SliceNDice: Mining Suspicious Multi-attribute Entity Groups with Multi-view Graphs

Anonymous Author(s)

ABSTRACT

Given the reach of web platforms, bad actors have considerable incentives to manipulate and defraud users at the expense of platform integrity. This has spurred research in numerous suspicious behavior detection tasks, including detection of sybil accounts, false information, and payment scams/fraud. In this paper, we draw the insight that many such initiatives can be tackled in a common framework by posing a detection task which seeks to find groups of entities which share too many properties with one another across multiple attributes (sybil accounts created at the same time and location, propaganda spreaders broadcasting articles with the same rhetoric and with similar reshares, etc.) Our work makes four core contributions: Firstly, we posit a novel *formulation* of this task as a multi-view graph mining problem, in which distinct views reflect distinct attribute similarities across entities, and contextual similarity and attribute importance are respected. Secondly, we propose a novel *suspiciousness metric* for scoring entity groups given the abnormality of their synchronicity across multiple views, which obeys intuitive desiderata that existing metrics do not. Finally, we propose the SliceNDice *algorithm* which enables efficient extraction of highly suspicious entity groups, and demonstrate its *practicality* in terms of strong detection performance and discoveries on Snapchat's large advertiser ecosystem (89% precision and numerous discoveries of real fraud rings), marked outperformance of baselines (over 0.95 precision/recall in simulated settings) and linear scalability.

CCS CONCEPTS

• **Information systems → Web log analysis**; • **Security and privacy → Social network security and privacy**; *Web application security*;

KEYWORDS

anomalous behavior, attributed data, multi-view graphs, outlier

ACM Reference format:
Anonymous Author(s). 2019. SliceNDice: Mining Suspicious Multi-attribute Entity Groups with Multi-view Graphs. In *Proceedings of ACM KDD Conference, Anchorage, Alaska, USA, August 2019 (KDD'19),* 11 pages.
https://doi.org/10.475/123_4

1 INTRODUCTION

Online services and social networks (Snapchat, Quora, Amazon, etc.) have become common means of engagement in human activities including socialization, information sharing and commerce. However, given their dissemination power and reach, they have been long-exploited by bad actors looking to manipulate user perception, spread misinformation, and falsely promote bad content. Such malicious activities are motivated by numerous factors including monetary incentives [40] to personal [39] and political interests [3]. Tackling online misbehavior is a challenging research problem, given its variance with respect to manifestation across platforms, incentive structures and time. Despite these challenges, researchers have made progress in a variety of scenarios including fake account creation [6, 41], bot follower detection [18, 34], malware detection [9] and more. However, many of these solutions require large, labeled datasets, and offer little to no extensibility. Unfortunately, given the ever-increasing multitude of new abuse vectors, application-specific anti-abuse solutions and rich labeled sets are not always feasible or timely, motivating research towards flexible, unsupervised methods.

In this work, we propose an unsupervised solution for a wide-class of misbehavior problems in which we are tasked with discovering group-level suspicious behavior across attributed data. Our work is inspired by prior works, which have demonstrated that such behaviors often occur in lockstep and are better discernible on a group, rather than individual level [21, 37]. However, we tackle a problem which is left unaddressed by prior work:

Problem 1 (Informal). *How can we quantify and automatically uncover highly-suspicious entity groups with multiple associated attribute types and values?*

This setting naturally extends to a number of abuse detection scenarios, such as discovering online e-commerce scammers given profile creation, posting and e-mail address attributes, or pinpointing fraudulent advertisers given ad URLs, targeting criteria and key phrases. Our work leverages the insight that groups of entities who share too many, and too unlikely, attribute values are unusual and worth investigation. We build on this intuition by (a) casting the problem of mining suspicious entity groups over multiple attributes as mining cohesive subgraphs across multiple views, (b) proposing a novel metric to quantify group suspiciousness in multi-view settings, (c) developing a scalable algorithm to mine such groups quickly, and (d) demonstrating effectiveness. Specifically, our contributions are as follows.

Formulation. *We propose modeling multi-attribute entity data as a multi-view graph, which captures notions of similarity via shared attribute values between entities.* In our model, each node represents an entity (e.g. account, organization), each view represents an attribute (e.g. files uploaded, IP addresses used, etc.) and each non-zero edge indicates some attribute value overlap (see Figure 1a)

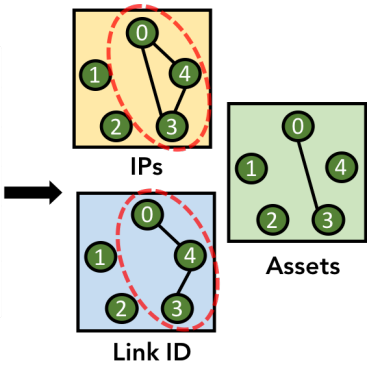
Suspiciousness Metric. *We design a novel suspiciousness metric, which quantifies subgraph (entity group) suspiciousness in multi-view graphs (multi-attribute settings).* We identify desiderata that suspiciousness metrics in this setting should obey, and prove that our metric adheres to these properties while previously proposed options do not.

Algorithm. *We propose SliceNDice, an algorithm which scalably extracts highly suspicious subgraphs in multi-view graphs.* Our algorithm uses a greedy, locally optimal search strategy to expand

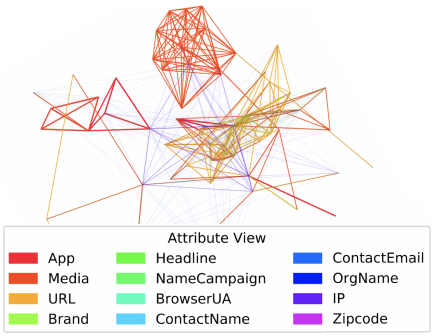
KDD'19, August 2019, Anchorage, Alaska, USA

ACM ISBN 123-4567-24-567/08/06…$15.00
https://doi.org/10.475/123_4

ID	IPs	Link ID	Assets
0	{1.2.3.4}	{b1c45}	{Cheap-Iphone.jpg}
1	{103.71.11.5}	{ytnw71}	{Smoothie.jpg}
2	{201.27.18.6}	{1m572d}	{main.jpg}
3	{112.11,16.1, 1.2.3.4}	{a6wu7}	{Promotion-1.jpg, Cheap-Iphone.jpg}
4	{1.2.3.4}	{a6wu7, b1c45}	{Cheap-Rolex.jpg}



(a) Our formulation: multi-attributed entity data as a multi-view graph



(b) SliceNDice detected e-commerce fraud

Figure 1: Our work tackles suspicious behavior detection in multi-attribute entity data using a multi-view graph mining formulation, shown in (a). Our SliceNDice algorithm offers scalable ranking and discovery of such behaviors across multiple entities and graph views, enabling us to uncover integrity violations like e-commerce fraud on the Snapchat ads platform, shown in (b).

seeds of similar nodes into larger subgraphs with more cohesion. We discuss design decisions which improve performance including careful seeding, context-aware similarity weighting and performance optimizations. Our practical implementation leverages all these, enabling efficient entity group mining.

Practicality. *We evaluate SliceNDice both on real-world data and simulated settings, demonstrating high detection performance and efficiency.* We show effectiveness in detecting suspicious behaviors on the Snapchat advertiser platform, with 12 distinct attributes and over 230,000 entities (advertiser organizations). Figure 1b shows a large network of e-commerce fraudsters uncovered by our approach (nodes are advertiser organizations, and edge colors indicate shared attributes, ranked in suspiciousness). We also conduct synthetic experiments to evaluate empirical performance against baselines, demonstrating significant improvements in multi-attribute suspicious entity group detection. Overall, our methods and results offer significant promise in bettering web platform integrity.

2 RELATED WORK

We discuss prior work in two related contexts below.

Mining entity groups. Prior works have shown that suspicious behaviors often manifest in synchronous group-level behaviors [21, 33, 37]. Several works assume inputs in the form of a single graph snapshot. [20, 32] mine communities using eigenplots from singular value decomposition (SVD) over adjacency matrices. [5, 8, 16] propose greedy pruning/expansion algorithms for identifying dense subgraphs. [7, 11] co-cluster nodes based on information theoretic measures relating to minimum-description length. Unlike these, our work entails mining groups based on overlap across multiple attributes and graph views. Several works consider multiple graph views, in a tensor formulation. [25, 30] use PARAFAC tensor decomposition to find dense communities and network attacks. [4, 17, 38] use greedy expansion methods for finding entity groups, while [31] uses greedy pruning to mine cross-graph quasi-cliques. [27, 36] use single graph clustering algorithms, followed by stitching afterwards. Our work is different in that we (a) cast multi-attribute group detection into a multi-view graph formulation, (b) simultaneously mine nodes and views to maximize a novel metric (c) use a compressed data representation, unlike other methods which incur massive space complexity from dense matrix/tensor representation.

Quantifying suspiciousness. Most approaches quantify behavioral suspiciousness as likelihood under a given model. Given labeled data, supervised learners have shown use in suspiciousness ranking tasks for video views [10], account names [12], registrations [41] and URL spamicity [24]. However, given considerations of label sparsity, many works posit unsupervised and semi-supervised graph-based techniques. [15, 35] propose information theoretic and Bayesian scoring functions for node suspiciousness in edge-attributed networks. [2, 34] use reconstruction error from low-rank models to quantify suspicious connectivity. [22] ranks node suspiciousness based on participation in highly-dense communities, [13, 14, 29] exploit propagation-based based methods with few known labels to measure account sybil likelihood, review authenticity and article misinformation propensity. Our work (a) focuses on groups rather than entities, and (b) needs no labels. Several methods exist for group subgraph/tensor scoring; [23] overviews. The most common are density [23], average degree [8, 16], subgraph weight, singular value [32]. [17] defines suspiciousness using log-likelihood of subtensor mass assuming assuming a Poisson model. Our work differs in that we (a) quantify suspiciousness in multi-view graphs, and (b) show that alternative metrics are unsuitable in this setting.

