No Place to Hide: Catching Fraudulent Entities in Tensors

Yikun Ban*
Peking University
banyikun@pku.edu.cn

Xin Liu
Tsinghua University
liuxin16@mails.tsinghua.edu.cn

Yitao Duan
Fintec.ai
duan@fintec.ai

Xue Liu
McGill University
xueliu@cs.mcgill.ca

Wei Xu
Tsinghua University
weixu@tsinghua.edu.cn

ABSTRACT

Many approaches focus on detecting dense blocks in the tensor of multimodal data to prevent fraudulent entities (e.g., accounts, links) from retweet boosting, hashtag hijacking, link advertising, etc. However, no existing method is effective to find the dense block if it only possesses high density on a subset of all dimensions in tensors. In this paper, we novelly identify dense-block detection with dense-subgraph mining, by modeling a tensor into a weighted graph without any density information lost. Based on the weighted graph, which we call information sharing graph (ISG), we propose an algorithm for finding multiple densest subgraphs, D-Spot, that is faster (up to 11x faster than the state-of-the-art algorithm) and can be computed in parallel. In an N-dimensional tensor, the entity group found by the ISG+D-Spot is at least 1/2 of the optimum with respect to density, compared with the 1/N guarantee ensured by competing methods. We use nine datasets to demonstrate that ISG+D-Spot becomes new state-of-the-art dense-block detection method in terms of accuracy specifically for fraud detection.

CCS CONCEPTS

• **Information systems → Wrappers (data mining)**.

KEYWORDS

Dense-block Detection; Graph Algorithms; Fraud Detection

1 INTRODUCTION

Fraud represents a serious threat to the integrity of social or review networks such as Twitter and Amazon, with people introducing fraudulent entities (e.g., fake accounts, reviews, etc.) to gain more publicity/profit over a brief period. For example, on a social network or media sharing website, people may wish to enhance their account's popularity by illegally buying more followers [27]; on

---

*Yikun did the project during his visit at Tsinghua University.

e-commerce websites, fraudsters may register multiple accounts to benefit from "new user" promotions.

Consider the typical log data generated from a social review site (e.g., Amazon), which contains four-dimensional features: users, products, timestamps, rating scores. These data are often formulated as a tensor, in which each dimension denotes a separate feature and an entry (tuple) of the tensor represents a review action. Based on previous studies [12, 30], fraudulent entities form dense blocks (sub-tensors) within the main tensor, such as when a mass of fraudulent user accounts create an enormous number of fake reviews for a set of products over a short period. Dense-block detection has also been applied to network intrusion detection [20, 30], retweet boosting detection [12], bot activities detection [30], and genetics applications [20, 26].

Various dense-block detection methods have been developed. One approach uses tensor decomposition, such as CP decomposition and higher-order singular value decomposition [20]. However, as observed in [32], such methods are outperformed by search-based techniques [12, 30, 32] in terms of accuracy, speed, and flexibility regarding support for different density metrics. Furthermore, [30, 32] provide an approximation guarantee for finding the densest/optimal block in a tensor.

We have examined the limitations of search-based methods for dense-block detection. First, these methods are incapable of detecting *hidden-densest blocks*. We define a hidden-densest block as one that does not have a high-density signal on all dimensions of a tensor, but evidently has a high density on a subset of all dimensions. Moreover, existing methods neglect the data type and distribution of each dimension on the tensor. Assuming that two dense blocks A and B have the same density, however, A is the densest on a subset of critical features, such as IP address and device ID, whereas B is the densest on some trivial features such as age and gender. Can we simply believe that A is as suspicious as B? Unfortunately, the answer when using existing methods is 'yes.'

To address these limitations, we propose a dense-block detection framework and focus on entities that form dense blocks on tensors. The proposed framework is designed using a novel approach. Given a tensor, the formation of dense blocks is the result of value sharing (the behavior whereby two or more different entries share a distinct value (entity) in the tensor). Based on this key point, we propose a novel *Information Sharing Graph (ISG)* model, which accurately captures each instance of value sharing. The transformation from dense blocks in a tensor to dense subgraphs in ISG leads us to propose a fast, high-accuracy algorithm, D-Spot, for determining fraudulent entities with a provable guarantee regarding the densities of the detected subgraphs.

In summary, the main contributions of this study are as follows:

1) **[Graph Model].** We propose the novel ISG model, which converts every value sharing in a tensor to the representation of weighted edges or nodes (entities). Furthermore, our graph model considers diverse data types and their corresponding distributions based on information theory to automatically prioritize multiple features.

2) **[Algorithm].** We propose the *D-Spot* algorithm, which is able to find multiple densest subgraphs in one run. And we theoretically prove that the multiple subgraphs found by D-Spot must contain some subgraphs that are at least 1/2 as dense as the optimum. In real-world graphs, D-Spot is up to 11× faster than the state-of-the-art competing algorithm.

3) **[Effectiveness].** In addition to dense blocks, ISG+D-Spot also effectively differentiates hidden-densest blocks from normal ones. In experiments using eight public real-world datasets, ISG+D-Spot detected fraudulent entities more accurately than conventional methods.

2 BACKGROUND

2.1 Economics of Fraudsters

As most fraudulent schemes are designed for financial gain, it is essential to understand the economics behind the fraud. Only when the benefits to a fraudster outweigh their costs will they perform a scam.

To maximize profits, fraudsters have to share/multiplex different resources (e.g., fake accounts, IP addresses, and device IDs) over multiple frauds. For example, [13] found that many users are associated with a particular group of followers on Twitter; [36] identified that many cases of phone number reuse; [4] observed that the IP addresses of many spam proxies and scam hosts fall into a few uniform ranges; and [38] revealed that fake accounts often conduct fraudulent activities over a short time period.

Thus, fraudulent activities often form dense blocks in a tensor (as described below) because of this resource sharing.

2.2 Related Work

Search-based dense-block detection in tensors. Previous studies [12, 20, 30] have shown the benefit of incorporating features such as timestamps and IP addresses, which are often formulated as a multi-dimensional tensor. Mining dense blocks with the aim of maximizing a density metric on tensors is a successful approach. CrossSpot [12] randomly chooses a seed block and then greedily adjusts it in each dimension until the local optimum is attained. This technique usually requires enormous seed blocks and does not provide any approximation guarantee for finding the global optimum. In contrast to adding feature values to seed blocks, M-Zoom [30] removes feature values from the initial tensor one by one using a similar greedy strategy, providing a $1/N$-approximation guarantee for finding the optimum (where N is the number of dimensions in the tensor). M-Biz [31] also starts from a seed block and then greedily adds or removes feature values until the block reaches a local optimum. Unlike M-Zoom, D-Cube [32] deletes a set of feature values on each step to reduce the number of iterations, and is implemented in a distributed disk-based manner. D-Cube provides the same approximation guarantee as M-Zoom.

Table 1: ISG+D-Spot vs. existing dense-block detection methods.

	MAF [20]	CrossSpot [12]	M-zoom [30]	M-biz [31]	D-cube [32]	ISG+D-Spot
Applicable to N-dimensional data?	√	√	√	√	√	√
Catch densest blocks?	√	√	√	√	√	√
Catch hidden-densest blocks?	×	×	×	×	×	√
%-Approximation Guarantee?	×	×	1/N	×	1/N	1/2

Tensor decomposition methods. Tensor decomposition [17] is often applied to detect dense blocks within tensors [20]. Scalable algorithms, such as those described in [23, 33, 37], have been developed for tensor decomposition. However, as observed in [12, 32], these methods are limited regarding the detection of dense blocks, and usually detect blocks with significantly lower densities, provide less flexibility with regard to the choice of density metric, and do not provide any approximation guarantee.

Dense-subgraph detection. A graph can be represented by a two-dimensional tensor, where an edge corresponds to a non-zero entry in the tensor. The mining of dense subgraphs has been extensively studied [18]. Detecting the densest subgraph is often formulated as finding the subgraph with the maximum average degree, and may use exact algorithms [10, 16] or approximate algorithms [6, 16]. Fraudar [11] is an extended approximate algorithm that can be applied to fraud detection in social or review graphs. CoreScope [29] tends to find dense subgraphs in which all nodes have a degree of at least k. Implicitly, singular value decomposition (SVD) also focuses on dense regions in matrixes. EigenSpoke [24] reads scatter plots of pairs of singular vectors to find patterns and chip communities, [7] extracts dense subgraphs using a spectral cluster framework, and [14, 27] use the top eigenvectors from SVD to identify abnormal users.

Other anomaly/fraud detection methods The use of belief propagation [3, 22] and HITS-like ideas [8, 9, 13] is intended to catch rare behavior patterns in graphs. Belief propagation has been used to assign labels to the nodes in a network representation of a Markov random field [3]. When adequate labeled data are available, classifiers can be constructed based on multi-kernel learning [2], support vector machines [35], and k-nearest neighbor [34] approaches.

3 DEFINITIONS AND MOTIVATION

In this section, we introduce the notations and definitions used throughout the paper, analyze the limitations of existing approaches, and describe our key motivations.

