 as default parameters. Thus, the size of lockstep blocks should be at least 100×10: they have at least 100 fake accounts and 10 customers.

Here we list a few hints of the parameter setting:

- *From distributions/statistics:* Anomalies on degree distributions or statistical analysis indicate the size of a minimum Lockstep Block. For example, we suggest practitioners (e.g., employers in Twitter whose task is to detect frauds) to set $n_{min} = 20$ because of the spike on the out-degree distribution at 20 (Jiang et al, 2014).
- *From applications:* Human labors and experts have frequently worked on fraud detection in a given dataset. They may suggest the parameteres according to the smallest attack they find in the network.

From later, the experimental results, we can see the dense blocks we find are much bigger than 100×10. Our algorithm is insensitive to m_{min} and n_{min}.

4.2. "Lockstep" propagation

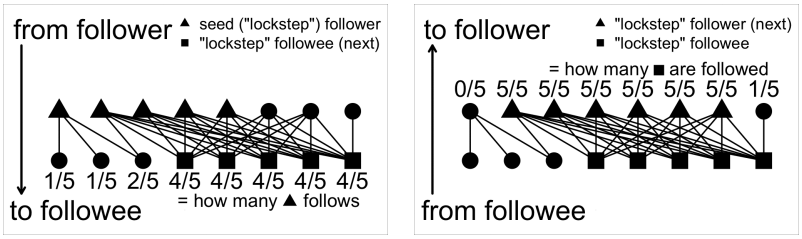
We now interpret how we start with the seeds and refine a group of followers and followees with lockstep behavior. The "lockstep" value of a followee is defined as the percentage of the seeds or "lockstep" followers who are its followers. Similarly, the "lockstep" value of a follower is defined as the percentage of the "lockstep" followees who are its followees. We need a threshold to decide which users are new "lockstep" followers/followees.

The *Scoop* algorithm recursively propagates this value from followers to followees, and vice versa, like what Belief Propagation method does. In more detail, we explain the steps as follows.

- *From follower to followee:* Figure 8a shows an example of a directed graph with followers at the top and followees at the bottom. We start with 5 "lockstep" followers as seeds for propagation. For each followee, we count how many its followers are in the seed set. We select the group of "lockstep" followees who have too many "lockstep" followers. Function $S2T$ shows how we propagate "lockstep" score from source nodes to target nodes.
- *From followee to follower:* Next for each follower, we count how many its followees are "lockstep". Figure 8b shows how we select new "lockstep" followers and exonerate those innocent with zero or one "lockstep" followee. Function $T2S$ shows how we propagate "lockstep" score from target nodes to source nodes.
- *Repeat until convergence:* Report the groups of "lockstep" followers and followees if they are not empty.

Input: S_0 is the seed. The Lockstep Block must be larger than $m \times n$ and at least d dense.
Output: Lockstep Block (S, T)
// Assert $\mid S_0 \mid \geq m$.
$T_0 = \{\}; i = 0;$
repeat
  $T_i = S2T(S_i, d);$
  if $\mid T_i \mid < n$ **then**
    return ({},{})
  end
  $S_{i+1} = T2S(T_i, d);$
  if $\mid S_{i+1} \mid < m$ **then**
    return ({},{})
  end
  $i = i + 1;$
until $S_i == S_{i-1}$;
return (S_i, T_{i-1})

Algorithm 3: $Scoop(S_0, m, n, d)$



(a) $S2T$: select "lockstep" followees: from (seed) followers to followees

(b) $T2S$: select "lockstep" followers: from followees to followers

Fig. 8. Iteratively propagate "lockstep" score between source nodes and target nodes: Find lockstep behavior of users by propagating "lockstep" value: select followers (followees) who have too many "lockstep" followees (followers).

Input: Source node set S and density d.
Output: Target node set T that has more than d of S.
// Assume adjacency matrix A is given.
return $T = \{j : \sum_{i \in S} A_{i,j} > d \mid S \mid\}$

Algorithm 4: $S2T(S, d)$

Input: Target node set T and density d.
Output: Source node set S that follows more than d of T.
// Assume adjacency matrix A is given.
return $S = \{i : \sum_{j \in T} A_{i,j} > d \mid T \mid\}$

Algorithm 5: $T2S(T, d)$

The derivation of $\hat{d}$ (Lemma 3.1) provides the lower limit of density d, and in "lockstep" propagation, we use $\hat{d}$ to select lockstep followers/followees (i.e., find lockstep blocks). Therefore, the lockstep blocks that LockInfer detects are too dense to appear, because the expected count of such big, dense blocks is smaller than 1. Overall, the derivation ensures the effectiveness of LockInfer.

4.3. Scalability

LockInfer has two computational stages: (1) handling singular value decomposition on large-scale sparse matrix; (2) read the pair-wise plots, collect "lockstep" seeds and propagate "lockstep" scores on the graph. For the first, HEigen (Kang, 2014), an accurate and efficient algorithm that implemented k-SVD on the highly scalable MapReduce environment, has proved that the time complexity is $\mathcal{O}(k(N+E))$ where N is the number of nodes and E is the number of edges. For the second stage, the time complexity is $\mathcal{O}(k^2N + kN + TE)$ where T is the number of iterations. As $k = 20$ and T is much smaller than N, and N is smaller than E, we know that the total time complexity of LockInfer is $\mathcal{O}(E)$, i.e., linear in the number of edges.