3 PROBLEM FORMULATION

The problem setting we consider (Problem 1) is commonly encountered in many real anti-abuse scenarios: a practitioner is given data spanning a large number of entities (i.e. users, organizations, objects) with a number of associated attribute types such as webpage URL, observed IP address, creation date, and associated (possibly multiple) values such as *xxx.com*, *5.10.15.20*, *5.10.15.25*, *01/01/19* and is tasked with finding suspicious behaviors such as fake sybil accounts or engagement boosters.

In tackling this problem, one must make several considerations: *What qualifies as suspicious? How can we quantify suspiciousness? How can we discover such behavior automatically?* We build from the intuition that suspicious behaviors often occur in lockstep across *multiple* entities (see Section 2), and are most discernible when considering relationships *between* entities. For example, it may be challenging without context to determine suspiciousness of an advertiser linking to URL *xxx.com*, logging in from IPs *5.10.15.20* and with creation date *01/01/19*; however, knowing that 99 other

Symbol	Definition
K	Number of total attributes (graph views)
$\mathcal{A}_i$	Set of possible values for attribute i
$\mathcal{A}_i(\cdot)$	Maps nodes to attr. i valueset: $\mathcal{A}_i : \mathcal{G} \rightarrow 2^{\mathcal{A}_i}$
$\mathcal{G}$	Multi-view graph over all views
$\vec{K}$	Indicator for multi-view graph views, $\vec{K} \in \{0,1\}^K$
$\mathcal{G}_i$	Single (i^{th}) view of multi-view graph $\mathcal{G}$
$\mathcal{G}_{\vec{K}}$	$\mathcal{G}$ over chosen $\sum\vec{K}$ views, $\mathcal{G}_{\vec{K}} = \{\mathcal{G}_i \mid \mathbb{1}(K_i)\}$
N	Number of nodes in $\mathcal{G}$
V	Volume of graph $\mathcal{G}_i$: $N(N-1)/2$
C_i	Mass of graph $\mathcal{G}_i$: $\sum_{(a,b)\in\mathcal{G}^2} w_i^{(a,b)}$
P_i	Density of graph $\mathcal{G}_i$: C_i/V
$w_i^{(a,b)}$	Edge weight between nodes a, b in $\mathcal{G}_i$, s.t. $w_i^{(a,b)} \in \mathbb{R}^+$
$\mathcal{X}$	Multi-view subgraph over all views
$\vec{k}$	Indicator for multi-view subgraph views, $\vec{k} \in \{0,1\}^K$
$\mathcal{X}_i$	i^{th} view of multi-view subgraph $\mathcal{X}$
$\mathcal{X}_{\vec{k}}$	$\mathcal{X}$ over chosen $\sum\vec{k}$ views, $\mathcal{X}_{\vec{k}} = \{X_i \mid \mathbb{1}(k_i)\}$
n	Number of nodes in $\mathcal{X}$
v	Volume of subgraph of $\mathcal{X}_i$: $n(n-1)/2$
c_i	Mass of subgraph $\mathcal{X}_i$: $\sum_{(a,b)\in\mathcal{X}^2} w_i^{(a,b)}$
ρ_i	Density of subgraph $\mathcal{X}_i$: c_i/v
z	Constraint on chosen subgraph views, $\lvert\vec{k}\rvert = z$
$f(\cdot)$	Mass-parameterized suspiciousness metric
$\hat{f}(\cdot)$	Density-parameterized suspiciousness metric

Table 1: Frequently used symbols and definitions.

advertisers share these exact properties, our perception of suspiciousness increases drastically. Thus, we focus on mining suspicious entity groups, where suspiciousness is governed by the degree of synchronicity between entities, and across various attributes.

We draw inspiration from graph mining literature; graphs offers significant promise in characterizing and analyzing between-entity similarities, and are natural data structures for the same. Graphs model individual entities as *nodes* and relationships between them as *edges*; for example, a graph could be used to describe *who-purchased-what* relationships between users and products on Amazon. Below, we discuss our approach for tackling Problem 1, by leveraging the concept of *multi-view* graphs. We motivate and discuss three associated building blocks: (a) using multi-view graphs to model our multi-attributed entity setting, (b) quantifying group suspiciousness, and (c) mining highly suspicious groups in such multi-view graphs. Throughout this paper, we utilize formal notation for reader clarity and convenience wherever possible; Table 1 summarizes the frequently used symbols and definitions, partitioned into attribute-related, graph-related and subgraph-related notation.

3.1 Representing Multi-attribute Data

In this work, we represent multi-attribute entity data as a *multi-view graph* (MVG). An MVG is a type of graph in which we have multiple *views* of interactions, usually in the form of distinct edge types; to extend our previous example, we could consider who-purchased-what, who-rates-what and who-viewed-what relationships as different views between users and products. Each view of an MVG can individually be considered as a single facet or mode of behavior, and spans over the same, fixed set of nodes. In our particular setting, we consider a representation of multi-attribute data in which we have a fixed set of nodes (entities), and their relationships across multiple views (attributes). Thus, edges in each graph view represent relationships between entities, and are weighted based on their attribute similarities in that attribute space.

Formally, we have a set of N entities with K associated attribute types over the attribute value spaces $\mathcal{A}_1 \ldots \mathcal{A}_K$. For notational convenience, we introduce attribute-value accessor mapping functions $\mathcal{A}_1 \ldots \mathcal{A}_K$ for the K attribute spaces respectively, such that $\mathcal{A}_i : \mathcal{G} \rightarrow 2^{\mathcal{A}_i}$. Effectively, $\mathcal{A}_i(a)$ denotes the subset of attribute values from $\mathcal{A}$ associated with node $a \in \mathcal{G}$. We can construe this as an MVG $\mathcal{G}$ on N nodes (entities) and K views (attributes), such that $\mathcal{G}$ is a set of individual graph views $\{\mathcal{G}_1 \ldots \mathcal{G}_K\}$. For convenience, we introduce notations $\mathcal{G}_i$ and $\mathcal{G}_{\vec{K}}$, to refer to a specific graph view, and a specific subset of graph views ($\vec{K}$ is a K-length vector indicator) respectively. We consider an edge $a \leftrightarrow b$ with $w_i^{(a,b)} > 0$ in view $\mathcal{G}_i$ to exist between nodes a,b if $\mathcal{A}_i(a) \cap \mathcal{A}_i(b) \neq \varnothing$, or informally a and b share at least one common feature value on the i^{th} attribute. If $\mathcal{A}_i(a) \cap \mathcal{A}_i(b) = \varnothing$ (no overlap between feature values), we consider that no edge between a,b exists in $\mathcal{G}_i$, or equivalently that $w_i^{(a,b)} = 0$. In general, we consider non-negative weights, so that $w_i^{(a,b)} \in \mathbb{R}^+$. We can consider many weighting strategies, but posit the notion that large weight between $\mathcal{A}_i(a)$ and $\mathcal{A}_i(b)$ indicates intuitively higher, or more rare similarities.

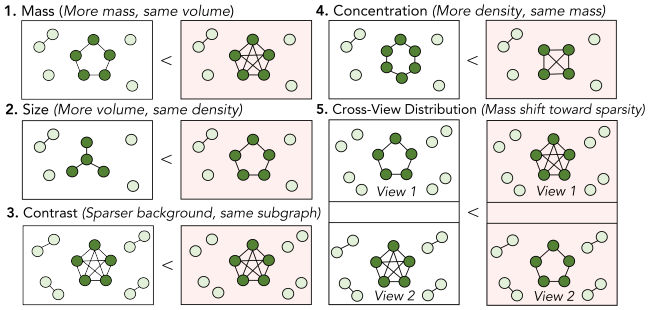


Figure 2: Illustration of Axioms 1-5 through toy examples.

3.2 Quantifying Group Suspiciousness

Given the above representation, our next aim is to define a means of quantifying the suspiciousness of an arbitrary multi-view subgraph (MVSG). This problem is important in practice, given its implications on manual review prioritization, and practitioner decision-making on enforcement and actioning against various discovered behaviors. Our basic intuition is that an subset of nodes (groups of entities) which are highly similar to each other (have considerable edge weights between them in multiple views) are suspicious. *But how can we compare across different MVSGs with varying edge weights, sizes and views in a principled way? For example, which is more suspicious: 5 organizations with the same IP address and URL, or 10 organizations with the same postal code and creation date?* The answer is not obvious; this motivates us to formally define criteria that any MVSG scoring metric *should* obey, and define principled ways of quantifying how suspiciousness increases or decreases in these contexts.