3.1 Notation and Formulations

Table 2 lists the notations used in this paper. We use $[N] = \{1, ..., N\}$ for brevity. Let $\mathbf{R}(A_1, ..., A_N, X) = \{t_0, ..., t_{|X|}\}$ be a relation with N-dimensional features, denoted by $\{A_1, ..., A_N\}$, and a dimensional entry identifiers, denoted by X. For each entry (tuple) $t \in \mathbf{R}$, $t = (a_1, ..., a_N, x)$, where $\forall n \in [N]$, we use $t[A_n]$ to denote the value of A_n in t, $t[A_n] = a_n$ and $t[X]$ to denote the identification of t, $t[X] = x$, $x \in X$. We define the mass of $\mathbf{R}$ as $|\mathbf{R}|$, which is the total number of such entries, $|\mathbf{R}| = |X|$. For each $n \in [N]$, we use $\mathbf{R}_n$ to

Table 2: Symbols and Definitions

Symbol	Interpretation
N	number of dimensions in a tensor
$[N]$	set $\{1,...,N\}$
$\mathbf{R}(A_1, ..., A_N, X)$	relation representing a tensor
A_n	n-th dimensional values of $\mathbf{R}$
$\mathbf{R}_n$	set of distinct values of A_n of $\mathbf{R}$
$t = (a_1, ..., a_N, x)$	an entry (tuple) of $\mathbf{R}$
$\mathcal{B}(\mathcal{A}_1, ..., \mathcal{A}_N, \mathcal{X})$	a block in $\mathbf{R}$
$\mathcal{B}_n$	set of distinct values of $\mathcal{A}_n$ of $\mathcal{B}$
U	target dimension in $\mathbf{R}$
$\mathbf{V} = \{u_1, ..., u\}$	set of distinct values of U
$\mathbf{G} = (\mathbf{V}, \mathbf{E})$	Information Sharing Graph
$\mathcal{S}_{i,j}$	$\mathcal{S}$-score between u_i and u_j
$\mathcal{S}_i$	$\mathcal{S}$-score of u_i
$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	subgraph in $\mathbf{G}$
$\mathcal{F}$	density metric

denote the set of distinct values of A_n. Thus, $\mathbf{R}$ naturally represents an N-dimensional tensor of size $|\mathbf{R}_1| \times ... \times |\mathbf{R}_N|$.

A block $\mathcal{B}$ in $\mathbf{R}$ is defined as $\mathcal{B}(\mathcal{A}_1, ..., \mathcal{A}_N, \mathcal{X}) = \{t \in \mathbf{R} : t[X] \in \mathcal{X}\}$ and $\mathcal{X} \subseteq X$. Additionally, the mass $|\mathcal{B}|$ is the number of entries of $\mathcal{B}$ and $\mathcal{B}_n$ is the set of distinct values of $\mathcal{A}_n$. Let $\mathcal{B}(a, A_n) = \{t \in \mathbf{R} : t[A_n] = a\}$ represent all entries that take the value a on A_n. The mass $|\mathcal{B}(a, A_n)|$ is the number of such entries. A simple example is given as follows.

Example 1 (Amazon review logs). *Assume a relation* $\mathbf{R}(\underline{user}, \underline{product}, \underline{timestamp}, X)$, *where* $\forall t \in \mathbf{R}$, $t = (a_1, a_2, a_3, x)$ *indicates a review action where user* a_1 *reviews product* a_2 *at timestamp* a_3, *and the identification of the action is* x. *Because* a_1 *may review* a_2 *at* a_3 *(we assume that* a_3 *represents a period) multiple times, X helps us distinguish each such action. The mass of* $\mathbf{R}$, *denoted by* $|\mathbf{R}|$, *is the number of all review actions in the dataset. The number of distinct users in* $\mathbf{R}$ *is* $|\mathbf{R}_1|$. *A block* $\mathcal{B}(a_1, user)$ *is the set of all rating entries operated by user* a_1, *and the number of overall entries of* $\mathcal{B}(a_1, user)$ *is* $|\mathcal{B}(a_1, user)|$.

First, we present a density metric that is known to be useful for fraud detection [30, 32]:

Definition 1. (Arithmetic Average Mass ρ). Given a block $\mathcal{B}(\mathcal{A}_1, ..., \mathcal{A}_N, \mathcal{X})$, the arithmetic average mass of $\mathcal{B}$ on dimensions $\mathcal{N}$ is

$$\rho(\mathcal{B}, \mathcal{N}) = \frac{|\mathcal{B}|}{\frac{1}{|\mathcal{N}|} \sum_{n \in \mathcal{N}} |\mathcal{B}_n|},$$

where $\mathcal{N}$ is a subset of $[N]$ and obviously $\rho \in [1.0, +\infty)$.

If block $\mathcal{B}$ *is dense in* $\mathbf{R}$, *then* $\rho(\mathcal{B}, [N]) > 1.0$.

Other density metrics listed in [32] are also effective for fraud detection. It is broadly true that all density measures are functions of the cardinalities of the dimensions and masses of $\mathcal{B}$ and $\mathbf{R}$. In $\mathbf{R}$, previous studies [12, 30–32] have focused on detecting the top-k densest blocks in terms of a density metric. In the remainder of this paper, we use the density metric ρ to illustrate our key points.

3.2 Shortcomings of Existing Approaches and Motivation

In practice, the blocks formed by fraudulent entities in $\mathbf{R}$ may be described by *hidden-densest blocks*. To illustrate hidden-densest blocks, we present the following definitions and examples.

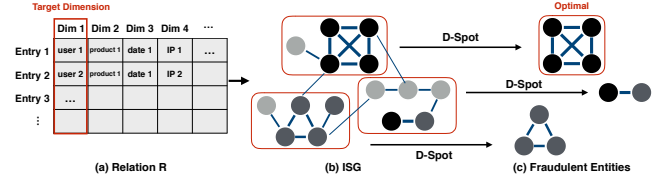**Figure 1: Workflow of ISG+D-Spot.**

Definition 2. In $\mathbf{R}(A_1, ..., A_N, X)$, we say that $\mathcal{B}(\mathcal{A}_1, ..., \mathcal{A}_N, \mathcal{X})$ is the densest on a dimension A_n if $\rho(\mathcal{B}, \{n\})$ is the maximal value of all possible $\rho(\hat{\mathcal{B}}, \{n\})$, where $\hat{\mathcal{B}}$ is any possible block in $\mathbf{R}$.

Definition 3. (Hidden-Densest Block). In $\mathbf{R}(A_1, ..., A_N, X)$, $\mathcal{B}(\mathcal{A}_1, ..., \mathcal{A}_N, \mathcal{X})$ is the hidden-densest block if $\mathcal{B}$ is the densest on a small subset of $\{A_1, ..., A_N\}$.

Example 2 (Registration logs). *In a registration dataset with 19 features, fake accounts only exhibit conspicuous resource sharing with respect to the IP address feature.*

Example 3 (TCP dumps). *The DARPA dataset [1] has 43 features, but the block formed by malicious connections is only the densest on two features.*

Thus, catching hidden-densest blocks has a significant utility in the real world. Unfortunately, the problem is intractable using existing approaches [12, 30–32].

First, assuming that the hidden-densest block $\mathcal{B}(\mathcal{A}_1, ..., \mathcal{A}_N, \mathcal{X})$ is only the densest on dimension A_N, we have that

$$\rho(\mathcal{B}, [N-1]) = \frac{|\mathcal{B}|}{\frac{1}{N-1} \sum_{n \in [N-1]} |\mathcal{B}_n|} \approx \rho(\mathcal{B}, [N]) = \frac{|\mathcal{B}|}{\frac{1}{N} \sum_{n \in [N]} |\mathcal{B}_n|}$$

when N is sufficiently large. Assuming $\rho(\mathcal{B}, [N-1])$ is very low, then the methods in [12, 30–32], which try to find the block $\mathcal{B}$ that maximizes $\rho(\mathcal{B}, [N])$, have a limited ability to detect the hidden-densest block.

Second, consider a block $\mathcal{B}$ formed by fraudulent entities, in which $\mathcal{B}$ is only the densest on $\{A_2, A_3, A_5\}$, and thus $\rho(\mathcal{B}, \{2, 3, 5\})$ is maximal. However, the techniques of [12, 30–32] cannot find $\{A_2, A_3, A_5\}$ when the feature combinations are exploded.

Furthermore, in $\mathbf{R}(A_1, ..., A_N, X)$, consider two blocks $\mathcal{B}_1$ and $\mathcal{B}_2$, where $\mathcal{B}_1$ is the densest on A_i, $\mathcal{B}_2$ is the densest on A_j, and $\rho(\mathcal{B}_1, [N]) = \rho(\mathcal{B}_2, [N])$. Does this indicate that $\mathcal{B}_1$ and $\mathcal{B}_2$ are equally suspicious? No, absolutely not, because A_i could be the IP address feature and A_j could be a trivial feature such as the user's age, location, or gender.

[Value Sharing]. Based on the considerations above, we design our approach from a different angle. The key reason behind the formation of dense blocks is value sharing. Given $t_1 \in \mathbf{R}$, a dimension A_n, and $t_1[A_n] = a$, we can identify value sharing when $\exists t_2 \in \mathbf{R}$, $t_2 \neq t_1$, and $t_2[A_n] = a$.

Obviously, if a block $\mathcal{B}$ is dense, $\rho(\mathcal{B}, [N]) > 1.0$, then value sharing must be occurring, i.e., **value sharing results in dense blocks**.

Therefore, detecting dense blocks is equivalent to catching value sharing signals. We propose ISG based on information theory and design the D-Spot algorithm to leverage graph features, allowing us to catch fraudulent entities within dense blocks and overcome the limitations mentioned above.

4 ISG BUILDING

In this section, we present the Information Sharing Graph (ISG), which is constructed on the relation **R**.

4.1 Problem Formulation

Catching fraudulent entities is equivalent to detecting a subset of distinct values in a certain dimension. Let U denote the target dimension in which a subset of distinct values form the fraudulent entities we wish to detect. In $\mathbf{R} = (A_1, ..., A_N, X)$, **we choose a dimension and set it as** U, **and denote the remaining** $(N-1)$ **dimensions as** K **dimensions**, $k \in [K]$, for brevity. We build the ISG of U, i.e., the weighted-undirected graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, in which $\mathbf{V} = \{u_1, ..., u_n\}$ is the set of distinct values of U.

In Example 1, $\mathbf{R}(\underline{user}, product, timestamp, X)$, we set $U = \underline{user}$ if we wish to detect fraudulent user accounts. In Example 2, we set $U = \underline{account}$ if we would like to identify fake accounts. In Example 3, we set $U = \underline{connection}$ to catch malicious connections.