5. Experimental results

In this section we present our empirical evaluation, first on a large, real-world graph, and then on synthetic graphs where the ground truth is known.

5.1. On "who-follows-whom" Weibo social graph

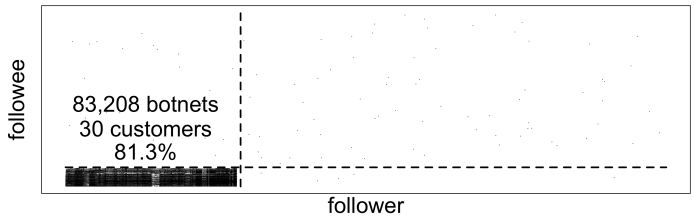
We operate our algorithm on the 100-million-node social graph Weibo. In applications it's quite unusual for the full SVD, including a full unitary decomposition, so, we tackled this problem with Truncated SVD, which can be much quicker and more economical than the compact SVD. In our WEIBO case with millions of nodes, we set the rank $k = 20$. The reason is, in Figure 7a, U_{19} vs U_{20}, we observe scatters (like a cloud of points) around the original point, indicating that the subspace in such high dimensions is random distributed. We suggest the practitioners to choose k according to the subspace plots. Since $k \ll N$ (N is the number of nodes), the algorithm is much faster than full SVD. Table 4 report the statistics of strange connectivity patterns that we find on the network.

- *"Blocks" and "staircase":* With the proposed rules and algorithm, we catch a dense block with the "ray" pattern and a staircase of three overlapping blocks with the "pearl" pattern on spectral-subspace plots. Figure 9 have show the adjacency matrix and their sets of followers F_0 and F_1-F_3.
- *High density, small "camouflage" and small "fame":* The density of every block is greater than 80%, while the density of the "staircase" is only 43%. It proves that the staircase consists of partially overlapping blocks. The camouflage, that is the connectivity between "lockstep" followers and other followees, is as small as 0.2% dense. The fame is smaller than 2%.

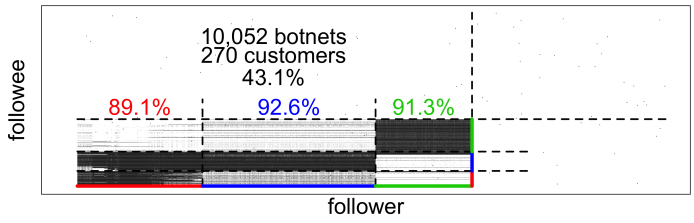
The above numbers validate the existence of non-overlapping and partially

	"ray" F_0	"pearl" F_1	"pearl" F_2	"pearl" F_3	"pearl" Total
Num. seeds	100	1,239	107	990	—
Size of block	$83,208 \times 30$	$3,188 \times 135$	$7,210 \times 79$	$2,457 \times 148$	$10,052 \times 270$
Density	81.3%	91.3%	92.6%	89.1%	43.1%
Camouflage	0.14%	0.06%	0.10%	0.05%	0.07%
Fame	0.05%	1.93%	1.94%	1.72%	1.73%
Out-degree	231±109	310±7	312±7	304±5	310±7
In-degree	2.0±1.4	9±6	10±6	17±13	12±9

Table 4. Statistics of connectivity patterns formed by groups of users with lockstep behavior: The density of the "block" and blocks in "staircase" is greater than 80%, while the WEIBO followers have little "camouflage" and followees have little "fame".



(a) Lockstep Block we find



(b) Lockstep Blocks in "staircase" pattern

Fig. 9. Blocks that LOCKINFER discovers: (a) 83K fans follows the same 30 idols in lockstep, forming a single Suspicious Block in adjacency matrix; (b) 10K fans follows 270 idols, forming a "staircase" of three overlapping blocks.

overlapping lockstep behavior and also the effectiveness of our method. Further, we give additional evidence of the similar personalities of the "lockstep" followers.

– *Strange profiles:* The login-names of 10,787 accounts from the "lockstep" users are like "a#####" (# is a digital number, for example, "a27217"). Their self-declared dates of birth are in lockstep the January 1^{st}. They were probably created by a script, as opposed to natural users. The details of some examples are listed in Table 5.
– *Small in-degree values of followers:* The average in-degree value of followers in the single "block" is as small as 2.0, while that of followers in the "staircase" is smaller than 20. The "lockstep" followers actively connect to their followees but they have little reputation themselves.
– *Similar out-degree values of followers:* The out-degree values of "lockstep" followers in the "staircase" are similarly around 300. In Figure 10, we plot the out-degree distribution of the graph in log-log scale and spot two spikes, which means abnormally high frequency of nodes who have around 300 followees.

Login-name	Date of birth	In-degree	Out-degree
a15681	1986:01:01	1	301
a21154	1975:06:04	2	301
a27217	1982:01:01	3	304
a27290	1980:01:01	5	107
a38887	1982:01:01	3	310
catty	1972:01:01	2	316

Table 5. Strange profiles we find: additional account information, of some suspicious followers, detected with LockInfer; logins, (self-declared) birthdays, and in-degrees, are all strange such as "a#####"-like nickname, January-1st birth date, and small in-degree.