To do so, we first propose several preliminaries which aid us in formalizing these criteria. Informally, we consider an MVSG $\mathcal{X}$ of $n \leq N$ nodes and $k \leq K$ views as a subset of nodes, views or both from $\mathcal{G}$; we denote this relationship compactly as $\mathcal{X} \subseteq \mathcal{G}$, and sometimes abuse notation (when clear) to refer to the associated node set as $\mathcal{X}$. We introduce similar indexing notation as in the MVG case, such that $\mathcal{X}_i$ and $\mathcal{X}_{\vec{k}}$ refer to a specific subgraph view, and a subset of subgraph views ($\vec{k}$ is a K-length vector indicator) respectively. We define the *mass* of $\mathcal{X}_i$ as $c_i = \sum_{(a,b)\in\mathcal{X}^2} w_i^{(a,b)}$, which represents the total sum of edge weights for all edges between nodes in $\mathcal{X}$. We

define the *volume* of $\mathcal{X}_i$ as $v = n$ *choose* $2 = n(n-1)/2$, which denotes the possible number of edges between n nodes. Note that the volume of $\mathcal{X}_i$ is invariant to the view chosen and is only dependent on n (thus, we drop the subscript). Finally, we define the *density* of $\mathcal{X}_i$ as the ratio between its mass and volume, or $\rho_i = c_i/v_i$. We define analogs for mass, volume and density of the associated MVG $\mathcal{G}_i$ with C_i, V and P_i respectively. In general, upper-case variables denote properties of $\mathcal{G}$, while lower-case letters denote properties of $\mathcal{X}$. Given these terms, we propose the following axioms which should be satisfied by an MVSG scoring metric:

Axiom 1 (Mass). *Given two subgraphs $\mathcal{X},\mathcal{X}' \subseteq \mathcal{G}$ with the same volume, and same mass in all except one view s.t. $c_i > c_i'$, $\mathcal{X}$ is more suspicious. Formally,*

$$c_i > c_i' \Rightarrow f(n,\vec{c},N,\vec{C}) > f(n,\vec{c}',N,\vec{C})$$

Axiom 2 (Size). *Given two subgraphs $\mathcal{X},\mathcal{X}' \subseteq \mathcal{G}$ with same densities $\vec{p}$, but different volume s.t. $v > v'$, $\mathcal{X}$ is more suspicious. Formally,*

$$v > v' \Rightarrow \hat{f}(n,\vec{p},N,\vec{P}) > \hat{f}(n',\vec{p},N,\vec{P})$$

Axiom 3 (Contrast). *Given two subgraphs $\mathcal{X} \subseteq \mathcal{G}$, $\mathcal{X}' \subseteq \mathcal{G}'$ with same masses $\vec{c}$ and size v, s.t. $\mathcal{G}$ and $\mathcal{G}'$ have the same density in all except one view s.t. $P_i < P_i'$, $\mathcal{X}$ is more suspicious. Formally,*

$$P_i < P_i' \Rightarrow \hat{f}(n,\vec{p},N,\vec{P}) > \hat{f}(n,\vec{p},N,\vec{P}')$$

Axiom 4 (Concentration). *Given two subgraphs $\mathcal{X},\mathcal{X}' \subseteq \mathcal{G}$ with same masses $\vec{c}$ but different volume s.t. $v < v'$, $\mathcal{X}$ is more suspicious. Formally,*

$$v < v' \Rightarrow f(n,\vec{c},N,\vec{C}) > f(n',\vec{c},N,\vec{C})$$

Axiom 5 (Cross-view Distribution). *Given two subgraphs $\mathcal{X},\mathcal{X}' \subseteq \mathcal{G}$ with same volume v and same mass in all except two views i,j with densities $P_i < P_j$ s.t. $\mathcal{X}$ has $c_i = M, c_j = m$ and $\mathcal{X}'$ has $c_i = m, c_j = M$ and $M > m$, $\mathcal{X}$ is more suspicious. Formally,*

$$P_i < P_j \wedge c_i > c_i' \wedge c_j < c_j' \wedge c_i + c_j = c_i' + c_j' \Rightarrow f(n,\vec{c},N,\vec{C}) > f(n,\vec{c}',N,\vec{C})$$

Figure 2 illustrates these axioms via toy examples. Informally, these axioms assert that when other subgraph attributes are held constant, suspiciousness constitutes: higher mass (Axiom 1), larger volume with fixed density (Axiom 2), higher sparsity in overall graph (Axiom 3), higher density (Axiom 4), and more mass distributed in sparser views (Axiom 5). These axioms serve to formalize desiderata, drawing from intuitions stemming from prior works; however, prior works [16, 20] do not consider multi-view cases, and as we show later are unable to satisfy these axioms. Notably, such metrics produce unexpected results when scoring MVSGs, and thus lead to misaligned expectations in resulting rankings. We propose the following problem:

Problem 2 (MVSG Suspiciousness Scoring). ***Given*** *an MVG $\mathcal{G}$,* ***define*** *an MVSG scoring metric $f : \mathcal{S} \to \mathbb{R}$ over the set $\mathcal{S}$ of candidate MVSGs which satisfies Axioms 1-5.*

We discuss details of our solution in Section 4.

3.3 Mining Suspicious Groups

Given an MVSG scoring metric f, our next goal is to automatically extract MVSGs which score highly with respect to f. This is a challenging problem, as computing f for each possible MVSG in $\mathcal{G}$ is intractable; there are $2^N - 1$ non-empty candidate node subsets, and $2^K - 1$ non-empty view subsets over which to consider them. Clearly, we must resort to intelligent heuristics to mine highly suspicious MVSGs while avoiding an enumeration strategy.

We make several practical considerations in approaching this task. Firstly, since suspiciousness is clearly related to shared attribute behaviors, we propose exploiting our data representation to identify candidate "seeds" of nodes/entities which are promising to evaluate in terms of f. Secondly, we consider that (a) entities may not exhibit suspicious behaviors in all K views/attributes simultaneously, but rather only a subset, and (b) in evaluation, practitioners can only interpretably parse a small number of relationship types between a group at once; thus, we only focus on MVSGs $\mathcal{X}$ (wlog) such that $|\vec{k}|_1 = z$, where $z \leq K$ is generally small and can be suitably chosen and adapted according to empirical interpretability. In effect, this simplifies our problem as we only consider evaluating and mining MVSGs defined over z views, so that we consider *(K choose z)* total view subsets, rather than $2^K - 1$. We propose the following problem:

Problem 3 (Mining Suspicious MVSGs). ***Given*** *an MVG $\mathcal{G}$ on K views, a suspiciousness metric $f : \mathcal{S} \to \mathbb{R}$ over the set $\mathcal{S}$ of candidate MVSGs, and an interpretability constraint $z \leq K$,* ***find*** *the highest scoring MVSGs $\mathcal{X} \subseteq \mathcal{G}$ (wlog) for which $|\vec{k}|_1 = z$.*

We detail our approach and implementation strategy for solving this problem in Section 5.

4 PROPOSED SUSPICIOUSNESS METRIC

We propose an MVSG scoring metric based on an underlying data model for $\mathcal{G}$ in which undirected edges between the N nodes are distributed i.i.d within each of the K views. For single graph views, this model is widely known as the Erdös-Rényi model [28]. In our scenario, we consider two extensions unexplored by prior work: (a) we consider multiple graph views, and (b) we consider weighted cell values instead of binary ones. The first facet is necessary to support multi-attribute or multi-view settings, in which behaviors in one view may be very different than another (i.e. shared IP addresses may be much rarer than shared postal codes). The second facet is necessary to support continuous edge weights $w_i^{(a,b)} \in \mathbb{R}^+$ capable of richly describing arbitrarily complex notions of similarity between multiple entities (i.e. incorporating both number of shared properties, as well as their rarities). To handle these extensions, we propose the Multi-View Erdös-Rényi-Exponential model (MVERE):

Definition 1 (MVERE Model). *A multi-view graph $\mathcal{G}$ generated by the* MVERE *model is defined such that $w_i^{(a,b)} \sim Exp(\lambda_i)$ for $a \leftrightarrow b \in \mathcal{G}_i$.*

The MVERE model is a natural fit in our setting for several reasons. Firstly, the Exponential distribution is continuous and defined on support $\mathbb{R}^+$, which is intuitive as similarity is generally non-negative. Secondly, it has mode 0, which is intuitive given that sharing behaviors are sparse (most entities should not share properties), and the likelihood of observing high-similarity drops rapidly.

Given that there are $V = N(N-1)/2$ edges (including 0-weight edges) in each view, we can derive the closed-form MLE simply as $\lambda_i = N(N-1)/(2C_i) = V/C_i = P_i^{-1}$. From this, we can write the distribution of single-view MVSG mass as follows:

Lemma 4.1 (MVSG subgraph mass). *The mass M_i of a* MVERE*-distributed subgraph of $Exp(\lambda_i)$ follows $M_i \sim Gamma(v, P_i^{-1})$*

Proof. This follows from convolution of the Exponential distribution; given $M_i \sim Exp(\lambda)$, $\sum_{i=1}^{g} M_i \sim Gamma(g,\lambda)$. □

Table 2: Comparison with alternative metrics.

Axiom Adherence	Mass [23]	AvgDeg [8]	Dens [23]	SingVal [32]	CSSusp [17]	SliceNDice
Mass	✔	✔	✔	✔	?[1]	✔
Size	✔	✘	✘	✔	?[1]	✔
Contrast	✘	✘	✘	✘	?[1]	✔
Concentration	✘	✘	✔	✔	?[1]	✔
Cross-View Distr.	✘	✘	✘	✘	✘	✔

This enables us to define the *suspiciousness* of a given MVSG $\mathcal{X}$ across multiple views in terms of the likelihood of observing some quantity of mass in those views. Our metric is defined as

Definition 2 (MVSG Scoring Metric f). *The suspiciousness, f, of an MVSG $\mathcal{X}$ with $M_i \sim Gamma(v, P_i^{-1})$ and volume v is the negative log-likelihood of its mass $\vec{c}$ under the* MVERE *model:*

$$f\left(n,\vec{c},N,\vec{C}\right) = -\log\left(\prod_{i=1}^{K} Pr(M_i = c_i)\right)$$

We can write this metric in longer form as follows:

$$\begin{aligned} f\left(n,\vec{c},N,\vec{C}\right) &= -\log\left(\prod_{i=1}^{K} Pr(M_i = c_i)\right) \\ &= \sum_{i=1}^{K} -v\log\left(\frac{V}{C_i}\right) + \log\Gamma(v) - (v-1)\log c_i - \frac{Vc_i}{C_i} \\ &= \sum_{i=1}^{K} v\log\frac{C_i}{V} + v\log v - v - \log v - v\log c_i + \log c_i + \frac{Vc_i}{C_i} \end{aligned}$$

The last line is due to $\log\Gamma(v) = \log v! - \log v$, after which we use Stirling's approximation to simplify $\log v! \approx v\log v - v$. It is sometimes convenient to write suspiciousness in terms of densities $\vec{\rho}, \vec{P}$; thus, we also introduce a so-parameterized variant $\hat{f}$ where we use $\rho_i = c_i/v$ and $P_i = C_i/v$ and simplify as

$$\hat{f}\left(n,\vec{\rho},N,\vec{P}\right) = \sum_{i=1}^{K} v\log(P_i) - v\log\rho_i - v + \log\rho_i + v\frac{\rho_i}{P_i}$$

The intuition for this metric is that high suspiciousness is indicated by low probability of observing certain mass. Since we are interested MVSGs with unlikely *high* density (indicating synchrony between entities), we consider only cases where $\rho_i > P_i$ for all views, to avoid focusing on excessively sparse MVSGs.