To specifically describe the process of value sharing, we present the two following definitions:

Definition 4. (Pairwise Value Sharing). Given $u_i, u_j \in \mathbf{V}$ and $a \in A_k$, we say that u_i and u_j share value a on A_k if $\exists t_1, t_2 \in \mathbf{R}$ such that $t_1[U] = u_i$, $t_2[U] = u_j$ and $t_1[A_k] = t_2[A_k] = a$.

Pairwise value sharing occurs when a distinct value is shared by multiple individual entities. Given a value sharing process in which a is shared by $\mathcal{V} \subset \mathbf{V}$, we denote this as $\frac{|\mathcal{V}|(|\mathcal{V}|-1)}{2}$ pairwise value sharing.

Definition 5. (Self-Value Sharing). Given $t_1 \in \mathbf{R}$, where $t_1[U] = u_i$, $u_i \in \mathbf{V}$, and $t_1[A_k] = a$, we say that u_i shares value a on A_k if $\exists t_2 \in \mathbf{R}$ and $t_2 \neq t_1$ such that $t_2[U] = u_i$ and $t_2[A_k] = a$.

Another type of value sharing occurs when the distinct value a is shared n times by an entity u_i, which can be represented by n instances of self-value sharing.

In ISG $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, for some edge $(u_i, u_j) \in \mathbf{E}$, $\mathcal{S}_{i,j}$ represents the information between u_i and u_j derived from the other K dimensions, and for some node $u_i \in \mathbf{V}$, $\mathcal{S}_i$ denotes the information of u_i calculated from the other K dimensions. From the definitions and notations defined in the previous section, Problem 1 gives a formal definition of how to build the ISG of a tensor.

Problem 1 (Building a pairwise information graph). **(1) Input:** *a relation* $\mathbf{R}$, *the target dimension* U, **(2) Output:** *the information sharing graph* $\mathbf{G} = (\mathbf{V}, \mathbf{E})$.

4.2 Building an ISG

Given a dimension A_k, the target dimension U, any $u_i \in \mathbf{V}$, and an entry $t_1 \in \mathbf{R}$ for which $t_1[U] = u_i$, then for each $a \in A_k$, we assume that the probability of $t_1[A_k] = a$ is $p^k(a)$.

Edge Construction. Based on information theory [28], the self-information of the event that u_i and u_j share a is:

$$\mathcal{I}_{i,j}^k(a) = \log(\frac{1}{p^k(a)})^2. \tag{1}$$

To compute the pairwise value sharing between u_i and u_j across all K dimensions, we propose the metric $\mathcal{S}$-score as the edge weight of ISG:

$$\mathcal{S}_{i,j} = \sum_{k=1}^{K} \sum_{a \in H_k(u_i, u_j)} \mathcal{I}_{i,j}^k(a), \tag{2}$$

where $H_k(u_i, u_j)$ is the set of all values shared by u_i and u_j on A_k. Note that $\mathcal{S}_{i,j} = 0.0$ if $\bigcup_{k=1}^{K} H_k(u_i, u_j) = \emptyset$.

Intuitively, if u_i and u_j *do not* have any shared values, which is to be expected in normal circumstances, we have zero information. Otherwise, we obtain some information. Thus, the higher the value of $\mathcal{S}_{i,j}$ is, the more similar u_i is to u_j. In practice, the $\mathcal{S}$-score has a large variance. For example, fraud user pairs sharing an IP subnet and device ID will have a high $\mathcal{S}$-score, whereas normal users are unlikely to share these values with anyone, and will thus have an $\mathcal{S}$-score close to zero. Additionally, the information we obtain for u_i and u_j sharing the value a is related to the overall probability of that value. For example, it would be much less surprising if they both follow Donald Trump on Twitter than if they both follow a relatively unknown user.

Node Setting. For a node $u_i \in \mathbf{V}$, let $\mathcal{B}(a, A_k, u_i, U)$ be the set $\{t \in \mathbf{R} : (t[A_k] = a) \wedge (t[U] = u_i)\}$. When $|\mathcal{B}(a, A_k, u_i, U)| \geq 2$. The information of forming $\mathcal{B}(a, A_k, u_i, U)$ is:

$$\mathcal{I}_i^k(a) = \log(\frac{1}{p^k(a)})^{|\mathcal{B}(a, A_k, u_i, U)|}. \tag{3}$$

We now define $\mathcal{S}_i$ to compute the self-value sharing for u_i across all K dimensions:

$$\mathcal{S}_i = \sum_{k=1}^{K} \sum_{\mathcal{B} \in H_k(u_i)} \mathcal{I}_i^k(a), \tag{4}$$

where $H_k(u_i)$ is the set $\{\mathcal{B}(a, A_k, u_i, U), \forall a \in \mathbf{R}_k\}$ and $\mathbf{R}_k$ is the set of distinct values of A_k. Note that $\mathcal{S}_i = 0.0$ if $\bigcup_{k=1}^{K} H_k(u_i) = \emptyset$.

In effect, self-value sharing occurs in certain fraud cases. For instance, a fraudulent user may create several fake reviews for a product/restaurant on Amazon/Yelp [25] over a few days. In terms of network attacks [19], a malicious TCP connection tends to attack a server multiple times.

Determining [$p^k(a)$]. We can extend the $\mathcal{S}$-score to accommodate different data types and distributions.

It is difficult to determine $p^k(a)$, as we do not always know the distribution of A_k. In this case, for dimensions that are attribute features, we assume a uniform distribution and simply set

$$p^k(a) = 1/|\mathbf{R}_k|. \tag{5}$$

This approximation works well for many fraud-related properties such as IP subnets and device IDs, which usually follow a Poisson distribution [12].

However, the uniform assumption works poorly for low-entropy distributions, such as the long-tail distribution, which is common in dimensions such as items purchased or users followed. Low entropy implies that many users behave similarly anyway, independent of frauds. Intuitively for such distributions, there is no surprise in following a celebrity (head of the distribution), but considerable information if they both follow someone at the tail. For example, 20% of users correspond to more than 80% of the "follows" in online social networks. The dense subgraphs between celebrities and their fans are very unlikely to be fraudulent. If feature A_k has a long-tail distribution, its entropy is very low. For example, the entropy of the uniform distribution over 50 values is 3.91, but the entropy of a

long-tail distribution with 90% of probabilities centered around one value is only 0.71. Therefore, we set $p^k(a)$ based on the empirical distribution as

$$p^k(a) = |\mathcal{B}(a, A_k)|/|\mathbf{R}|, \qquad (6)$$

when the values in A_k have low entropy. We also provide an interface so that users can define their own $p^k(a)$ function.

Optimization of ISG Construction. In theory, a graph with $|\mathbf{V}|$ nodes has $O(|\mathbf{V}|^2)$ edges. Naively, therefore, it takes $O(|\mathbf{V}|^2)$ time for graph initialization and traversal.

To reduce the complexity of building the ISG, we use the *key-value* approach. The *key* corresponds to a value a on A_k and the *value* represents the block $\mathcal{B}(a, A_k)$. Let $\mathcal{V} \subseteq \mathbf{V}$ denote the entities that occur in $\mathcal{B}(a, A_k)$. As each pair $(u_i, u_j) \in \mathcal{V}$ shares a, we increase the value of $\mathcal{S}_{i,j}$ by $\mathcal{I}_{i,j}^k(a)$. Additionally, for each $u_i \in \mathcal{V}$, there exists some $\mathcal{B}(a, A_k, u_i, U) \subseteq \mathcal{B}(a, A_k)$. Thus, we increase the value of $\mathcal{S}_i$ by $\mathcal{I}_i^k(a)$ if $|\mathcal{B}(a, A_k, u_i, U)| \geq 2$.

To build the ISG, we compute all *key-value* pairs across K dimensions by traversing $\mathbf{R}$ in parallel. Thus, it takes $O(K|\mathbf{R}| + |\mathbf{E}|)$ time to build the graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$. Note that we only retain positive $\mathcal{S}_{i,j}$ and $\mathcal{S}_i$. In practice, $\mathbf{G}$ is usually sparse, which is discussed in Section 6.1.

4.3 Key Observations on ISG

Given a relation $\mathbf{R} = (A_1, ..., A_N, X)$ in which we set $U = A_N$, we construct the ISG of U, $\mathbf{G} = (\mathbf{V}, \mathbf{E})$. Assuming there is a **fraudulent block** $\mathcal{B}$ in $\mathbf{R}$, $\mathcal{B} = (\mathcal{A}_1, ..., \mathcal{A}_N, \mathcal{X})$ is transformed into a subgraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ in $\mathbf{G}$, where $\mathcal{V}$ is the set of distinct values of $\mathcal{A}_N$ and an edge $\mathcal{S}_{i,j} \in \mathcal{E}$ denotes the information between u_i and u_j calculated from the other K dimensions. Then, $\mathcal{V}$ is the fraud group comprised of fraudulent entities that we wish to detect.

We summarize three critical observations of $\mathcal{G}$ that directly lead to the algorithms presented in Section 5.2. Given $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we define the **edge density** of $\mathcal{G}$ as

$$\rho_{edge}(\mathcal{G}) = \frac{|\mathcal{E}|}{|\mathcal{V}|(|\mathcal{V}| - 1)}$$

1) The value of $\mathcal{S}_{i,j}$ or $\mathcal{S}_i$ is unusually high. Value sharing may happen frequently, but sharing across certain features, even certain values, is more suspicious than others. Intuitively, it might be suspicious if two users share an IP address or follow the same random "nobody" on Twitter. However, it is not so suspicious if they have a common gender, city, or follow the same celebrity. In other words, certain value sharing is likely to be fraudulent because the probability of sharing across a particular dimension, or at a certain value, is quite low. Thus, the information value is high, which is accurately captured by $\mathcal{S}_{i,j}$ and $\mathcal{S}_i$.