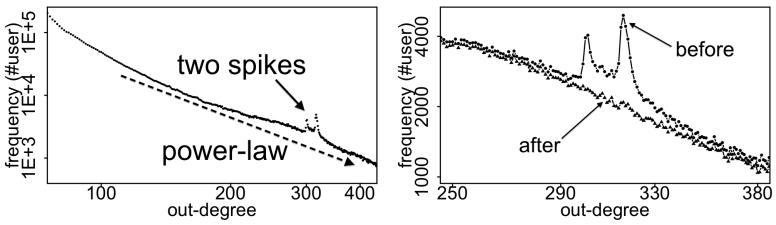
Out-degree	#User	Out-degree	#User	Out-degree	#User	Deviation
270	3,436	280	3,048	290	2,773	
300	3,944	**301**	**4,043**	302	3,418	**Spike**
311	2,852	312	2,679	323	2,836	
315	4,373	**316**	**4,918**	317	4,414	**Spike**
320	2,821	330	1,976	340	1,650	

Table 6. Strange deviations of out-degree distribution: the distribution has two spikes at out-degree values 301 and 316. We aim at spotting the deviations, explaining them with connectivity patterns, and restored the normal distribution.

After we remove the "lockstep" followers, we find out that the spikes disappear and the distribution becomes smoother.

For the last point, we want to say, most graphs exhibit smooth degree distributions, often obeying a heavy-tailed distribution (power law (Faloutsos et al, 1999; Broder et al, 2000), lognormal, etc). Deviations from smoothness are strange: Border *et al.* (Broder et al, 2000) said that in the case of the web graph, the spikes were due to link farms. Thus, if the removal of some "lockstep" users makes the degree plots smoother, then we have one more reason to believe that indeed those users were strange.

We report surprising deviations from typical patterns like power-law-like degree distributions in Table 6 and Figure 10a. The out-degree distribution in log-log scale has two spikes at out-degree 301 and 316. One direct explanation is the strange behaviors of nodes (followers) that connect to the same number of other nodes (followees). Figure 10b shows that the out-degree distribution is



(a) two spikes on distribution (b) smoother after our operation

Fig. 10. Restored typical pattern of out-degree distribution: the log-log distribution has spikes and becomes smoother after we remove followers with "lockstep behavior". The followers have similar out-degree values, i.e., similar numbers of followees from the same group.

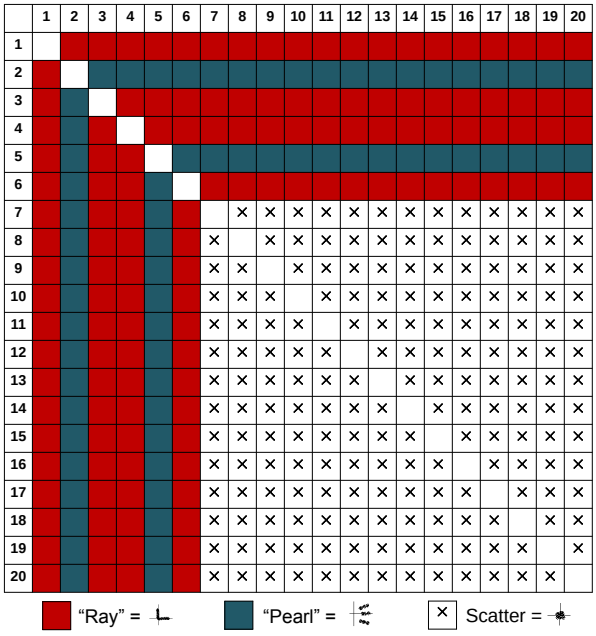


Fig. 11. LockInfer is insensitive to parameter k. In fact, we can choose $k = 6$ to catch all the "Ray" and "Pearl" patterns. Our setting $k = 20$ is sufficient.

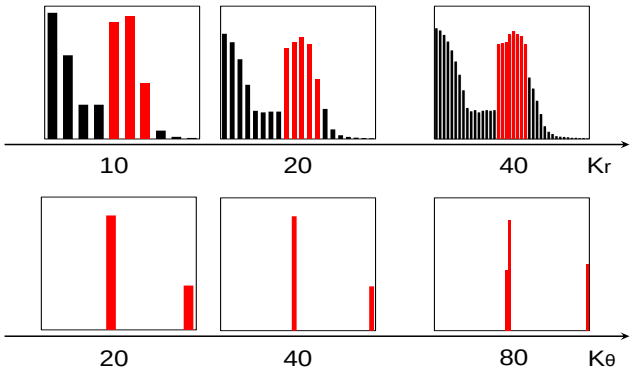


Fig. 12. LockInfer is insensitive to parameter K_r and K_θ. Our algorithm can consistently detect the spikes, varying the values of these parameters.

restored to the power-law-like, typical pattern, after we remove the connections from the follower groups who offer lockstep behaviors. The lockstep followers are exactly the nodes who create the two spikes. They have similar out-degree values, or in other words, they connect to the similar groups of followees.