Lemma 4.2 (Adherence to Axioms). *Our proposed suspiciousness metric f (and $\hat{f}$) satisfies each of the MVSG scoring Axioms 1-5.*

Proof. We give the full proofs in Section 8. □

4.1 Issues with Alternative Metrics

One may ask, *why not use previously established metrics of suspiciousness?* We next show that these metrics produce results which violate one or more proposed Axioms, and are thus unsuitable for our problem. We compare their performance on the toy settings from Figure 2. Each subgraph pair illustrates one of Axioms 1-5, and the shaded figures indicates higher intuitive suspiciousness. We consider 5 alternative metrics: mass (Mass) and density (Dens) [23], average degree (AvgDeg) [8], singular value (SingVal) [32] and CSSusp (metric from [17]).

[1] CSSusp is limited to discrete edge counts and cannot handle continuous mass settings.

Overview of Alternative Metrics. Prior work has suggested Mass, AvgDeg and Dens as suspiciousness metrics for single graph views [4, 16, 38]. We extend these to multi-view cases by construing an *aggregated* view with edge weights summed across the K views. [17] proposes CSSusp for suspiciousness in discrete, multi-modal tensor data; we can apply this by construing an MVSG $\mathcal{X}$ as a 3-mode tensor of $n \times n \times K$. In short, other metrics are agnostic to differences across views, hence they all *violate Axiom 5* (Cross-View Distribution). Below, we discuss the specifics of each alternative metric, and their limitations with respect to Axioms 1-4.

Mass: Mass is defined as $\sum_{i=1}^{K} c_i$, or the sum over all edge weights and views. Table 2 shows that it violates Axioms 2 (Size) and 3 (Contrast) by not considering subgraph size v or graph density P_i.

AvgDeg: Average degree is defined as $\sum_{i=1}^{K} c_i/n$, or average Mass per node. It does not consider subgraph density ρ_i and thus violates Axioms 2 (Size) and 4 (Concentration). It also violates Axiom 3 (Contrast), by not considering graph density P_i.

Dens: Density is defined as $\sum_{i=1}^{K} c_i/v$, or average Mass per edge, over v edges. It trivially violates Axiom 2 (Size) by not considering the ratio of subgraph density and size. It also violates Axiom 3 (Contrast) by not considering graph density P_i.

SingVal: Singular values are factor "weights" associated with the singular value decomposition $\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^T$; here, we consider the leading singular value $\Sigma_{0,0}$ over the view-aggregated $\mathbf{A}$. [34] shows that for i.i.d. cells in $\mathbf{A}$, $\Sigma_{0,0} = \sqrt{n\rho_i}$, though this does not hold generally. Under this assumption, the metric violates Axiom 3 (Contrast), by not considering graph density P_i.

CSSusp: [17] defines block suspiciousness as $-\log(Pr(M = \sum_{i=1}^{K} c_i))$, where M is subtensor mass under assumption that cells are discrete, Poisson draws. However, this constrains adherence to Axioms 1-4 (Mass, Size, Contrast and Concentration) only for discrete settings, and is unusable for continuous edge weights/cell values. This limitation is notable, as later shown in Sections 5.1 and 6.

5 PROPOSED ALGORITHM: SliceNDice

Given the metric defined in the previous section, we next aim to efficiently mine highly suspicious groups, as proposed in Problem 3. At a high-level, our approach is to start with a small MVSG over a few nodes and views, and expand it greedily, evaluating f until a local optimum is reached. Our main considerations are twofold: *How can we scalably expand a given seed until convergence?* and *How can we select good seeds in the first place?* We next address these questions.

Our goal is to find MVSGs $\mathcal{X} \subseteq \mathcal{G}$ (wlog) which score highest on f, and also meet the interpretability constraint $|\vec{k}|_1 = z$. As mentioned in Section 3.3, full enumeration is combinatorial and computationally intractable in large data. Thus, we resort to a greedy approach which allows scalable convergence to locally optimal MVSGs. Our approach, SliceNDice, is outlined in Algorithm 1.

In short, the algorithm begins by seeding an MVSG defined over a few nodes and z views according to some notion of suitability, and then takes utilizes an alternating maximization approach to improve the seed: the node set is kept fixed while the view set is updated, and subsequently the view set is kept fixed while the node set is updated. The updation steps only occur when f increases, and since suspiciousness is bounded, we ensure convergence to a local optimum. Next, we discuss updating and seeding choices, where we cover the referenced updation and seeding methods.

Updating choices. In order to find a highly suspicious group of entities, we aim to optimize the view set and node set selection via

the UpdateViews and UpdateNodes methods. UpdateViews can be written concisely as $\text{argmax}_{\vec{k}} f\left(n,\vec{c},N,\vec{C}\right)$, subject to $|\vec{k}|=z$. This is straightforward given our metric; we independently choose the top-z most suspicious views, given the fixed node set from the prior iteration. For UpdateNodes, we limit our search space to adding or removing a single node in the MVSG, which is dramatically more tractable than the 2^N-1 possible node set choices over $\mathcal{G}$. We write UpdateNodes concisely as $\text{argmax}_{\mathcal{X}'} f\left(n,\vec{c},N,\vec{C}\right)$, subject to $|\mathcal{X}'\setminus\mathcal{X}|+|\mathcal{X}\setminus\mathcal{X}'|\leq 1$, meaning that each update changes the node set by, at most, a single entity (one more or one less).

Seeding choices. Clearly, update quality relies on reasonable seeding strategy which is able to find candidate suspicious MVSGs and also explore $\mathcal{G}$. The SeedViews method can be achieved in multiple ways; ultimately, the goal is to chooses z initial views such that the seeds expand to a diverse set suspicious MVSGs. Given the desire for diversity, a reasonable approach is to sample z views uniformly as done in prior work [19, 38]. However, a downside with random sampling is that it does not respect our intuition regarding the non-uniform value of entity similarity across views. For example, consider that a view of *country* similarity has only 195 unique values, whereas a view of *IP Address* similarity has 2^{32} unique values; naturally, it is much more common for overlap in the former than the latter, despite the latter having a higher signal-to-noise ratio. Thus, in practice, we aim to sample views in a weighted fashion, favoring those in which overlap occurs less frequently. We considered using the inverse of view densities $\vec{\rho}$ as weights, but we observe that density is highly sensitive to outliers from skewed value frequencies. We instead use the inverse of the q^{th} frequency percentiles across views as more robust estimates of their overlap propensity ($q\geq 95$ works well in practice). The effect is that lower signal-to-noise ratio views such as *country* are more rarely sampled.

Defining SeedNodes is more challenging. As Section 4 mentions, we target overly dense MVSGs $\mathcal{X}$ (wlog) for which $\rho_i > P_i$ for all views. The key challenge is identifying seeds which satisfy this constraint, and thus offer promise in expanding to more suspicious MVSGs. Again, one could consider randomly sampling node sets and discarding unsatisfactory ones, but given that there are z constraints (one per view), the probability of satisfaction rapidly decreases as z increases (Section 6 elaborates). To this end, we propose a carefully designed, greedy seeding technique called GreedySeed (see Algorithm 2) which enables us to quickly discover good candidates. Our approach exploits that that satisfactory seeds occur when entities share more similarities, and strategically constructs a node set across views which share properties with each other. Essentially, we initialize a candidate seed with two nodes that share a similarity in a (random) one of the chosen views, and try to incrementally add other nodes connected to an existing seed node in views where the constraint is yet unsatisfied. If unable to do so after a number of attempts, we start fresh. The process is stochastic, and thus enables high seed diversity and diverse suspicious MVSG discovery.

5.1 Implementation

We describe several considerations in practical implementation, which improve result relevance and computational efficiency.