2) $|\mathcal{V}|$ is usually large. Fraudsters perform the same actions many times to achieve economies of scale. Thus, we expect to find multiple pairwise complicities among fraudulent accounts. A number of studies have found that large cluster sizes are a crucial indicator of fraud [5, 38]. Intuitively, while it is natural for a few family members to share an IP address, it is highly suspicious when dozens of users share one.

3) The closer $\rho_{edge}(\mathcal{G})$ is to 1.0, the more suspicious $\mathcal{G}$ is. Fraudsters usually operate a number of accounts for the same job, and thus it is likely that users manipulated by the same fraudster

will share the same set of values. Thus, the $\mathcal{G}$ formed by the fraud group will be well-connected.

Appearance of legitimate entities on ISG. In $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, given some u_i that we assume to be **legitimate**, let $h(u_i)$ denote the set of its neighbor nodes. We have two findings. (1) For u_i, $\mathcal{S}_i + \sum_{u_j \in \mathbf{V}} \mathcal{S}_{i,j} \to 0$ because u_i is unlikely to share values with others. Even if exists, the shared values should have a high probability (see observation 1) and therefore small $\mathcal{S}_{i,j}$. (2) The subgraph $\mathcal{G}$ induced by $h(u_i)$ is typically not well-connected, as resource sharing is uncommon in the real world. If $\mathcal{G}$ is well-connected, $|h(u_i)|$ is quite small compared with the fraud group size (see observation 2).

In summary, the techniques described in [12, 20, 30–32] work directly on the tensor, indicating that they consider value sharing on each dimension, and even certain values, as equivalent. In contrast, ISG assigns each instance of value sharing a theoretical weight based on the edges and nodes of the ISG, which is more effective for identifying the (hidden-) densest blocks (comparison in Sec.6.2).

5 SPOTTING FRAUD

Based on the observations in Section 4.3, we now describe our method for finding objective subgraphs in $\mathbf{G}$. This section is divided into two parts: first, we define a density metric $\mathcal{F}_\mathcal{G}$, and then we illustrate the proposed D-Spot algorithm.

5.1 Density Metric and Problem Definition

To find the objective $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we define a density metric $\mathcal{F}_\mathcal{G}$ as [6, 11]:

$$\mathcal{F}_\mathcal{G} = \frac{\sum_{(u_i, u_j) \in \mathcal{E}} \mathcal{S}_{i,j} + \sum_{u_i \in \mathcal{V}} \mathcal{S}_i}{|\mathcal{V}|}. \qquad (7)$$

The form of $\mathcal{F}_\mathcal{G}$ satisfies the three key observations of $\mathcal{G}$ in Section 4.3.

(1) Keeping $|V|$ fixed, we have that $\sum_{(u_i, u_j) \in \mathcal{E}} \mathcal{S}_{i,j} + \sum_{u_i \in \mathcal{V}} \mathcal{S}_i \uparrow \Rightarrow \mathcal{F}_\mathcal{G} \uparrow$.
(2) Keeping $\mathcal{S}_{i,j}$, $\mathcal{S}_i$, and $\rho_{edge}(\mathcal{G})$ fixed, we have that $|V| \uparrow \Rightarrow \mathcal{F}_\mathcal{G} \uparrow$.
(3) Keeping $\mathcal{S}_{i,j}$, $\mathcal{S}_i$, and $|\mathcal{V}|$ fixed, we have that $\rho_{edge}(\mathcal{G}) \uparrow \Rightarrow \mathcal{F}_\mathcal{G} \uparrow$.

Thus, our subgraph-detection problem can be defined as follows:
Problem 2 (Detecting dense subgraphs). **(1) Input:** *the information sharing graph* $\mathbf{G} = (\mathbf{V}, \mathbf{E})$. **(2) Find:** *multiple subgraphs of* $\mathbf{G}$ *that maximize* $\mathcal{F}$.

5.2 D-Spot (Algorithm 1-3)

In real-world datasets, there are usually numerous fraud groups forming multiple dense subgraphs. Based on the considerations described above, we propose D-Spot (Algorithms 1–3). Compared with other well-known algorithms for finding the densest subgraph [6, 11], D-Spot has two differences:

(1) D-Spot can detect multiple densest subgraphs simultaneously. D-Spot first partitions the graph, and then detects a single densest subgraph in each partition.
(2) D-Spot is faster. First, instead of removing nodes one by one, D-Spot removes a set of nodes at once, reducing the number of iterations. Second, it detects the single densest subgraph

in partition $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, rather than in graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, where $|\mathcal{V}| << |\mathbf{V}|$ and $|\mathcal{E}| << |\mathbf{E}|$.

D-Spot consists of two main steps: (1) Given $\mathbf{G}$, divide $\mathbf{G}$ into multiple partitions (Algorithm 1); (2) In each $\mathcal{G}$, find a single dense subgraph (Algorithms 2 and 3).

Algorithm 1: graph partitioning. Let $\hat{\mathcal{G}}$ be a dense subgraph formed by a fraud group that we are about to detect. In $\mathbf{G}$, there are usually multiple $\hat{\mathcal{G}}$s, where each $\hat{\mathcal{G}}$ should be independent or connected with others by small values of $\mathcal{S}_{i,j}$. Thus we let $\mathcal{G}$s be the connected components of $\mathbf{G}$ as partitions (line 6). For each $\mathcal{G} \in \mathcal{G}$s, we run Algorithms 2 and 3 (lines 7–9) to find $\hat{\mathcal{G}}$. Finally, Algorithm 1 returns multiple dense subgraphs $\hat{\mathcal{G}}$s (line 10). Note that there is a guarantee that $\hat{\mathcal{G}}$s must contain the $\hat{\mathcal{G}}$ that is at least 1/2 of the optimum of $\mathbf{G}$ in terms of $\mathcal{F}$ (proof in Sec.6.3).

Information pruning (recommended). As mentioned before, Fraud entities usually have surprising similarities that are quantified by $\mathcal{S}_{i,j}$. We want to delete edges with regular weights and thus we provide a threshold for removing edges:

$$\theta = \frac{\sum_{(u_i, u_j) \in \mathbf{E}} \mathcal{S}_{i,j}}{|\mathbf{V}|(|\mathbf{V}| - 1)} \quad (8)$$

It is easy to see that θ (conservative) is the average information of all possible pairs (u_i, u_j). Thus, we iterate through all edges in $\mathbf{G}$, and remove those for which $\mathcal{S}_{i,j} < \theta$ (lines 3–5). In all experiments of this paper, we used θ and get the expected conclusion that the performance of D-Spot using θ is hardly different from no pruning but using θ is able to significantly decrease the running cost of D-Spot.

Algorithm 1 *find multiple dense subgraphs* in $\mathbf{G}$

Require: $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, θ (Eq.8), $w()$ (Eq. 9)
Ensure: $\hat{\mathcal{G}}$s
1: $\hat{\mathcal{G}}$s $\leftarrow \emptyset$
2: **if** needed **then**
3:     **for** each $\mathcal{S}_{i,j} \in \mathbf{E}$ **do**
4:         **if** $\mathcal{S}_{i,j} < \theta$ **then**
5:             remove $\mathcal{S}_{i,j}$
6: $\mathcal{G}$s $\leftarrow$ connected components of $\mathbf{G}$
7: **for** each $\mathcal{G} \in \mathcal{G}$s **do**
8:     $\hat{\mathcal{G}} \leftarrow$ *find a dense subgraph* $(\mathcal{G}, w())$
9:     $\hat{\mathcal{G}}$s $\leftarrow \hat{\mathcal{G}}$s $\cup \{\hat{\mathcal{G}}\}$
10: **return** $\hat{\mathcal{G}}$s

Algorithms 2 and 3: find a dense subgraph. Initially, let $\mathcal{V}_c$ be a copy of $\mathcal{V}$. In each iteration (lines 5–14), we delete *a set of nodes* (R, line 6) from $\mathcal{V}_c$ until $\mathcal{V}_c$ is empty. Of all the $\mathcal{V}_c$ constructed during the execution of the algorithm, that maximizing $\mathcal{F}$ ($\hat{R}$, line 15) is returned as the output of the algorithm.

Given a subgraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, for some $u_i \in \mathcal{V}$, we define $w(u_i, \mathcal{G})$ as

$$w(u_i, \mathcal{G}) = \sum_{(u_j \in \mathcal{V}) \wedge ((u_i, u_j) \in \mathcal{E})} \mathcal{S}_{i,j} + \mathcal{S}_i. \quad (9)$$

Lines 1–4 initialize the parameters used in the algorithm. $Dict2$ records the w value of each node. $Dict1$ records the order in which

---

Algorithm 2 *find a dense subgraph*

Require: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $w()$(Eq. 9)
Ensure: $\hat{\mathcal{G}}$
1: $\mathcal{V}_c \leftarrow copy(\mathcal{V})$
2: $\mathcal{S}_{sum} \leftarrow \sum_{(u_i, u_j) \in \mathcal{E}} \mathcal{S}_{i,j} + \sum_{u_i \in \mathcal{V}} \mathcal{S}_i$
3: $\forall u \in \mathcal{V}, Dict1[u] \leftarrow 0, Dict2[u] = w(u, \mathcal{G})$
4: $index \leftarrow 0, \mathcal{F}^{max} \leftarrow \frac{\mathcal{S}_{sum}}{|\mathcal{V}_c|}, top \leftarrow 0$
5: **while** $\mathcal{V}_c \neq \emptyset$ **do**
6:     $R \leftarrow \{u \in \mathcal{V}_c : Dict2[u] \leq \frac{2\sum_{(u_i, u_j) \in \mathcal{E}} \mathcal{S}_{i,j} + \sum_{u_i \in \mathcal{V}_c} \mathcal{S}_i}{|\mathcal{V}_c|}\}$ (Eq. 10)
7:     sort R in increasing order of $Dict2[u]$
8:     **for** each $u \in R$ **do**
9:         $\mathcal{V}_c \leftarrow \mathcal{V}_c - u, \mathcal{S}_{sum} \leftarrow \mathcal{S}_{sum} - Dict2[u]$
10:         $index \leftarrow index + 1, Dict1[u] \leftarrow index$
11:         $\mathcal{F} = \frac{\mathcal{S}_{sum}}{|\mathcal{V}_c|}$
12:         **if** $\mathcal{F} > \mathcal{F}^{max}$ **then**
13:             $\mathcal{F}^{max} \leftarrow \mathcal{F}, top \leftarrow index$
14:         $Dict2 \leftarrow$ *update edges* $(u, \mathcal{V}_c, Dict2, \mathcal{G})$
15: $\hat{R} \leftarrow \{u \in \mathcal{V} : Dict1[u] > top\}$
16: **return** $\hat{\mathcal{G}}$ (the subgraph induced by $\hat{R}$)