Insensitivity to parameters: We set $k = 20$ for the trade-off between the number of spectral-subspace plots and the time cost of SVD. When reading the polar coordinates and creating the radius/angle marginals, we divide the radius axis into $K_r = 20$ bins as default, and angle axis into $K_\theta = 2K_r = 40$ bins because θ can be either positive or negative.

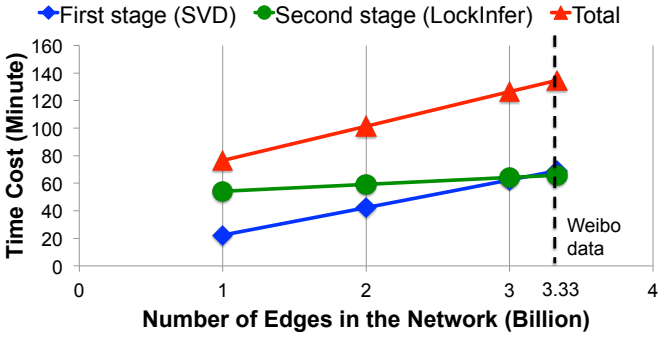


Fig. 13. Our LockInfer is scalable: The time complexity is linear in the number of edges.

	"Ray" I	"Ray" II	"Ray" III	"Ray" IV
Seed	74	57	30	103
Block	414×30	178×44	139×15	373×12
Density	61.4%	58.2%	66.7%	57.4%
Camouflage	0.65%	1.74%	1.87%	0.69%
Fame	6.06%	6.59%	12.0%	12.8%

Table 7. Lockstep Blocks on IMDb: we operate LockInfer on the dataset and find dense blocks with the four "rays".

In fact, our LockInfer algorithm is insensitive to the parameters k, K_r and K_θ. All the parameters help find red parts in Figure 7(c) and 7(d), indicating spikes on the binplots of radius/angle marginals. We operate on this Tencent Weibo real data and change the values of the parameters to check the robustness.

- For parameter k, Figure 11 shows that even though when k is as small as 6, LockInfer catches all the "Ray" and "Pearl" patterns. Our algorithm can automatically tell if there is some anomalous pattern in a spectral-subspace plot, with which we set the value of k. A bigger k will better ensure the robustness in practice.
- For K_r and K_θ, Figure 12 shows that when we change the values of K_r and K_θ, the spikes are easy to be caught: When $K_r \in \{10, 20, 40\}$ and $K_\theta \in \{20, 40, 80\}$, our algorithm can find the same red parts on the binplots.

Scalability: Figure 13 plots the wall-clock time, including the time cost of the first stage (SVD computation), the second tage (LockInfer), and the total procedure, versus the size of the graph (number of edges in billion-level), while we randomly sample the edges of Tencent Weibo social graph with a setting of the number of edges: 1, 2, and 3 billion. LockInfer costs 134.73 minutes (2h15min) to process the Weibo data. The time complexity of our algorithm is linear to the scale of the social graph and thus scalable to be applied in real applications.

5.2. On "actor-movie" IMDb dataset

We conduct experiment on IMDb dataset, i.e., a large bipartite graph between actors and movies/television series. In Figure 14, we spot "rays" on spectral-

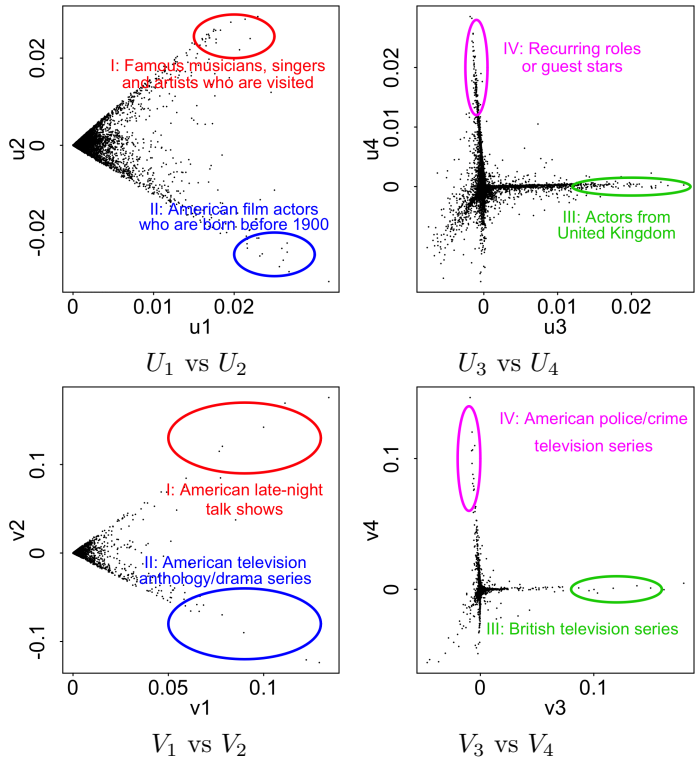


Fig. 14. "Rays" on spectral subspaces of IMDb: we spot tilting "rays", which show main connectivity patterns between actors and movies.

subspace plots of both U_1 vs U_2 and U_3 vs U_4. Table 7 shows surprising connectivity patterns on IMDb. Starting from the seeds from "rays", we scoop for dense blocks. The information of them are as follows.