Result quality. The quality of resulting MVSGs depends on the degree to which they accurately reflect truly suspicious synchronous behaviors. In reality, not all synchronicity is equally suspicious; consider two accounts with uploaded files named *Test.jpg* and

Algorithm 1 SliceNDice

Require: MVG $\mathcal{G}$ (N nodes, K views, $\vec{C}$ masses), constraint $z \leq K$

```
k⃗ ← SeedViews(G, z)                          ▷ choose z views
X_k⃗ ← SeedNodes(G, k⃗)                        ▷ n nodes, c⃗ masses
S ← f(n, c⃗, N, C⃗)                            ▷ compute suspiciousness metric
do                                            ▷ alternating optimization
    S' ← S
    k⃗ ← UpdateViews(G, X)                     ▷ revise view set
    X_k⃗ ← UpdateNodes(G, X_k⃗)                ▷ revise node set
    S ← f(n, c⃗, N, C⃗)
while S > S'                                  ▷ repeat until S converges
return (X_k⃗, S)
```

Algorithm 2 GreedySeed

Require: MVG $\mathcal{G}$ (N nodes, K views, P dens.), views $\vec{k}$

```
define Shuffle(S): return S in random order
define Choose(S, r): return r random elements from S
V ← {i | 𝟙(k_i)}                             ▷ chosen view set
H^ve_i(a) ← a ⇒ A_i^{-1}(a) ∀a ∈ A, i ∈ V     ▷ value-to-entity hashmap
H^ev_i(a) ← a ⇒ A_i(a) ∀a ∈ G, i ∈ V          ▷ entity-to-value hashmap
i ← Choose(V, 1)                              ▷ choose a view
a ← Choose({a | |H^ve_i(a)| ≥ 2}, 1)          ▷ choose a shared value
X ← Choose(H^ve_i(a), 2)                      ▷ initialize seed with similar entities
for view i ∈ Shuffle(V) do
    t ← 0                                     ▷ keep track of attempts
    while (ρ_i < P_i and t < 20) do           ▷ try to satisfy constraint
        e_1 ∈ Shuffle(X, 1)                   ▷ choose entity already in X
        a ← Choose(H^ev_i(e_1), 1)            ▷ choose a shared value
        e_2 ← Choose(H^ve_i(a), 1)            ▷ choose a similar entity
        X ← X ∪ e_2                           ▷ grow seed
        t ← t + 1
    end while
    if (ρ_i < P_i) then
        go to 6                               ▷ start fresh if constraint not yet met
    end if
end for
return X_k⃗
```

Cheap-gucci-ad.jpg, which reflect different propensities for overlap. To avoid this type of *spurious* synchronicity, we use techniques from natural language processing to carefully weight similarities in the MVG construction. Firstly, we enable value blacklisting for highly common stop-words (e.g. *Test.jpg*). Overlap on a stopworded value produces 0 mass (no penalty). Next, we infer weights for other value overlaps according to the value inverse document (entity) frequencies (IEF) [1]. We define the IEF for value v in view i as: $ief(v_i) = \left(N/\log\left(1+|\mathcal{A}_i^{-1}(v_i)|\right)\right)^2$, which polarizes common and rare value importance. Then, we let $w_i^{(a,b)} = \sum_{v_i\in\mathcal{A}(a)\cap\mathcal{A}(b)} ief(v_i)$, so that edge weight between two nodes (entities) depends on both number and rarity of shared values. Finally, after mining and ranking many MVSGs, we aggregate and filter results for expert review by pruning "redundant" MVSGs which covering the same set of nodes; we use a Jaccard similarity threshold τ to determine overlap.

Computational efficiency. Next, we discuss several optimizations to improve the speed of suspicious MVSG discovery. Firstly, we observe that SliceNDice is trivially parallelizable. In our implementation, we are able to run thousands of seed generation and expansion processes simultaneously by running in a multi-threaded setting, and aggregating a ranked list afterwards. Another notable performance optimization involves the computation of view masses $\vec{c}$ in UpdateNodes, which is the slowest part of SliceNDice. A naïve approach to measure synchronicity given n nodes is to quadratically evaluate pairwise similarity, which is highly inefficient. We instead observe that it is possible to compute c_i by cheaply maintaining the number of value frequencies in a view-specific hashmap $\mathcal{J}_i$, such

that $\mathcal{J}_i(v) = |\{e \in \mathcal{X} \mid v \in \mathcal{A}_i(e)\}|$. Specifically, $\mathcal{J}_i(v)$ indicates that $\mathcal{J}_i(v)^2 - \mathcal{J}_i(v)$ similarities exist in the subgraph view on the value v, and since each of them contribute $ief(v)$ weight, we can write the total mass as $c_i = \sum_{v_i \in \mathcal{A}_i} ief(v_i)(\mathcal{J}_i(v_i))^2 - \mathcal{J}_i(v_i)$. This makes it possible to calculate view mass in linear time with respect to the number of subgraph nodes, which drastically improves efficiency.

6 EVALUATION

Our experiments aim to answer the following questions.

- **Q1. Detection performance:** How effective is SliceNDice in detecting suspicious behaviors in real and simulated settings? How does it perform in comparison to prior works?
- **Q2. Efficiency:** How do GreedySeed and SliceNDice scale theoretically and empirically on large datasets?

6.1 Datasets

We use both real and simulated settings in evaluation.

Snapchat advertiser platform. We use a dataset of advertiser organization accounts on Snapchat, a leading social media platform. Our data consists of $N = 230K$ organizations created on Snapchat from 2016-2019. Each organization is associated with several single-value (e.g. contact email) and multi-value attributes (e.g names of ad campaigns). All in all, we use $K = 12$ attributes deigned most relevant to suspicious behavior detection, given by Snapchat's Ad Review team:

- **Account details (6):** Organization name, e-mails, contact name, zip-code, login IP addresses, browser user agents.
- **Advertisement content (6):** Ad campaign name (e.g. *Superbowl Website*), media asset hashes, campaign headlines (e.g. *90% off glasses!*), brand name (e.g. *GAP*), iOS/Android app identifiers, and external URLs.

Based on information from the domain experts, we pruned 1.7K organizations from the original data, primarily including advertisement agencies and known affiliate networks which can have high levels of synchrony (often marketing for the same subsets of companies), and limited our focus to the remaining $228K$ organizations.

Simulated settings. We additionally considered several simulated attack settings, based on realistic attacker assumptions. Our intuition is that attack nodes will have higher propensity to share attribute values than normal ones, and may target varying attributes and have varying sophistication. Our simulation parameters include N (entity count) and K (total attribute types), $\vec{u}$ (length-K, cardinalities of attribute value spaces), n,k (entities and attributes per attack), c (number of attacks), λ (value count per normal entity), and τ (attack temperature, s.t. attackers choose from a restricted attribute space with cardinalities $\tau^{-1}\vec{u}$. Together, these parameters can express a wide variety of attack types. Our specific attack procedure is:

(1) Initialize N normal entities. For each attr. i, draw $Poisson(\lambda)$ specific values uniformly over $[1,u_i]$.
(2) Conduct an attack, by randomly sampling n entities and k attributes. For each attr. i, draw $Poisson(2\lambda)$ specific attr. values uniformly over $[1,\tau^{-1}u_i]$. Repeat c times.

Unless otherwise mentioned, for each scenario we fix parameters as $N = 500$ nodes, $K = 10$ views, $u_i = 50i$ attr. cardinality, $n = 50$ nodes per attack, $k = 3$ views per attack, $\lambda = 5$ mean values drawn per node and attribute, and $\tau = 10$ temperature.

6.2 Detection performance

We discuss performance in both settings.

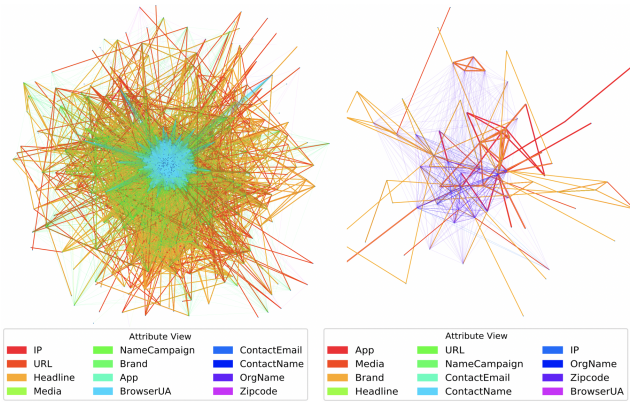
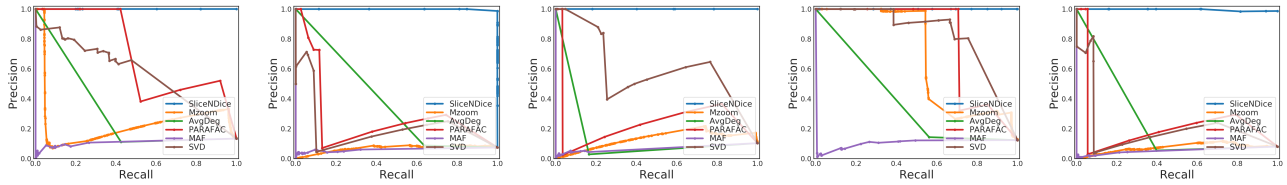


Figure 3: SliceNDice detects both blatant (left) and more stealthy (right) fraudsters on Snapchat's ad platform. Attribute Views in each Legend are sorted by suspiciousness (Red = Highest, Purple = Lowest).

Snapchat advertiser platform. We deployed SliceNDice on Google Cloud compute engine instances with high vCPU count to enable effective parallelization across many seeds over 2 days. During this time, we yielded a total of 6,050 suspected entity group behaviors within the Snapchat data. We fixed $z = 3$ for this run, based on both input from the Ad Review team, as well as the empirical observation that low z prevents discovery of more distributed, stealthier fraud which can only be uncovered by overlaying multiple graph views, whereas high z hurts domain expert interpretability and increases difficulty to satisfy density constraints; $z = 3$ balances these two extremes. We applied the pruning process (described in Section 5.1) using a Jaccard Index threshold of $\tau = .05$, such that the resulting subgraphs are not redundant, producing 534 distinct groups. We note that the number of distinct blocks was fairly robust to τ; for example, a more liberal threshold of $\tau = 0.25$ yielded 636 distinct groups. We chose τ conservatively in order to minimize redundancy in results shared with the Ad Review team. We evaluated our methods both quantitatively and qualitatively in coordination with them. We were not able to compare with other suspicious group mining methods which rely on block decomposition or search [8, 17, 30, 32, 38] as these are highly dense matrices/tensors which are too large to manifest for this scale.