Algorithm 3 *update edges*

Require: $u_i, \mathcal{V}_c, Dict2, \mathcal{G} = (\mathcal{V}, \mathcal{E})$
Ensure: $Dict2$
1: **for** each $u_j \in \mathcal{V}_c$ **do**
2:     **if** $(u_i, u_j) \in \mathcal{E}$ **then**
3:         $Dict2[u_j] \leftarrow Dict2[u_j] - \mathcal{S}_{i,j}$
4:         remove (u_i, u_j) from $\mathcal{E}$
5: **return** $Dict2$

---

the nodes are deleted (line 10), which allows us to determine the value of $\hat{R}$ that maximizes $\mathcal{F}$. Line 6 determines which R are deleted in each iteration. R is confirmed by $\{u \in \mathcal{V} : w(u, \mathcal{G}) \leq \overline{w}\}$ (line 6), where the average $\overline{w}$ is given by:

$$\begin{aligned}\overline{w} &= \frac{\sum_{u \in \mathcal{V}} w(u, \mathcal{G})}{|\mathcal{V}|} \\ &= \frac{2\sum_{(u_i, u_j) \in \mathcal{E}} \mathcal{S}_{i,j} + \sum_{u_i \in \mathcal{V}} \mathcal{S}_i}{|\mathcal{V}|} \leq 2\mathcal{F}_{\mathcal{G}},\end{aligned} \quad (10)$$

because each edge $\mathcal{S}_{i,j}$ is counted twice in $\sum_{u \in \mathcal{V}} w(u, \mathcal{G})$. In lines 7–14, the nodes in R are removed from $\mathcal{V}_c$ in each iteration (In contrast, [11] recomputes all nodes and finds those with the minimal w after deleting a node). As removing a subset of R may result in a higher value of $\mathcal{F}$, D-Spot records each change of $\mathcal{F}$, as if the nodes were removed one by one (lines 8–14). Algorithm 3 describes how the edges are updated after a node is removed, requiring a total of $|\mathcal{E}|$ updates. Finally, Algorithm 2 returns the subgraph $\hat{\mathcal{G}}$ induced by $\hat{R}$, the set of nodes achieving $\mathcal{F}^{max}$, according to top and $Dict1$ (line 15).

Summary. As R contains at least one node, the worst-case time complexity of Algorithm 2 is $O(|\mathcal{V}|^2 + |\mathcal{E}|)$. In practice, the worst case is too pessimistic. In line 6, R usually contains plenty of nodes, significantly reducing the number of scans of $\mathcal{V}_c$ (see Section 7.2).

6 ANALYSIS

6.1 Complexity.

In the graph initialization stage, it takes $O(K|\mathbf{R}| + |\mathbf{E}|)$ time to build **G** based on the optimization in Section 4.2. In D-Spot, the cost of partitioning **G** is $O(|\mathbf{E}|)$, and detecting a dense block in a partition $\mathcal{G}$ requires $O(|\mathcal{E}| + |\mathcal{V}|^2)$ operations, where $|\mathcal{E}| << |\mathbf{E}|$, $|\mathcal{V}| << |\mathbf{V}|$. Thus, the complexity of ISG+D-Spot is linear with respect to $|\mathbf{E}|$.

In the worst case, admittedly, $|\mathbf{E}| = |\mathbf{V}|^2$ when there is some dimension A_k in which $|\mathbf{R}_k| = 1$. However, that is too pessimistic. In the target fraud attacks, fraud groups typically exhibit strong value sharing while legit entities should not. Hence, we expect **G** to be sparse because the u_i only have positive edges with a small subset of **V**. We constructed a version of **G** using several real-world datasets (see Fig. 3), and the edge densities were all less than 0.06.

6.2 Effectiveness of ISG+D-Spot

THEOREM 6. *(Spotting the Hidden-Densest Block). Given a dense block $\mathcal{B}(\mathcal{A}_1, ..., \mathcal{A}_N, \mathcal{X})$ in which the target dimension $U = A_N$ and $\mathcal{V}$ denotes the set of distinct values of $\mathcal{A}_N$, a shared value a exists in $\mathcal{B}$ such that, $\forall u \in \mathcal{V}, \exists t \in \mathcal{B}$ satisfying $(t[U] = u) \land (t[A_k] = a)$. Then, $\mathcal{B}$ must form a dense subgraph $\mathcal{G}$ in **G**.*

PROOF. Using the optimization algorithm in Section 4.2, we build **G** by scanning all values in **R** once. Hence, the block $\mathcal{B}(a, A_k)$ must be found. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the subgraph induced by $\mathcal{V}$ in **G**. Then, $\forall (u_i, u_j) \in \mathcal{E}$, the edge $S_{i,j} \geq \mathcal{I}_{i,j}^k(a)$. Hence, $\rho_{edge}(\mathcal{G}) = 1.0$ and $\mathcal{F}_\mathcal{G} = \frac{\sum_{\forall(u_i,u_j)\in\mathcal{E}} S_{i,j} + \sum_{u_i \in \mathcal{V}} S_i}{|\mathcal{V}|} \geq \frac{|\mathcal{V}|(|\mathcal{V}|-1)\mathcal{I}_{i,j}^k(a)}{|\mathcal{V}|} = (|\mathcal{V}| - 1)\mathcal{I}_{i,j}^k(a)$. □

Observation. (Effectiveness of ISG+D-Spot) Consider a hidden-densest block $\mathcal{B}(\mathcal{A}_1, ..., \mathcal{A}_{N-1}, \mathcal{A}_N, \mathcal{X})$ of size $|\mathcal{X}| \times ... \times |\mathcal{X}| \times 1$ and $|\mathcal{B}| = |\mathcal{X}|$, i.e., $\mathcal{B}$ is the densest on A_N by sharing the value a. Then, assuming the target dimension $U = A_1$ and the fraudulent entities $\mathcal{V}$ are distinct values of $\mathcal{A}_1$, ISG+D-Spot captures $\mathcal{V}$ more accurately than other algorithms based on tensors (denoted as Tensor+Other Algorithms).

PROOF. Let us consider a non-dense block $\hat{\mathcal{B}}(\hat{\mathcal{A}}_1, ..., \hat{\mathcal{A}}_N, \mathcal{X})$ of size $|\mathcal{X}| \times ... \times |\mathcal{X}|$, $|\hat{\mathcal{B}}| = |\mathcal{X}|$, and let $\hat{\mathcal{V}}$ denote the distinct values of $\hat{\mathcal{A}}_1$. Denoting legitimate entities as $\hat{\mathcal{V}}$ and fraudulent entities as $\mathcal{V}$, we now discuss the difference between ISG+D-Spot and Tensor+Other Algorithms.

[*Working on the tensor*]. On **R**, $\hat{\mathcal{B}}$ is not dense and thus $\rho(\hat{\mathcal{B}}, [N]) = 1$. For $\mathcal{B}$, because $\{|\mathcal{B}_1|, ..., |\mathcal{B}_{N-1}|, |\mathcal{B}_N|\} = \{|\mathcal{X}|, ..., |\mathcal{X}|, 1\}$, we have $\rho(\mathcal{B}, [N]) = \frac{|\mathcal{B}|}{\frac{1}{N}\sum_{n\in[N]} |\mathcal{B}_n|} \approx 1$ for sufficiently large N.

[*Working on ISG*]. On **G**, let $\hat{\mathcal{G}}$ denote the subgraph induced by $\hat{\mathcal{V}}$ and $\mathcal{G}$ denote the subgraph formed by $\mathcal{V}$. We know that $\hat{\mathcal{G}}$, $\mathcal{F}_{\hat{\mathcal{G}}} = 0$, because $\hat{\mathcal{B}}$ does not have any shared values. For $\mathcal{G}$, $\mathcal{F}_\mathcal{G} = (\mathcal{V} - 1)\mathcal{I}_{i,j}^k(a)$ according to Theorem 6.

[*Other Algorithms*]. M-Zoom[30] and D-Cube [32] are known to find blocks that are at least $1/N$ of the optimum in term of ρ on **R** ($\frac{1}{N}$-Approximation guarantee).

[*D-Spot*]. In Section 6.3, we will show that the subgraph detected by D-Spot is at least $1/2$ of the optimum in term of $\mathcal{F}_\mathcal{G}$ on ISG ($\frac{1}{2}$-Approximation guarantee).

In summary, Tensor+Other Algorithms vs. ISG+D-Spot corresponds to:

$$\left(\rho(\hat{\mathcal{B}}, [N]) = 1 \mid \rho(\mathcal{B}, [N]) \approx 1 + (\frac{1}{N}\text{-Approximation}) \right)$$

$$vs. \left(\mathcal{F}_{\hat{\mathcal{G}}} = 0 \mid \mathcal{F}_\mathcal{G} = (\mathcal{V} - 1)\mathcal{I}_{i,j}^k(a) + (\frac{1}{2}\text{-Approximation}) \right)$$

Therefore, ISG+D-Spot catches fraudulent entities within hidden-densest blocks more accurately than Tensor+Other Algorithms. □

From the observation, ISG+D-Spot can effectively detect hidden-densest blocks. Similarly, when $\mathcal{B}$ becomes denser, the $\mathcal{G}$ formed by $\mathcal{B}$ will also be much denser, and thus ISG+D-Spot will be more accurate in detecting the densest block.