- Block I consists of American late-night talk shows. The actors are musicians, singers and artists who are guests on the shows.
- Block II has many famous film actors who were born before 1900. Their television anthology/drama series were produced in around 1950s.
- Block III has British actors/actresses and their performances/shows on *BBC*.
- Block IV has recurring roles or guest stars who were invited to play in American police/crime series, like *CSI*, *NYPD Blue* and *Cold Case*.

From Table 7, we observe that

- Lockstep blocks of large scale: From the top 4 blocks' statistics we can see that the number of actors/actresses/directors ranges from 100 to 500, while the number of movies ranges from 10 to 50. So, the actor communities often have population as large as 500 but not in thousands.
- Big density: The density values range from 50% to 70%, indicating that more than half of the actors in the community join all the movies, and more than half of the movies are made by almost all the actors in the community.
- Small "camouflage": The camouflage values are often very small, because ac-

		Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
LockInfer	Precision	**0.86**	**0.83**	**0.92**	0.89	**0.99**
	Recall	**0.82**	**0.76**	**0.82**	**0.75**	**0.70**
Autopart	Precision	0.66	0.67	0.52	**0.91**	0.94
	Recall	0.52	0.46	0.46	0.21	0.49
Outrank	Precision	0.52	0.23	0.43	0.59	0.69
	Recall	0.42	0.56	0.32	0.69	0.70
Oddball	Precision	0.26	0.43	0.32	0.49	0.29
	Recall	0.32	0.26	0.65	0.29	0.64
		Cluster 6	Cluster 7	Cluster 8	Cluster 9	Cluster 10
LockInfer	Precision	**0.93**	**0.86**	**0.82**	**0.99**	**0.77**
	Recall	0.28	**0.76**	0.74	**0.72**	**0.85**
Autopart	Precision	0.56	0.33	0.29	0.76	0.61
	Recall	**0.89**	0.31	0.18	0.43	0.61
Outrank	Precision	0.36	0.63	0.72	0.79	0.59
	Recall	0.82	0.36	**0.85**	0.69	0.40
Oddball	Precision	0.22	0.63	0.44	0.56	0.70
	Recall	0.21	0.56	0.34	0.43	0.65

Table 8. Compare performances of our LockInfer and the state-of-the-art methods on US patent data: For most of the document clusters, LockInfer consistently outperforms the previous algorithms.

tors in those communties are ordinary. The real, famous movie stars do not make only one style of movies but many different kinds of films.

– Big "fame": The actors join some other movies outside the block to make more money, which can explain the ∼10% "fame" values.

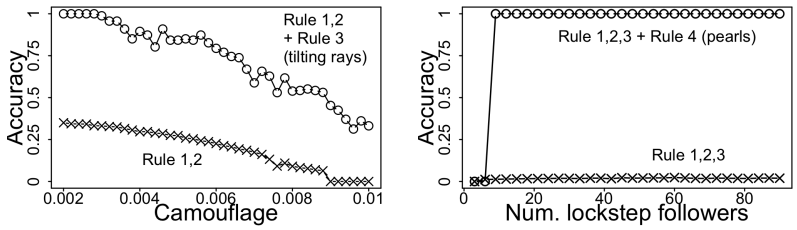
5.3. On US patent citation network

In this section, we test the effectiveness of our LockInfer algorithm on public US patent data in (Bronwyn, 2001). The dataset has 3,774,768 vertices (patents) and 16,518,947 edges (citations). Trappey *et al.* (Trappey, 2001) have carefully extracted key phrases and using fancy document clustering methods to assign the patents into 10 clusters. We use their results as the ground truth and run our algorithm on the public dataset. We evaluate the performance by popular metrics such as *precision* and *recall*. Table 8 shows the performance of our LockInfer algorithm (when $k = 20$), comparing with the state-of-the-art algorithm Autopart (Chakrabarti, 2004), Outrank (Moonesinghe, 2008) and Oddball (Akoglu, 2010).

We can observe that our LockInfer algorithm consistently outperforms the previous methods. The traditional works can infer strange substructure in large graphs including dense cliques, bipartite cores and stars. Comparing the experimental resuts, we can see that

– Autopart can automatically partition the graphs and catch the outliers. The patents in cluster 4 are in an isolated component of the graph, so, Autopart can give very high precision. However, its recall value is not good, because it catches the outliers but not the major part of this component.

– Outrank can propagate value of the vertices through the network. We initial the algorithm with PageRank value and for cluster 8, it reaches the best performance, because the patents in cluster 8 form a very dense clique. Our LockInfer can have comparatively high precision (0.82) and recall (0.74).



(a) Rule 3 (tilting "rays") helps when lockstepers have "camouflage"

(b) Rule 4 ("pearls") helps when lockstepers form "staircase"

Fig. 15. Effectiveness of Rule 3-4: if the accuracy is higher, the performance is better.

– Oddball extracts the features to assign the vertices (patents) into clusters. However, the in-degree, out-degree and some other features cannot reflect how strongly connected (cited with each other) the citations are.
– Our LockInfer algorithm often has very high (0.8∼1.0) precision and recall values. We demonstrate that only using the citation structure, we can reach almost the same performance as careful document clustering methods with key phrases.