Quantitative Evaluation: We sort the suspected sub-graphs in descending order and submitted those in the top 50 groups to the Ad Review team for in-depth, manual validation of the constituent 2736 organizations. We provided the domain experts with 3 assets to aid evaluation: (a) network visualizations like those in Figure 3, (b) mappings between the highest frequency occurring attribute values for each attribute, and all organization entities associated with those attributes, and (c) entity-level indices listing all instances of attribute synchrony and associated values, for each organization involved per group. From surveying these organizations, reviewers found that an overwhelming 2435 were connected to fraudulent behaviors that violated Snapchat's advertiser platform Terms of Service, resulting in an organization-level precision of 89%. The organizations spanned diverse behaviors, including individuals who created multiple accounts to making multiple (similar) accounts to generate impressions before defaulting early on their spending budget (avoiding payment), those selling counterfeit goods or running e-commerce scams, fraudulent surveys with falsely promised



(a) High attack sync. (b) Low attack sync. (c) High-signal attacks (d) Low-signal attacks (e) High-dim attacks

Figure 4: SliceNDice (blue, near the top) consistently achieves extremely high precision/recall on various, realistic attack settings, despite changing attack complexity. Other methods fail due to their inability to respect differences between attributes/views and overemphasis on global density.

rewards, and more. Intuitively, the diversity of these behaviors supports the flexibility of our problem formulation; despite SliceNDice's agnosticism to the multiple types of abuse, it was able to uncover them automatically using a shared paradigm.

Qualitative Evaluation: We inspect two instances of advertiser fraud, shown in Figure 3, with help from the Ad Review team. Although SliceNDice selected groups based on suspiciousness according to only the top ($z = 3$) views, we show the composites of similarities for all 12 attributes to illustrate the differences between the two attacks. On the left, we show the first case of *blatant* fraud across 500 organizations, which are connected across all of the considered attributes to varying degrees. Many of these organizations were accessed using a common subset of IP addresses, with the cluster near the center all sharing browser user agents. Upon further inspection, we find that these accounts are engaging in e-commerce fraud, and link to a series of URLs that follow the pattern *contact-usXX.[redacted].com*, where *XX* ranges from 01-27. Multiple accounts link to these common URLs and share common advertisement headlines, which combined with shared login activities ranks the group very highly according to our metric. Our second case study is of a smaller ring of 70 organizations, which could be considered *stealthy* fraud. These organizations appear to market a line of Tarot card and Horoscope applications. We noticed attempts taken by the fraudsters to cloak this ring from detection: no app identifier appears more than 4 times across the dataset, but most of the organizations have identifiers associating to similar app variants. This discovery illustrates SliceNDice's ability to "string together" shared properties across multiple attribute views to discover otherwise hard-to-discern behaviors.

Simulated settings. We consider detection performance on simulations matching several realistic scenarios.

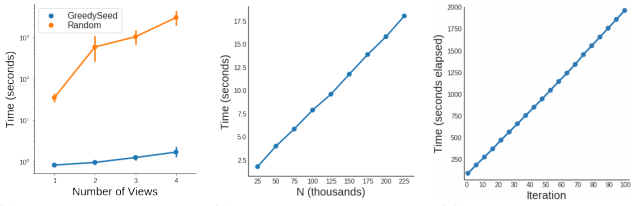
(1) **High attacker synchrony:** Default settings; attacks sample attributes from 1/10th of associated attribute spaces, making them much denser than the norm.
(2) **Low attacker synchrony:** $\tau = 2$. Attribute spaces are restricted to 1/2 and thus much harder to detect.
(3) **High-signal attribute attacks:** Attack views are sampled with weights $\propto \vec{u}$ (more likely to land in sparse views).
(4) **Low-signal attribute attacks:** Attack views are sampled with weights $\propto 1/\vec{u}$ (more likely to land in dense views).
(5) **Attacks in high dimensionality:** $K = 30$. Attacks are highly distributed in 3× higher dimensionality.

Our detection task is to classify each attribute overlap as suspicious or non-suspicious; thus, we aim to label each nonzero entry ("behavior") in the resulting $N \times N \times K$ tensor. We evaluate SliceNDice's precision/recall performance along with several state-of-the-art group detection approaches in this task. For each method, we penalize each behavior in a discovered block using the block score given by that method, and sum results over multiple detected blocks. The intuition is that a good detector will penalize behaviors associated with attack entities and attributes more highly. We compare against PARAFAC decomposition [25], MultiAspectForensics (MAF) [26], Mzoom [38], SVD [32], and AvgDeg [8]. For SVD and AvgDeg which only operate on single graph views, we use the aggregated view adjacency matrix, whereas for others we use the adjacency tensor. SliceNDice utilizes the compressed representation discussed in Section 5.1. For SVD, we use singular value (`SingVal`) as the block score. For PARAFAC and MAF, we use the block norm which is effectively the higher order `SingVal`. AvgDeg uses the average subgraph degree (`AvgDeg`), and Mzoom measures mass suspiciousness under the Poisson assumption (`CSSusp`). Section 8 gives further details regarding baseline comparison.

Figure 4 shows precision/recall curves for all 5 attack scenarios; note that SliceNDice significantly outperforms competitors in all cases, often maintaining over 90% precision while making limited false positives. Alternatives quickly drop to low precision for substantial recall, due to their inability to both (a) find and score groups while accounting for differences across attributes (Axiom 5), and (b) correctly discern the suspicious from non-suspicious views, even when the right subgraph is discovered. In practice, this allow attributes with low signal-to-noise ratio and higher natural density to overpower suspicious attacks which occur in less dense views.

6.3 Efficiency

We consider efficiency of both SliceNDice, as well as our seeding algorithm, GreedySeed. The time complexity of SeedViews is trivially $O(z)$, as it involves only choosing z random views. We can write the complexity of any suitable method for SeedNodes loosely as $O(z\bar{t})$, given z views and $\bar{t}$ iterations to satisfy each density constraint $\rho_i > P_i$ successively. However, in practice, this notion of $\bar{t}$ is ill-defined and can adversely affect performance. We explore practical performance in average time to find suitable seeds which satisfy constraints, for both random seeding and our GreedySeed: Figure 5a shows that our GreedySeed finds seeds 100−1000× faster than random seeding on real data; note the log scale. Random seeding struggles significantly in meeting constraints as the number of views increases, further widening the performance gap between the methods. Each iteration of UpdateNodes is $O(NK\bar{A})$ given N nodes, K views, and $\bar{A}$ values per attribute. UpdateViews is simply $O(z \log z)$, as UpdateNodes already updates subgraph masses $\vec{c}$ across all views. The runtime in practice is dominated by UpdateNodes; assuming $\bar{T}$ iterations per MVSG, the overall SliceNDice time complexity is $O(NK\bar{A}\bar{T})$. Figures 5b-5c show that SliceNDice scales



(a) Time w.r.t seeding **(b) Time w.r.t entities** **(c) Time w.r.t iters.**

Figure 5: Our GreedySeed finds suitable seeds $100-1000\times$ faster than random seeding (a). Moreover, SliceNDice scales linearly in number of entities (b) and iterations (c).

linearly with respect to both entities (over fixed iterations) and iterations (over fixed entities). Moreover, the overall space complexity is $O(K\bar{A})$, due to the compact attribute-oriented data representation described above. Note that alternative group mining methods which rely on block decomposition or search were infeasible to run on real data, due to the sheer attribute sharing density in the tensor representation, which grows quadratically with each shared value.

7 CONCLUSION

In this work, we tackled the problem of scoring and discovering suspicious behavior in multi-attribute entity data. Our work makes several notable contributions. Firstly, we construe this data as a multi-view graph, and formulate this task in terms of mining *suspiciously dense multi-view subgraphs* (MVSGs). We next propose and formalize intuitive desiderata (Axioms 1-5) that MVSG scoring metrics should obey to match human intuition, and designed a novel suspiciousness metric based on the proposed MVERE model which satisfies these metrics, unlike alternatives. Next, we proposed the SliceNDice algorithm, which enables scalable ranking and discovery of MVSGs suspicious according to our metric, and discussed practical implementation details which help result relevance and computational efficiency. Finally, we demonstrated strong empirical results, including experiments on real data from the Snapchat advertiser platform where we achieved 89% precision over 2.7K organizations and uncovered numerous fraudulent advertiser rings, consistently high precision/recall (over 97%) and outperformance of several state-of-the-art group mining algorithms, and linear scalability.