6.3 Accuracy Guarantee of D-Spot

For brevity, we use $[\mathcal{V}]$ to denote a subgraph induced by the set of nodes $\mathcal{V}$.

THEOREM 7. *(Algorithm 1 Guarantee). Given $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, let $\mathcal{G}s = \{\mathcal{G}_1, ..., \mathcal{G}_n\}$ denote the connected components of **G**. Let $\mathcal{F}_i^{opt}$ denote the optimal $\mathcal{F}$ on $\mathcal{G}_i$, i.e., $\nexists \mathcal{G}' \subseteq \mathcal{G}_i$ satisfying $\mathcal{F}_{\mathcal{G}'} > \mathcal{F}_i^{opt}$. Then, if $\mathcal{F}_n^{opt}$ is the maximal value of $\{\mathcal{F}_1^{opt}, ..., \mathcal{F}_n^{opt}\}$, $\mathcal{F}_n^{opt}$ must be the optimum in terms of $\mathcal{F}$ on **G**.*

PROOF. Given any two sets of nodes $\mathcal{V}_1$ and $\mathcal{V}_2$ and assuming there are no edges connecting $\mathcal{V}_1$ and $\mathcal{V}_2$, we assume that $\mathcal{F}_{[\mathcal{V}_1]} > \mathcal{F}_{[\mathcal{V}_2]} \Rightarrow \frac{c_1}{|\mathcal{V}_1|} > \frac{c_2}{|\mathcal{V}_2|} \Rightarrow c_1|\mathcal{V}_2| > c_2|\mathcal{V}_1|$. Then,

$$\mathcal{F}_{[\mathcal{V}_1]} - \mathcal{F}_{[\mathcal{V}_1 \cup \mathcal{V}_2]} \Rightarrow \frac{c_1}{|\mathcal{V}_1|} - \frac{c_1 + c_2}{|\mathcal{V}_1| + |\mathcal{V}_2|}$$

$$\Rightarrow \frac{c_1|\mathcal{V}_2| - c_2|\mathcal{V}_1|}{|\mathcal{V}_1|(|\mathcal{V}_1| + |\mathcal{V}_2|)} > \frac{c_2|\mathcal{V}_1| - c_2|\mathcal{V}_1|}{|\mathcal{V}_1|(|\mathcal{V}_1| + |\mathcal{V}_2|)} = 0.$$

Thus, for any $\mathcal{V}_1$ and $\mathcal{V}_2$ that are not connected by any edges, it follows that $\mathcal{F}_{[\mathcal{V}_1 \cup \mathcal{V}_2]} \leq \max(\mathcal{F}_{[\mathcal{V}_1]}, \mathcal{F}_{[\mathcal{V}_2]})$ (Conclusion 1).

In $\mathcal{G}_n = (\mathcal{V}_n, \mathcal{E}_n)$, we use $\hat{\mathcal{V}}$ to denote the set of nodes satisfying $\hat{\mathcal{V}} \subseteq \mathcal{V}_n$ and $\mathcal{F}_{[\hat{\mathcal{V}}]} = \mathcal{F}_n^{opt}$. Let $\mathcal{V}'$ be a set of nodes satisfying $\mathcal{V}' \subset \mathbf{V}$ and $\mathcal{V}' \cap \hat{\mathcal{V}} = \emptyset$. Now, let us consider two conditions.

First, if $\mathcal{V}' \subset \mathcal{V}_n$, then $\mathcal{F}_{[\mathcal{V}']} \leq \mathcal{F}_{[\hat{\mathcal{V}}]}$ and $\mathcal{F}_{[\mathcal{V}' \cup \hat{\mathcal{V}}]} \leq \mathcal{F}_{[\hat{\mathcal{V}}]}$ because $\mathcal{F}_{[\hat{\mathcal{V}}]}$ is the optimum on $\mathcal{G}_n$.

Second, if $\mathcal{V}' \cap \mathcal{V}_n = \emptyset$, then $\mathcal{F}_{[\mathcal{V}']} \leq \mathcal{F}_{[\hat{\mathcal{V}}]}$ and $\mathcal{F}_{[\mathcal{V}' \cup \hat{\mathcal{V}}]} \leq \mathcal{F}_{[\hat{\mathcal{V}}]}$ by Conclusion 1 and because $\mathcal{F}_{[\hat{\mathcal{V}}]}$ is the maximum of $\{\mathcal{F}_1^{opt}, ..., \mathcal{F}_n^{opt}\}$.

If $\mathcal{V}' \cap \mathcal{V}_n \neq \emptyset$, then $\mathcal{V}'$ can be divided into two parts conforming with the two conditions stated above.

Therefore, $\nexists \mathcal{V}' \subset \mathbf{V}$ satisfies $\mathcal{F}_{[\mathcal{V}']} > \mathcal{F}_{[\hat{\mathcal{V}}]}$ or $\mathcal{F}_{[\mathcal{V}' \cup \hat{\mathcal{V}}]} > \mathcal{F}_{[\hat{\mathcal{V}}]}$. We can conclude that $\mathcal{F}_n^{opt} = \mathcal{F}_{[\hat{\mathcal{V}}]}$ must be the optimum in terms of $\mathcal{F}$ on **G**. □

THEOREM 8. *(Algorithm 2 Guarantee). Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, let Q^* be a subset of nodes maximizing $\mathcal{F}_{[Q^*]}$ in $\mathcal{G}$. Let $[Q]$ be the subgraph returned by Algorithm 2 with $\mathcal{F}_{[Q]}$. Then, $\mathcal{F}_{[Q]} \geq \frac{1}{2}\mathcal{F}_{[Q^*]}$.*

PROOF. Consider the optimal set Q^*. We know that, $\forall u \in Q^*$, $w(u, [Q^*]) \geq \mathcal{F}_{[Q^*]}$, because if we remove a node u for which

$w(u, [Q^*]) < \mathcal{F}_{[Q^*]}$,

$$\mathcal{F}' = \frac{|Q^*|\mathcal{F}_{[Q^*]} - w(u, Q^*)}{|Q^*| - 1} > \frac{|Q^*|\mathcal{F}_{Q^*} - \mathcal{F}_{Q^*}}{|Q^*| - 1} = \mathcal{F}_{[Q^*]},$$

which contradicts the definition of Q^*.

Denote the first node that Algorithm 2 removes from Q^* as u_i, $u_i \in R$, and denote the node set before Algorithm 2 starts removing R as Q'. Because $Q^* \subseteq Q'$, we have $w(u_i, [Q^*]) \le w(u_i, [Q'])$. According to Algorithm 2 (line 6), $w(u_i, [Q']) \le 2\mathcal{F}_{[Q']}$ (Eq. 10). Additionally, Algorithm 2 returns the best solution when deleting nodes one by one, and so $\mathcal{F}_{[Q]} \ge \mathcal{F}_{[Q']}$. We conclude that

$$\mathcal{F}_{[Q]} \ge \mathcal{F}_{[Q']} \ge \frac{w(u_i, [Q'])}{2} \ge \frac{w(u_i, [Q^*])}{2} \ge \frac{\mathcal{F}_{Q^*}}{2}.$$

□

In summary, let $\{\mathcal{G}_1, ..., \mathcal{G}_n\}$ be the subgraphs returned by D-Spot, and $\{\mathcal{F}_{\mathcal{G}_1}, ..., \mathcal{F}_{\mathcal{G}_n}\}$ be the corresponding scores. Then, based on Theorems 7 and 8, $\mathcal{F}^{max} = max(\mathcal{F}_{\mathcal{G}_1}, ..., \mathcal{F}_{\mathcal{G}_n})$ is at least 1/2 of the optimum in terms of $\mathcal{F}$ on **G** (1/2-Approximation guarantee).

7 EVALUATION

A series of evaluation experiments were conducted under the following conditions:

Implementation. We implemented ISG+D-Spot in Python, and conducted all experiments on a server with two 2.20 GHz Intel(R) CPUs and 64 GB memory.

Baselines. We selected several state-of-the-art dense-block detection methods (M-Zoom [30], M-Biz [31], and D-Cube [32]) as the baselines (using open source code). To obtain optimal performance, we run three different density metrics from [32] for each baseline: ρ **(ari)**, Geometric Average Mass **(geo)**, and Suspiciousness **(sus)**.

Suspiciousness Score Setting. For the baselines, we considered a detected block $\mathcal{B}(\mathcal{A}_1, ..., \mathcal{A}_N, \mathcal{X})$ and let $\theta = \rho(\mathcal{B}, N)$. For any unique value a within $\mathcal{B}$, we then set the suspiciousness score of a to θ. If a occurred in multiple detected dense blocks, we chose the one with the maximal value of θ. Regarding ISG+D-Spot, given a detected subgraph $\hat{\mathcal{G}} = (\hat{\mathcal{V}}, \hat{\mathcal{E}})$, for a unique value $a \in \hat{\mathcal{V}}$, we set the suspiciousness score of a to $w(a, \hat{\mathcal{G}})$ (Eq. 9). Finally, we evaluated the ranking of the suspiciousness scores of unique values using the standard area under the ROC curve (AUC) metric.

7.1 Datasets

Table 3 summarizes the datasets used in the experiments.