5.4. On synthetic data

Here we want to validate the effectiveness of Rule 3 (tilting "rays") and 4 ("pearls"). We inject a group of followers and followees operating in lockstep on a 1-million-node random power law graph. The goal is to predict who are the injected nodes. We adopt *Accuracy* to qualify the performance, which is the ratio of correct predictions.

First, we add camouflage to the followers, i.e., we increase the density of connections between the followers and other followees on the graph from 0 to 0.01. We compare the performance of different versions of our algorithm: one considers Rule 3 when it selects seeds from spectral-subspace plots, and the other does not. Rule 3 says when the followers have camouflage, the rays tilt. Figure 15a shows that both accuracy values decrease with the camouflage increasing, and the algorithm that considers Rule 3 performs much better.

Second, we inject partially overlapping lockstep behavior. In other words, we put a "staircase" in the adjacency matrix. We change the size of the staircase, i.e., the number of followers. One of the algorithms compared here considers Rule 4 and the other does not. Rule 4 says when there is a staircase, some spectral-subspace plots have "pearls". Figure 15(b) shows that our algorithm that fully considers all the rules is sensitive to the number of "lockstep" followers. When it is bigger than 7, which is big enough for the behaviors to show footprints in the spectral subspaces, we can catch over 95% of the followers, while the version that does not consider Rule 4 fails to predict them.

Can the state-of-the-art methods catch the blocks with camouflage? We perform the Autopart, Outrank and Oddball on the synthetic dataset. The precision and recall of the three previous methods and our two versions of LockInfer (with and without Rule 3) have been listed in the Table 9.

We can observe that when the block has camouflage, only our method that

Camouflage	0.005		0.010	
	Precision	Recall	Precision	Recall
LockInfer with Rule 3	**0.80**	**0.78**	**0.64**	**0.47**
LockInfer without Rule 3	0.34	0.32	0.10	0.08
Autopart	0.45	0.33	0.21	0.20
Outrank	0.20	0.25	0.01	0.02
Oddball	0.02	0.02	0.03	0.04

Table 9. Our LockInfer wins when there is "camouflage" in the synthetic data: The "camouflage" cannot evade our LockInfer detection with Rule 3 ("tilting rays").

Staircase size (#user)	50		100	
	Precision	Recall	Precision	Recall
LockInfer with Rule 4	**0.99**	**1.00**	**1.00**	**1.00**
LockInfer without Rule 4	0.00	0.00	0.02	0.03
Autopart	0.15	0.16	0.14	0.20
Outrank	0.24	0.19	0.32	0.32
Oddball	0.07	0.05	0.22	0.14

Table 10. Our LockInfer wins when there is "staircase" in the synthetic data: The "staircase" cannot evade our LockInfer detection with Rule 4 ("pearls").

uses Rule 3 can catch it. All the previous methods we have tested and the version of LockInfer that does not use Rule 3 fail—small precision and recall.

Does LockInfer outperform the other methods on the "staircase"? We run similar experiments on the synthetic data with a dense "staircase". Table 10 shows the performances of our methods and the other baseline algorithms.

We can see that LockInfer significantly outperforms the existing methods, especially LockInfer without the Rule 4. Our LockInfer has almost perfect detection performance (99%-100%).

6. Conclusion

In this paper, we proposed a novel method to infer lockstep behaviors from connectivity patterns on large graphs like "who-follows-whom" social graph. We offer new understanding into the plots of spectral subspaces. "Ray" and "pearl" patterns are created by different types of lockstep behaviors (non-overlapping and partially-overlapping patterns). Using the insights, we derive a fast algorithm to detect such behavior patterns. We demonstrate the effectiveness of our method on both real graphs and synthetic data with injected lockstep behaviors.

References

Becker, Richard A., Chris Volinsky, and Allan R. Wilks. "Fraud detection in telecommunications: History and lessons learned." Technometrics 52, no. 1 (2010).

Chau, Duen Horng, Shashank Pandit, and Christos Faloutsos. "Detecting fraudulent personalities in networks of online auctioneers." In Knowledge Discovery in Databases: PKDD 2006, pp. 103-114. Springer Berlin Heidelberg, 2006.

Beutel, Alex, Wanhong Xu, Venkatesan Guruswami, Christopher Palow, and Christos Faloutsos. "CopyCatch: stopping group attacks by spotting lockstep behavior in social networks." In Proceedings of the 22nd international conference on World Wide Web, pp. 119-130. International World Wide Web Conferences Steering Committee, 2013.

Leskovec, Jure, Kevin J. Lang, Anirban Dasgupta, and Michael W. Mahoney. "Statistical properties of community structure in large social and information networks." In Proceedings of the 17th international conference on World Wide Web, pp. 695-704. ACM, 2008.

Fortunato, Santo. "Community detection in graphs." Physics Reports 486, no. 3 (2010): 75-174.

Chen, Jie, and Yousef Saad. "Dense subgraph extraction with application to community detection." Knowledge and Data Engineering, IEEE Transactions on 24, no. 7 (2012): 1216-1230.

Zha, Hongyuan, Xiaofeng He, Chris Ding, Horst Simon, and Ming Gu. "Bipartite graph partitioning and data clustering." In Proceedings of the tenth international conference on Information and knowledge management, pp. 25-32. ACM, 2001.