REFERENCES

[1] Charu C Aggarwal and ChengXiang Zhai. 2012. *Mining text data.* Springer Science & Business Media.
[2] Leman Akoglu, Mary McGlohon, and Christos Faloutsos. 2010. Oddball: Spotting anomalies in weighted graphs. In *PAKDD*. Springer, 410–421.
[3] Alessandro Bessi and Emilio Ferrara. 2016. Social bots distort the 2016 US Presidential election online discussion. (2016).
[4] Alex Beutel, Wanhong Xu, Venkatesan Guruswami, Christopher Palow, and Christos Faloutsos. 2013. Copycatch: stopping group attacks by spotting lockstep behavior in social networks. In *WWW*. ACM, 119–130.
[5] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *JSM* 2008, 10 (2008), P10008.
[6] Qiang Cao, Xiaowei Yang, Jieqi Yu, and Christopher Palow. 2014. Uncovering large groups of active malicious accounts in online social networks. In *CCS*. ACM, 477–488.
[7] Deepayan Chakrabarti, Spiros Papadimitriou, Dharmendra S Modha, and Christos Faloutsos. 2004. Fully automatic cross-associations. In *KDD*. ACM, 79–88.
[8] Moses Charikar. 2000. Greedy approximation algorithms for finding dense components in a graph. In *APPROX*. Springer, 84–95.
[9] Duen Horng "Polo" Chau, Carey Nachenberg, Jeffrey Wilhelm, Adam Wright, and Christos Faloutsos. 2011. Polonium: Tera-scale graph mining and inference for malware detection. In *SDM*. SIAM, 131–142.
[10] Liang Chen, Yipeng Zhou, and Dah Ming Chiu. 2015. Analysis and detection of fake views in online video services. *TOMM* 11, 2s (2015), 44.
[11] Inderjit S Dhillon, Subramanyam Mallela, and Dharmendra S Modha. 2003. Information-theoretic co-clustering. In *KDD*. ACM, 89–98.
[12] David Mandell Freeman. 2013. Using naive bayes to detect spammy names in social networks. In *WAIS*. ACM, 3–12.
[13] Gisel Bastidas Guacho, Sara Abdali, Neil Shah, and Evangelos E Papalexakis. 2018. Semi-supervised Content-based Detection of Misinformation via Tensor Embeddings. *ASONAM* (2018).
[14] Zoltán Gyöngyi, Hector Garcia-Molina, and Jan Pedersen. 2004. Combating web spam with trustrank. In *VLDB*. 576–587.
[15] Bryan Hooi, Neil Shah, Alex Beutel, Stephan Günnemann, Leman Akoglu, Mohit Kumar, Disha Makhija, and Christos Faloutsos. 2016. Birdnest: Bayesian inference for ratings-fraud detection. In *SDM*. SIAM, 495–503.
[16] Bryan Hooi, Hyun Ah Song, Alex Beutel, Neil Shah, Kijung Shin, and Christos Faloutsos. 2016. Fraudar: Bounding graph fraud in the face of camouflage. In *KDD*. ACM, 895–904.
[17] Meng Jiang, Alex Beutel, Peng Cui, Bryan Hooi, Shiqiang Yang, and Christos Faloutsos. 2016. Spotting suspicious behaviors in multimodal data: A general metric and algorithms. *TKDE* 28, 8 (2016), 2187–2200.
[18] Meng Jiang, Peng Cui, Alex Beutel, Christos Faloutsos, and Shiqiang Yang. 2014. CatchSync: Catching Synchronized Behavior in Large Directed Graphs. (2014).
[19] Meng Jiang, Peng Cui, Alex Beutel, Christos Faloutsos, and Shiqiang Yang. 2014. Inferring strange behavior from connectivity pattern in social networks. In *PAKDD*. Springer, 126–138.
[20] Meng Jiang, Peng Cui, Alex Beutel, Christos Faloutsos, and Shiqiang Yang. 2016. Inferring lockstep behavior from connectivity pattern in large graphs. *PAKDD* 48, 2 (2016), 399–428.
[21] Srijan Kumar and Neil Shah. 2018. False information on web and social media: A survey. *arXiv preprint arXiv:1804.08559* (2018).
[22] Hemank Lamba, Bryan Hooi, Kijung Shin, Christos Faloutsos, and Jürgen Pfeffer. 2017. zoo R ank: Ranking Suspicious Entities in Time-Evolving Tensors. In *ECML-PKDD*. Springer, 68–84.
[23] Victor E Lee, Ning Ruan, Ruoming Jin, and Charu Aggarwal. 2010. A survey of algorithms for dense subgraph discovery. In *Managing and Mining Graph Data*. Springer, 303–336.
[24] Justin Ma, Lawrence K Saul, Stefan Savage, and Geoffrey M Voelker. 2009. Beyond blacklists: learning to detect malicious web sites from suspicious URLs. In *KDD*. ACM, 1245–1254.
[25] Hing-Hao Mao, Chung-Jung Wu, Evangelos E Papalexakis, Christos Faloutsos, Kuo-Chen Lee, and Tien-Cheu Kao. 2014. MalSpot: Multi 2 malicious network behavior patterns analysis. In *PAKDD*. Springer, 1–14.
[26] Koji Maruhashi, Fan Guo, and Christos Faloutsos. 2011. Multiaspectforensics: Pattern mining on large-scale heterogeneous networks with tensor analysis. In *Advances in Social Networks Analysis and Mining (ASONAM), 2011 International Conference on*. IEEE, 203–210.
[27] Ahmed Metwally, Jia-Yu Pan, Minh Doan, and Christos Faloutsos. 2015. Scalable community discovery from multi-faceted graphs. In *BigData*. IEEE, 1053–1062.
[28] Mark EJ Newman, Duncan J Watts, and Steven H Strogatz. 2002. Random graph models of social networks. *PNAS* 99, suppl 1 (2002), 2566–2572.
[29] Shashank Pandit, Duen Horng Chau, Samuel Wang, and Christos Faloutsos. 2007. Netprobe: a fast and scalable system for fraud detection in online auction networks. In *WWW*. ACM, 201–210.
[30] Evangelos E Papalexakis, Leman Akoglu, and Dino Ienco. 2013. Do more views of a graph help? Community detection and clustering in multi-graphs.. In *FUSION*. Citeseer, 899–905.
[31] Jian Pei, Daxin Jiang, and Aidong Zhang. 2005. On mining cross-graph quasi-cliques. In *KDD*. ACM, 228–238.
[32] B Aditya Prakash, Ashwin Sridharan, Mukund Seshadri, Sridhar Machiraju, and Christos Faloutsos. 2010. Eigenspokes: Surprising patterns and scalable community chipping in large graphs. In *PAKDD*. Springer, 435–448.
[33] Neil Shah. 2017. FLOCK: Combating Astroturfing on Livestreaming Platforms. In *WWW*. IW3C2, 1083–1091.
[34] Neil Shah, Alex Beutel, Brian Gallagher, and Christos Faloutsos. 2014. Spotting suspicious link behavior with fbox. In *ICDM*. IEEE, 959–964.
[35] Neil Shah, Alex Beutel, Bryan Hooi, Leman Akoglu, Stephan Gunnemann, Disha Makhija, Mohit Kumar, and Christos Faloutsos. 2016. Edgecentric: Anomaly detection in edge-attributed networks. In *ICDMW*. IEEE, 327–334.
[36] Neil Shah, Danai Koutra, Tianmin Zou, Brian Gallagher, and Christos Faloutsos. 2015. Timecrunch: Interpretable dynamic graph summarization. In *KDD*. ACM, 1055–1064.
[37] Neil Shah, Hemank Lamba, Alex Beutel, and Christos Faloutsos. 2017. The Many Faces of Link Fraud. In *ICDM*. IEEE, 1069–1074.
[38] Kijung Shin, Bryan Hooi, and Christos Faloutsos. 2016. M-zoom: Fast dense-block detection in tensors with quality guarantees. In *ECML-PKDD*. Springer, 264–280.
[39] Peter K Smith, Jess Mahdavi, Manuel Carvalho, Sonja Fisher, Shanette Russell, and Neil Tippett. 2008. Cyberbullying: Its nature and impact in secondary school pupils. *J. of Child Psych.* 49, 4 (2008), 376–385.
[40] Kurt Thomas, Damon McCoy, Chris Grier, Alek Kolcz, and Vern Paxson. 2013. Trafficking Fraudulent Accounts: The Role of the Underground Market in Twitter Spam and Abuse.. In *USENIX Security*. 195–210.
[41] Cao Xiao, David Mandell Freeman, and Theodore Hwa. 2015. Detecting clusters of fake accounts in online social networks. In *WAIS*. ACM, 91–101.

8 REPRODUCIBILITY

8.1 Satisfaction of Axioms

Below, we show that our suspiciousness metric f (and $\hat{f}$) satisfies Axioms 1-5, and posited in Lemma 4.2. In each case, we consider how f or $\hat{f}$ changes as individual properties vary; since they are simply reparameterizations of one another, we suffice it to prove adherence to each axiom using the more convenient parameterization. We reproduce the axioms with each proof below for reader convenience.

Axiom (Mass). *Given two subgraphs $\mathcal{X},\mathcal{X}' \subseteq \mathcal{G}$ with the same volume, and same mass in all except one view s.t. $c_i > c_i'$, $\mathcal{X}$ is more suspicious. Formally,*

$$c_i > c_i' \Rightarrow f(n,\vec{c},N,\vec{C}) > f(n,\vec{c}',N,\vec{C})$$

Proof of Axiom 1 (Mass).

$$\frac{\partial f_i}{\partial c_i} = \frac{V}{C_i} - \frac{v-1}{c_i}$$

Because we are operating under the constraint $P_i < p_i$, then $P_i^{-1} > p_i^{-1}$ which implies that $\frac{V}{C_i} > \frac{v}{c_i} > \frac{v-1}{c_i}$. Therefore, $\frac{\partial f_i}{\partial c_i} > 0$ and so as mass increases, suspiciousness increases. □

Axiom (Size). *Given two subgraphs $\mathcal{X},\mathcal{X}' \subseteq \mathcal{G}$ with same densities $\vec{p}$, but different volume s.t. $v > v'$, $\mathcal{X}$ is more suspicious. Formally,*

$$v > v' \Rightarrow \hat{f}(n,\vec{p},N,\vec{P}) > \hat{f}(n',\vec{p},N,\vec{P})$$

Proof of Axiom 2 (Size). The derivative of $\hat{f}_i$ with respect to subgraph volume is:

$$\frac{\partial \hat{f}_i}{\partial v} = \log P_i - \log \rho_i - 1 + \frac{\rho_i}{P_i} = \frac{\rho_i}{P_i} - \log\frac{\rho_i}{P_i} - 1$$

This implies that $\frac{\partial \hat{f}_i}{\partial v} > 0$, because $x - \log x > 1$ always holds when $x > 1$ (which holds given $\rho_i > \rho_i$). Therefore, suspiciousness increases as volume increases. □