Synthetic is a series of datasets we synthesized using the same method as in [12]. First, we generated random seven-dimensional relations $\mathbf{R}(A_1, ..., A_7, X)$, in which $|\mathbf{R}| = 10000$ and the size of **R** is $1000 \times 500 \times ... \times 500$. In **R**, we assume that A_1 corresponds to users and the other six dimensions are features. To specifically check the detection performance of the hidden-densest block using each method, we injected a dense block $\mathcal{B}(\mathcal{A}_1, ..., \mathcal{A}_7, \mathcal{X})$ into **R** five separate times, with each injection assigned a different configuration to generate five datasets. For $\mathcal{B}$, $|\mathcal{B}_1| = 50$ and $|\mathcal{B}| = 500$. We introduce the parameter λ, which denotes the number of dimensions on which $\mathcal{B}$ is the densest. For example, when $\lambda = 1$, the size of $\mathcal{B}$ is $50 \times \mathbf{12} \times 25... \times 25$; when $\lambda = 5$, the size of $\mathcal{B}$ is $50 \times \mathbf{12} \times ... \times \mathbf{12} \times 25$. Obviously, $\rho(\mathcal{B}, 7) > \rho(\mathbf{R}, 7)$ and $\mathcal{B}$ is the hidden-densest block when λ is small. Finally, we labeled the users within $\mathcal{B}$ as "fraud".

Amazon [15]. AmaOffice, AmaBaby, and AmaTools are three collections of reviews about office products, baby-related products, and tool products, respectively, on Amazon. They can be modeled using the relation $\mathbf{R}(user, product, timestamp, X)$. For each entry $t \in \mathbf{R}$, $t = (u, p, t, x)$ indicates a review x that user u reviewed product p at time t. According to the specific cases of fraud discovered by previous studies [12, 38], fraudulent groups usually exhibit suspicious synchronized behavior in social networks. For instance, a large group of users may surprisingly review the same group of products over a short period. Thus, we use a similar method as in [12, 30, 32, 38]. We use a dense block $\mathcal{B}$ to represent the synchronized behavior, where $\mathcal{B}(user, product, timestamp, X)$ has a size of $200 \times 30 \times 1$. In total, we injected four such blocks $\mathcal{B}$ with a mass randomly selected from [1000, 2000]. The users in the injected blocks were labeled as "malicious."

Yelp [25]. The YelpChi, YelpNYU, and YelpZip datasets [21, 25] contain restaurant reviews submitted to Yelp. They can be represented by the relation $\mathbf{R}(user, restaurant, date, X)$, where each entry $t = (u, r, d, x)$ denotes a review x by user u of restaurant r on date d. Note that all three datasets include labels indicating whether or not each review is fake. The detection of malicious reviews or users is studied in [25] using text information. In these datasets, we focus on detecting fraudulent restaurants that purchase fake reviews using the three-dimensional features. Intuitively, the more fake reviews a restaurant has, the more suspicious it is. As some legitimate users have the potential of reviewing fraudulent restaurants, we label a restaurant as "fraudulent" if it has received more than 40 fake reviews.

DARPA [19] was collected by the Cyber Systems and Technology Group in 1998 regarding network attacks in TCP dumps. The data has the form $\mathbf{R}(sourceIP, targetIP, timestamp, X)$. Each entry $t = (IP_1, IP_2, t)$ represents a connection made from IP_1 to IP_2 at time t. The dataset includes labels indicating whether or not each connection is malicious. In practice, the punishment for malicious connections is to block the corresponding IP address. Thus, we compared the detection performance of suspicious IP addresses. We labeled an IP address as suspicious if it was involved in a malicious connection.

AirForce [1] was used for the KDD Cup 1999, and has also been considered in [30, 32]. This dataset includes a wide variety of simulated intrusions in a military network environment. However, it does not contain any specific IP addresses. According to the cardinality of each dimension, we chose the top-2 features and built the relation $\mathbf{R}(src\ bytes, dst\ bytes, connections, X)$, where src bytes denotes the number of data bytes sent from source to destination and dst bytes denotes the number of data bytes sent from destination to source. The target dimension U was set to be *connections*. Note that this dataset includes labels indicating whether or not each connection is malicious.

7.2 Speed and Accuracy of D-Spot

First, we measured the speed and accuracy with which D-Spot detected dense subgraphs in real-world graphs. We compared the

Table 3: Multi-dimensional datasets used in our experiments

| | Synthetic | TCP Dumps | | Review Data | | | | | |
		DARPA	AirForce	YelpChi	YelpNYU	YelpZip	AmaOffice	AmaBaby	AmaTools
Entries (Mass)	10K	4.6M	30K	67K	359K	1.14M	53K	160K	134K
Dimensions	7	3	3	3	3	3	3	3	3

performance of D-Spot with that of another dense-subgraph detection algorithm, Fraudar [11], the extension of [6], which maximizes the density metric by greedily deleting nodes **one by one**. We used the three Amazon datasets and applied D-Spot and Fraudar to the same bipartite graph built on the first two dimensions, *users* and *products*, where each edge in the graph represents an entry. We measured the wall-clock time (average over three runs) required to detect the top-4 subgraphs. Figure 2 illustrates the runtime and performance of the two algorithms.
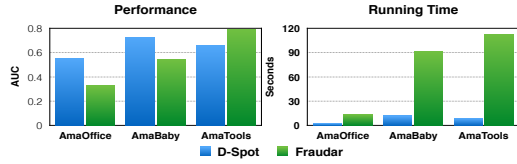

Figure 2: Comparison of D-Spot and Fraudar using the Amazon datasets.

D-Spot provides the best trade-off between speed and accuracy. Specifically, D-Spot is up to 11× faster than Fraudar. This supports our claim in Section 5.2 that the worst-case time complexity of D-Spot ($O(|\mathcal{V}|^2 + |\mathcal{E}|)$) is too pessimistic.

7.3 Effectiveness of ISG+D-Spot

This section illustrates the effectiveness of ISG+D-Spot for detecting fraudulent entities on multi-dimensional tensors. ISG+D-Spot exhibits extraordinary performance compared with the baseline methods (Fraudar is not in the baselines as it only works on the bipartite graph).

Synthetic. Table 4 presents the detection performance of each method for the hidden-densest block. We assume that the injected block $\mathcal{B}$ is the hidden-densest block when $\lambda \leq 3$. In detail, ISG+D-Spot achieves extraordinary performance even when $\lambda = 1$, because each instance of value sharing in $\mathcal{B}$ is accurately captured by ISG and D-Spot, providing a higher accuracy guarantee than the baselines (Theorems 2 and 3). When $\lambda > 3$, the performance of each method improves because the density of $\mathcal{B}$ increases as λ increases.

Amazon. Table 5 presents the results for catching suspicious users by detecting the top-4 dense blocks on the Amazon datasets. ISG+D-Spot detects synchronized behavior accurately. The typical attack scenario involves a mass of fraudulent users creating massive numbers of fake reviews for a comparatively small group of products over a short period. This behavior is represented by the injected blocks. ISG+D-Spot exhibits robust and near-perfect performance. However, the other baselines produce worse performance on the AmaOffice and AmaBaby datasets, even with the multiple supported metrics.

Table 4: Performance (AUC) on the Synthetic datasets

	$\lambda = 1$	$\lambda = 2$	$\lambda = 3$	$\lambda = 4$	$\lambda = 5$
M-Zoom (ari)	0.5005	0.5005	0.5000	0.5489	0.6567
M-Zoom (geo)	0.5005	0.5005	0.5000	0.6789	0.7543
M-Zoom (sus)	0.7404	0.7715	0.8238	0.9685	0.9767
M-Biz (ari)	0.5005	0.5005	0.5005	0.5005	0.6834
M-Biz (geo)	0.5005	0.5005	0.5005	0.6235	0.7230
M-Biz (sus)	0.6916	0.7638	0.8067	0.9844	**0.9948**
D-Cube (ari)	0.5005	0.5005	0.5005	0.5670	0.6432
D-Cube (geo)	0.5005	0.5005	0.5340	0.6876	0.6542
D-Cube (sus)	**0.8279**	**0.8712**	**0.9148**	**0.9909**	0.9725
ISG+D-Spot	**0.9843**	**0.9957**	**0.9949**	**1.0000**	**1.0000**

Table 5: Performance (AUC) on the Amazon datasets

	AmaOffice	AmaBaby	AmaTools
M-Zoom (ari)	0.6795	0.5894	0.8689
M-Zoom (geo)	0.8049	0.8049	1.0000
M-Zoom (sus)	0.7553	0.6944	0.6503
M-Biz (ari)	0.6328	0.5461	0.8384
M-Biz (geo)	0.8049	**0.8339**	1.0000
M-Biz (sus)	0.7478	0.6944	0.6503
D-Cube (ari)	0.7127	0.5956	0.6250
D-Cube (geo)	**0.8115**	0.7561	1.0000
D-Cube (sus)	0.7412	0.6190	0.5907
ISG+D-Spot	**0.8358**	**0.9995**	**1.0000**

Yelp. Table 6 reports the (highest) accuracy with which collusive restaurants were detected by each method. In summary, using ISG+D-Spot results in the highest accuracy across all three datasets, because D-Spot applies a higher theoretical bound to the ISG.

Table 6: Performance (AUC) on the Yelp datasets

	YelpChi	YelpNYU	YelpZip
M-Zoom (ari)	0.9174	0.6669	0.8859
M-Zoom (geo)	0.9752	0.8826	0.9274
M-Zoom (sus)	0.9831	**0.9451**	0.9426
M-Biz (ari)	0.9174	0.6669	0.8863
M-Biz (geo)	0.9757	0.8826	0.9271
M-Biz (sus)	**0.9831**	0.9345	**0.9403**
D-Cube (ari)	0.5000	0.5000	0.9033
D-Cube (geo)	0.9793	0.9223	0.9376
D-Cube (sus)	0.9810	0.9007	0.9365
ISG+D-Spot	**0.9875**	**0.9546**	**0.9529**

DARPA. Table 7 lists the accuracy of each method for detecting the source IP and the target IP. ISG+D-Spot assigns each IP address

a specific suspiciousness score. We chose a detected IP with the highest score and found that the IP participated in more than 1M malicious connections. The top ten suspicious IPs were all involved in more than 10k malicious connections. Thus, using ISG+D-Spot would enable us to crack down on these malicious IP addresses in the real world.