Gnnemann, Stephan, Brigitte Boden, Ines Frber, and Thomas Seidl. "Efficient Mining of Combined Subspace and Subgraph Clusters in Graphs with Feature Vectors." In Advances in Knowledge Discovery and Data Mining, pp. 261-275. Springer Berlin Heidelberg, 2013.

Broder, Andrei, Ravi Kumar, Farzin Maghoul, Prabhakar Raghavan, Sridhar Rajagopalan, Raymie Stata, Andrew Tomkins, and Janet Wiener. "Graph structure in the web." Computer networks 33, no. 1 (2000): 309-320.

Faloutsos, Michalis, Petros Faloutsos, and Christos Faloutsos. "On power-law relationships of the internet topology." In ACM SIGCOMM Computer Communication Review, vol. 29, no. 4, pp. 251-262. ACM, 1999.

Chung, Fan, and Linyuan Lu. "The average distances in random graphs with given expected degrees." Proceedings of the National Academy of Sciences 99, no. 25 (2002): 15879-15882.

Jiang, Meng, Peng Cui, Alex Beutel, Christos Faloutsos, and Shiqiang Yang. "Inferring strange behavior from connectivity pattern in social networks." In Advances in Knowledge Discovery and Data Mining, pp. 126-138. Springer International Publishing, 2014.

Chakrabarti, Soumen. "Mining the Web: Discovering Knowledge from Hypertext Data." (2002).

Aggarwal, Charu C. and Wang, Haixun. "Managing and Mining Graph Data." (2010).

Yan, Xifeng, and Jiawei Han. "CloseGraph: mining closed frequent graph patterns." In Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 286-295. ACM, 2003.

Lahiri, Mayank, and Tanya Y. Berger-Wolf. "Periodic subgraph mining in dynamic networks." Knowledge and information systems 24, no. 3 (2010): 467-497.

Bahmani, Bahman, Ravi Kumar, and Sergei Vassilvitskii. "Densest subgraph in streaming and mapreduce." Proceedings of the VLDB Endowment 5, no. 5 (2012): 454-465.

Jiang, Meng, Peng Cui, Alex Beutel, Christos Faloutsos, and Shiqiang Yang. "CatchSync: catching synchronized behavior in large directed graphs." In Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 941-950. ACM, 2014.

Jiang, Meng, Peng Cui, Alex Beutel, Christos Faloutsos, and Shiqiang Yang. "Catching Synchronized Behaviors in Large Networks: A Graph Mining Approach." ACM Transactions on Knowledge Discovery from Data (TKDD), 2015.

Pei, Jian, Daxin Jiang, and Aidong Zhang. "On mining cross-graph quasi-cliques." In Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining, pp. 228-238. ACM, 2005.

Jiang, Daxin, and Jian Pei. "Mining frequent cross-graph quasi-cliques." ACM Transactions on Knowledge Discovery from Data (TKDD) 2, no. 4 (2009): 16.

Karypis, George, and Vipin Kumar. "A fast and high quality multilevel scheme for partitioning irregular graphs." SIAM Journal on scientific Computing 20, no. 1 (1998): 359-392.

Dhillon, Inderjit S., Yuqiang Guan, and Brian Kulis. "Weighted graph cuts without eigenvectors a multilevel approach." IEEE Trans. Pattern Anal. Mach. Intell. 29, no. 11 (2007): 1944-1957.

Leskovec, Jure, Kevin J. Lang, Anirban Dasgupta, and Michael W. Mahoney. "Statistical properties of community structure in large social and information networks." In Proceedings of the 17th international conference on World Wide Web, pp. 695-704. ACM, 2008.

Feng, Jing, Xiao He, Bettina Konte, Christian Bhm, and Claudia Plant. "Summarization-based mining bipartite graphs." In Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 1249-1257. ACM, 2012.

Jiang, Meng, Peng Cui, Alex Beutel, Christos Faloutsos, and Shiqiang Yang. "Detecting suspicious following behavior in multimillion-node social networks." In Proceedings of the companion publication of the 23rd international conference on World wide web companion, pp. 305-306. International World Wide Web Conferences Steering Committee, 2014.

Jiang, Meng, Bryan Hooi, Alex Beutel, Shiqiang Yang, Peng Cui, and Christos Faloutsos. "A

General Suspiciousness Metric for Dense Blocks in Multimodal Data." In Proceedings of IEEE International Conference on Data Mining. IEEE, 2015.

Ng, Andrew Y., Michael I. Jordan, and Yair Weiss. "On Spectral Clustering Analysis and an algorithm." Proceedings of Advances in Neural Information Processing Systems. Cambridge, MA: MIT Press 14 (2001): 849-856.

Huang, Ling, Donghui Yan, Michael I. Jordan, and Nina Taft. "Spectral Clustering with Perturbed Data." In NIPS, vol. 21. 2008.

Yan, Donghui, Ling Huang, and Michael I. Jordan. "Fast approximate spectral clustering." In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 907-916. ACM, 2009.

Prakash, B. Aditya, Ashwin Sridharan, Mukund Seshadri, Sridhar Machiraju, and Christos Faloutsos. "Eigenspokes: Surprising patterns and scalable community chipping in large graphs." In Advances in Knowledge Discovery and Data Mining, pp. 435-448. Springer Berlin Heidelberg, 2010.