Axiom (Contrast). *Given two subgraphs $\mathcal{X} \subseteq \mathcal{G}$, $\mathcal{X}' \subseteq \mathcal{G}'$ with same masses $\vec{c}$ and size v, s.t. $\mathcal{G}$ and $\mathcal{G}'$ have the same density in all except one view s.t. $P_i < P_i'$, $\mathcal{X}$ is more suspicious. Formally,*

$$P_i < P_i' \Rightarrow \hat{f}(n,\vec{p},N,\vec{P}) > \hat{f}(n,\vec{p},N,\vec{P}')$$

Proof of Axiom 3 (Contrast). The derivative of $\hat{f}_i$ suspiciousness with respect to density P_i is:

$$\frac{\partial \hat{f}_i}{\partial P_i} = v\log P_i + v\frac{\rho_i}{P_i} = \frac{v}{P_i} - v\frac{\rho_i}{{P_i}^2} = \frac{v}{P_i}\left(1 - \frac{\rho_i}{P_i}\right)$$

This implies that $\frac{\partial \hat{f}_i}{\partial P_i} < 0$, because $v/P_i > 0$ and $\rho_i > P_i$. Thus, as graph density increases, suspiciousness decreases. Alternatively, as sparsity increases, suspiciousness increases. □

Axiom (Concentration). *Given two subgraphs $\mathcal{X},\mathcal{X}' \subseteq \mathcal{G}$ with same masses $\vec{c}$ but different volume s.t. $v < v'$, $\mathcal{X}$ is more suspicious. Formally,*

$$v < v' \Rightarrow f(n,\vec{c},N,\vec{C}) > f(n',\vec{c},N,\vec{C})$$

Proof of Axiom 4 (Concentration). The derivative of view i's contribution suspiciousness (parameterized by mass) and w.r.t the volume is:

$$\frac{\partial \hat{f}_i}{\partial v} = \log\frac{C_i}{V} + \log v + 1 - 1 - \frac{1}{v} - \log c_i = \log P_i + \log \rho_i^{-1} - \frac{1}{v}$$
$$= \log\frac{P_i}{\rho_i} - \frac{1}{v} = -\log\frac{\rho_i}{P_i} - \frac{1}{v}$$

$-\log\frac{\rho_i}{P_i} <= -1$ because $\frac{\rho_i}{P_i} > 1$, so therefore $\frac{\partial f_i}{\partial v} < 0$. Therefore, for a fixed sub-graph mass c_i, suspiciousness decreases as volume increases. □

Axiom (Cross-view Distribution). *Given two subgraphs $\mathcal{X},\mathcal{X}' \subseteq \mathcal{G}$ with same volume v and same mass in all except two views i,j with densities $P_i < P_j$ s.t. $\mathcal{X}$ has $c_i = M, c_j = m$ and $\mathcal{X}'$ has $c_i = m, c_j = M$ and $M > m$, $\mathcal{X}$ is more suspicious. Formally,*

$$P_i < P_j \wedge c_i > c_i' \wedge c_j < c_j' \wedge c_i + c_j = c_i' + c_j' \Rightarrow$$
$$f(n,\vec{c},N,\vec{C}) > f(n,\vec{c}',N,\vec{C})$$

Proof of Axiom 5 (Cross-View Distribution). Assume that view i is sparser than view j ($P_i < P_k$), and we are considering a subgraph which has identical mass in both views ($c_i = c_j$). Adding mass to the sparser view i will increase suspiciousness more than adding the same amount of mass to the denser view j because:

$$\frac{\partial f_i}{\partial v} - \frac{\partial f_j}{\partial v} = \left(\frac{V}{C_i} - \frac{v-1}{c_i}\right) - \left(\frac{V}{C_j} - \frac{v-1}{c_j}\right) = P_i^{-1} - P_j^{-1} > 0$$

□

8.2 Baseline Implementations for Comparison

We compared against 5 baselines in Section 6: PARAFAC [25], MAF [26], Mzoom [38], AvgDeg [8] and SVD [32]. Below, we give background and detail our implementations for these baselines.

8.2.1 PARAFAC. PARAFAC [30] is one of the most common tensor decomposition approaches, and can be seen as the higher-order analog to matrix singular value decomposition. An F-rank PARAFAC decomposition aims to approximate a multimodal tensor as a sum of F rank-one factors which, when summed, best reconstruct the tensor according to a Frobenius loss. In our case, the decomposition produces factor matrices $\mathbf{A}$ of $N \times r$, $\mathbf{B}$ of $N \times r$ and $\mathbf{C}$ of $K \times r$, such that we can write the tensor $\mathcal{T}$ associated with MVG $\mathcal{G}$ as

$$\mathcal{T} \approx \sum_{f=1}^{F} a^f \circ \mathbf{b}^f \circ c^f$$

where a^f denotes the f^{th} column vector of $\mathbf{A}$ (analogue for b^f and c^f), and $[\vec{a} \circ \vec{b} \circ \vec{c}](i,j,k) = a_i b_j c_k$. Since PARAFAC gives continuous scores per node in the $\vec{a}$ and $\vec{b}$ vectors for each rank-one factor, we sum them and then use the decision threshold suggested in [38] ($2.0/\sqrt{N}$) to mark nodes above that threshold as part of the block. We then select the top k views which individually have the highest singular value over the associated submatrix, and penalize the associated entries with the norm of the f^{th} rank-one tensor (closest approximation of `SingVal` in higher dimensions). We use the Python `tensorly` library implementation, and $F = 5$ decomposition.

8.2.2 MAF. MAF [26] also utilizes PARAFAC decomposition, but proposes a different node inclusion method. Their intuition is to look for the largest "bands" of nodes which have similar factor scores, as they are likely clusters. Since in our case, $\vec{a}$ and $\vec{b}$ both reflect node scores, we sum them to produce the resulting node factor scores. We then compute a log-spaced histogram over these using 20 bins (as proposed by the authors), and sort the histogram from highest to lowest frequency bins. We move down this histogram, including nodes in each bin until reaching the 90% energy criterion or 50% bin size threshold proposed by the authors. We mark these nodes as included in the block, and select the top k views with the highest associated submatrix singular values, as for PARAFAC. We

likewise penalize entries in this block using the associated block norm. We use the Python `tensorly` library implementation, and specify a $F=5$ decomposition.

8.2.3 Mzoom. Mzoom [38] proposes a greedy method for dense subtensor mining, which is flexible in handling various block-level suspiciousness metrics. It has been shown outperform [17] in terms of discovering blocks which maximize CSSusp metric, and hence we use it over the method proposed in [17]. The algorithm works by starting with the original tensor $\mathcal{T}$, and greedily shaves values from modes which maximize the benefit in terms of improving CSSusp. When a block $\mathcal{T}'$ is found which maximizes CSSusp over the local search, Mzoom prunes it from the overall tensor in order to not repeatedly converge to that block, and repeats the next iteration with $\mathcal{T}-\mathcal{T}'$. As Mzoom does not allow selection of a fixed k views, we only modify their implementation to add the constraint that for any blocks found with $\geq k$ views, we limit output to the first k for fairness. We penalize entries in each block with that block's CSSusp score. We use the authors' original implementation, which was written in Java (and available on their website) and specify 500 blocks to be produced (unless the tensor is fully deflated/empty sooner).

8.2.4 SVD. SVD [32], as discussed in Section 4.1, is a matrix decomposition method which aims to produce a low-rank optimal reconstruction of $\mathbf{A}$ according to Frobenius norm. In our case, since we aggregate over the K views and produce a resulting $N\times N$ matrix for $\mathcal{G}$, a rank F SVD decomposes the matrix $\mathbf{A}=\mathbf{U}\Sigma\mathbf{V}^T$, where $\mathbf{U}$,$\mathbf{V}$ are $N\times F$, and **Sigma** is $F\times F$ and diagonal, containing the singular values. Loosely, SVD effectively discovers low-rank structures in the form of clusters, such that $\mathbf{U}$,$\mathbf{V}$ indicate cluster affinity and Σ indicates cluster strength or scale. We take a similar approach as for PARAFAC, in that for each rank f, we sum the factor scores u^f and v^f and use a decision threshold of $2.0/\sqrt{1000}$ to mark node membership in the given block. Then, we rank the individual views according to their respective leading singular values, and choose the top k for inclusion. We then penalize entries in the block with the submatrix score. We use the Python `scipy` package, and specifically the *svds* method to run sparse SVD, for which we use an $F=5$ decomposition.

8.2.5 AvgDeg. [8] proposes an algorithm, which we call AvgDeg, for greedily mining dense subgraphs according to the AvgDeg notion of suspiciousness. The algorithm proposed gives a 2-approximation in terms of returning the maximally dense subgraph, and works by considering a single graph $\mathcal{G}$, and greedily shaving nodes with minimum degree while keeping track of the AvgDeg metric at each iteration. Upon convergence, the algorithm returns a subgraph which scores highly given AvgDeg. As for SVD, we consider $\mathcal{G}$ to be aggregated over all K views. Though the algorithm proposed by the author was initially defined in terms of unweighted graphs, we adapt it to weighted graph setting by shaving nodes greedily with minimum *weighted degree* rather than the adjacent edge count. Upon discovery of one subgraph $\mathcal{G}'$, we repeat the next iteration with $\mathcal{G}-\mathcal{G}'$. After finding the nodes for each subgraph, we choose the top k views with the highest individual AvgDeg for inclusion in the block. We request up to 500 blocks, but in practice find that the algorithm converges very quickly because it incorrectly prunes a large number of nodes in earlier iterations.

8.2.6 SliceNDice. We use the standard implementation as described in Section 5.1, evaluating over 500 blocks. Our implementation is written in Python, and will be made available publicly.