Ying, Xiaowei, and Xintao Wu. "On Randomness Measures for Social Networks." In SDM, vol. 9, pp. 709-720. 2009.

Wu, Leting, Xiaowei Ying, Xintao Wu, and Zhi-Hua Zhou. "Line orthogonality in adjacency eigenspace with application to community partition." In Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume Three, pp. 2349-2354. AAAI Press, 2011.

Wauthier, Fabian L., Nebojsa Jojic, and Michael I. Jordan. "Active spectral clustering via iterative uncertainty reduction." In Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 1339-1347. ACM, 2012.

Clauset, Aaron, Mark EJ Newman, and Cristopher Moore. "Finding community structure in very large networks." Physical review E 70, no. 6 (2004): 066111.

Newman, Mark EJ. "Finding community structure in networks using the eigenvectors of matrices." Physical review E 74, no. 3 (2006): 036104.

Wakita, Ken, and Toshiyuki Tsurumi. "Finding community structure in mega-scale social networks:[extended abstract]." In Proceedings of the 16th international conference on World Wide Web, pp. 1275-1276. ACM, 2007.

Satuluri, Venu, and Srinivasan Parthasarathy. "Scalable graph clustering using stochastic flows: applications to community discovery." In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 737-746. ACM, 2009.

Kalman, Dan. "A Singularly Valuable Decomposition: The SVD of a Matrix." The College Mathematics Journal 27, 2-23, 1996.

Brownrigg, D. R. K. "The weighted median filter." Communications of the ACM 27, no. 8 (1984): 807-818.

Chakrabarti, Deepayan. "Autopart: Parameter-free graph partitioning and outlier detection." PKDD, pp. 112–124, 2004.

Moonesinghe, HDK and Tan, Pang-Ning. "Outrank: a graph-based outlier detection framework using random walk." International Journal on Artificial Intelligence Tools, vol. 17, no. 1, pp. 19–36, 2008.

Akoglu, Leman and McGlohon, Mary and Faloutsos, Christos. "Oddball: Spotting anomalies in weighted graphs." AKDDM, vol. 17, no. 1, pp. 410–421, 2010.

Kang, U., Brendan Meeder, Evangelos E. Papalexakis, and Christos Faloutsos. "Heigen: Spectral analysis for billion-scale graphs." Knowledge and Data Engineering, IEEE Transactions on 26, no. 2 (2014): 350-362.

Bronwyn H. Hall, Adam B. Jaffe, and Manuel Trajtenberg. "The NBER patent citations data file: Lessons, insights and methodological tools." In NBER Working Papers 8498, National Bureau of Economic Research, Inc, 2001.

Trappey, CharlesV. and Trappey, AmyJ.C. and Wu, Chun-Yi. "Clustering patents using non-exhaustive overlaps." In Journal of Systems Science and Systems Engineering, vol. 19, no. 2, pp. 162–181, 2001.

Author Biographies

Meng Jiang received the B.E. degree from the Department of Computer Science and Technology of Tsinghua University, Beijing, in 2010. He is pursuing the Ph.D. degree and his main research interests include data mining and social network analysis. He was sponsored by China Scholarship Council to visit Carnegie Mellon University, US, from Aug. 2012 to May 2013. He has published over 10 papers on user behavior prediction and social recommendation in top conferences of the relevant field.

Peng Cui received the Ph.D. degree in computer science in 2010 from Tsinghua University and he is an Assistant Professor at Tsinghua. He has vast research interests in data mining, multimedia processing, and social network analysis. Until now, he has published more than 30 papers in conferences such as SIGIR, AAAI, etc. and journals such as IEEE TMM, DMKD, etc. Now his research is sponsored by National Science Foundation of China, Samsung, Tencent, etc. He also serves as Guest Editor, Co-Chair, PC member, and Reviewer of several high-level international conferences, and journals.

Alex Beutel received the computer science and physics diploma from Duke University, Durham, North Carolina, USA. He is working toward the PhD degree in the School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA. He won the Facebook Graduate Fellowship for 2014-2015. He researches large scale network behavior analysis including group behavior analysis. He has published papers in KDD 2012, WWW 2013, WWW 2014, SDM 2014 and many other international conferences.

Christos Faloutsos is a professor at Carnegie Mellon University. He has received the Presidential Young Investigator Award by the US National Science Foundation in 1989, the Research Contributions Award at ICDM 2006, the SIGKDD Innovations Award in 2010, 19 "best paper" awards (including two "test of time" awards), and four teaching awards. He is an ACM Fellow and has published more than 200 refereed articles, 11 book chapters, and one monograph. He holds six patents and he has given more than 30 tutorials and more than 10 invited distinguished lectures.

Shiqiang Yang received the B.E. and M.E. degrees from the Department of Computer Science and Technology of Tsinghua University in 1977 and 1983, respectively. He is now a Professor at Tsinghua University. His research interests include multimedia technology and systems, video compression and streaming, content-based retrieval for multimedia information. multimedia content security, and digital right management. He has published more than 100 papers and MPEG standard proposals.

Correspondence and offprint requests to: Meng Jiang, Department of Computer Science and Technology, Tsinghua University, Beijing, China. Email: jm06@mails.tsinghua.edu.cn