Table 7: Performance (AUC) on the DARPA dataset

Dataset	DARPA	
	U = source IP	U = target IP
M-Zoom (ari)	0.5649	0.5584
M-Zoom (geo)	0.7086	**0.5714**
M-Zoom (sus)	0.6989	0.3878
M-Biz (ari)	0.5649	0.5584
M-Biz (geo)	**0.7502**	0.5679
M-Biz (sus)	0.6989	0.3878
D-Cube (ari)	0.3728	0.5323
D-Cube (geo)	0.4083	0.3926
D-Cube (sus)	0.4002	0.3720
ISG+D-Spot	**0.7561**	**0.8181**

AirForce. As this dataset does not contain IP addresses, we set the target dimension $U = connections$. We randomly sample 30k connections from the dataset [1] three times. Table 8 lists the accuracy of each method on samples 1–3. Malicious connections form dense blocks on the two-dimensional features. The results demonstrate that ISG+D-Spot effectively detected the densest blocks.

Table 8: Performance (AUC) on the AirForce dataset

	Sample 1	Sample 2	Sample 3
M-Zoom (ari)	0.8696	0.8675	0.8644
M-Zoom (geo)	0.9693	0.9693	0.9738
M-Zoom (sus)	0.9684	0.9683	0.9726
M-Biz (ari)	0.9038	0.8848	0.8885
M-Biz (geo)	0.9694	0.9693	0.9741
M-Biz (sus)	0.9684	0.9683	0.9726
D-Cube (ari)	**0.9824**	**0.9823**	0.9851
D-Cube (geo)	0.9695	0.9691	**0.9862**
D-Cube (sus)	0.9697	0.9692	0.9737
ISG+D-Spot	**0.9835**	**0.9824**	**0.9877**

7.4 Scalability

As mentioned in Sec.6.1, the ISG **G**s built using real-world tensors are typically sparse, as value sharing should only conspicuously appear in fraudulent entities. We implemented **G** on the three Amazon datasets (details in Figure 3). The edge densities of **G** are quite low (lower than 0.06) across all datasets, which indicates that the worst-case time complexity discussed in Section 6.1 rarely occurs. Figure 3 reports the runtime of ISG+D-Spot on the three Amazon datasets, where the number of edges was varied by subsampling entries in the dataset. In practice, |**E**| increases near-linearly with the mass of the dataset. Additionally, because the time complexity of ISG+D-Spot is linear with respect to |**E**|, ISG+D-Spot exhibits near-linear scaling with the mass of the dataset. Figure 3 demonstrates our conclusion.
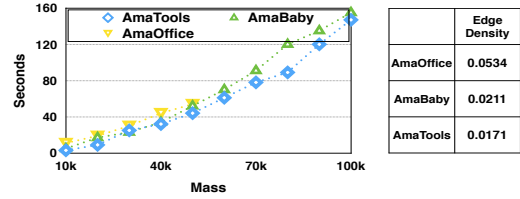

Figure 3: ISG+D-Spot runs in near-linear time with respect to the mass of the dataset.

7.5 Feature Prioritization

This section is to demonstrate that ISG+D-Spot is more robust to resist noisy features than existing approaches. ISG automatically weighs each feature and continuously accumulates value sharing by one scan of the tensor, and D-Spot amounts to finds entities with the maximum of value sharing. We conducted the following experiment to demonstrate our conclusion.

Registration is a dataset derived from an e-commerce company, in which each record contains two crucial features, IP subnet and phone prefix, and three noisy features, IP city, phone city, and timestamp. The dataset also includes labels showing whether or not the account is a "zombie" account. Thus, it can be formulated as $R(accounts, IP, phone, IP\ city, phone\ city, timestamp, X)$. To compare the detection performance of malicious accounts, we applied each method on various R by successively appending 1–5 features to $R(accounts, X)$.

Table 9: Performance (AUC) on the Registration dataset. 'C' represents 'crucial feature' and 'N' represents 'noisy feature'

	1C	2C	2C+1N	2C+2N	2C+3N
M-Zoom (ari)	0.5000	0.5000	0.5031	0.5000	0.5430
M-Zoom (geo)	0.7676	0.8880	**0.8827**	**0.8744**	**0.8439**
M-Zoom (sus)	0.7466	0.8328	0.4009	0.4878	0.4874
M-Biz (ari)	0.5000	0.5000	0.5000	0.5000	0.5004
M-Biz (geo)	**0.7677**	0.8842	0.8827	0.8744	0.8439
M-Biz (sus)	0.7466	0.8328	0.4009	0.4878	0.4874
D-Cube (ari)	0.7073	0.8189	0.8213	0.8295	0.7987
D-Cube (geo)	0.7073	**0.9201**	0.8586	0.8312	0.7324
D-Cube (sus)	0.7522	0.8956	0.7877	0.7642	0.7080
ISG+D-Spot	**0.7699**	**0.9946**	**0.9935**	**0.9917**	**0.9859**

Table 9 gives the variation of each method with regard to the added noisy features (3– 5 dimensions). As each account only possesses one entry, R is quite sparse. We found that existing methods usually miss small-scale instances of value sharing because their density is close to the legitimate range on R. For example, a 51-member group sharing a single IP subnet was missed by the baseline methods. However, ISG amplifies each instance of value sharing through its information-theoretic and graph features, allowing D-Spot to accurately capture fraudulent entities.

8 CONCLUSIONS AND FUTURE WORK

In this paper, we novelly identified dense-block detection with dense-subgraph mining, by modeling a tensor in ISG. Additionally, we propose a multiple dense-subgraphs detection algorithm that is faster and can be computed in parallel. In future, ISG + D-Spot will be implemented on Apache Spark [39] to support very large tensors.

REFERENCES

[1] 1999. Kdd cup 1999 data. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html..

[2] Melih Abdulhayoglu, Melih Abdulhayoglu, Melih Abdulhayoglu, and Melih Abdulhayoglu. 2017. HinDroid: An Intelligent Android Malware Detection System Based on Structured Heterogeneous Information Network. In *ACM SIGKDD*. 1507–1515.

[3] Leman Akoglu, Rishi Chandy, and Christos Faloutsos. 2013. Opinion Fraud Detection in Online Reviews by Network Effects.. In *ICWSM*. The AAAI Press.

[4] David S Anderson, Chris Fleizach, Stefan Savage, and Geoffrey M Voelker. 2007. Spamscatter: characterizing internet scam hosting infrastructure. In *Usenix Security Symposium on Usenix Security Symposium*. 1132–1141.

[5] Qiang Cao, Xiaowei Yang, Jieqi Yu, and Christopher Palow. 2014. Uncovering Large Groups of Active Malicious Accounts in Online Social Networks. In *ACM CCS*. 477–488.

[6] Moses Charikar. 2000. Greedy approximation algorithms for finding dense components in a graph. In *International Workshop on Approximation Algorithms for Combinatorial Optimization*. Springer, 84–95.

[7] Jie Chen and Yousef Saad. 2012. Dense Subgraph Extraction with Application to Community Detection. *IEEE TKDE* 24, 7 (2012), 1216–1230.

[8] Hector Garcia-Molina and Jan Pedersen. 2004. Combating web spam with trustrank. In *VLDB*. 576–587.

[9] Saptarshi Ghosh, Bimal Viswanath, Farshad Kooti, Naveen Kumar Sharma, Gautam Korlam, Fabricio Benevenuto, Niloy Ganguly, and Krishna Phani Gummadi. 2012. Understanding and combating link farming in the Twitter social network. In *WWW*.

[10] A. V Goldberg. 1984. *Finding a Maximum Density Subgraph*. Technical Report.

[11] 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 *ACM SIGKDD*. 895–904.

[12] 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. *IEEE TKDE* 28, 8 (2016), 2187–2200.

[13] Meng Jiang, Peng Cui, Alex Beutel, Christos Faloutsos, and Shiqiang Yang. 2014. CatchSync: catching synchronized behavior in large directed graphs. In *ACM SIGKDD*. 941–950.

[14] Meng Jiang, Peng Cui, Alex Beutel, Christos Faloutsos, and Shiqiang Yang. 2016. Inferring lockstep behavior from connectivity pattern in large graphs. *Knowledge & Information Systems* 48, 2 (2016), 399–428.

[15] McAuley Julian. [n. d.]. Amazon product data. http://jmcauley.ucsd.edu/data/amazon/.

[16] Samir Khuller and Barna Saha. 2009. On Finding Dense Subgraphs. In *Automata, Languages and Programming, International Colloquium, ICALP 2009, Rhodes, Greece, July 5-12, 2009, Proceedings*. 597–608.

[17] Tamara G. Kolda and Brett W. Bader. 2009. Tensor Decompositions and Applications. *Siam Review* 51, 3 (2009), 455–500.

[18] Victor E. Lee, Ruan Ning, Ruoming Jin, and Charu Aggarwal. 2010. *A Survey of Algorithms for Dense Subgraph Discovery*. 303–336 pages.

[19] R. P. Lippmann, D. J. Fried, I. Graf, J. W. Haines, K. R. Kendall, D. McClung, D. Weber, S. E. Webster, D. Wyschogrod, R. K. Cunningham, and M. A. Zissman. 2000. Evaluating intrusion detection systems: the 1998 DARPA off-line intrusion detection evaluation. In *Proceedings DARPA Information Survivability Conference and Exposition. DISCEX'00*, Vol. 2. 12–26 vol.2. https://doi.org/10.1109/DISCEX. 2000.821506

[20] Koji Maruhashi, Fan Guo, and Christos Faloutsos. 2011. MultiAspectForensics: Pattern Mining on Large-Scale Heterogeneous Networks with Tensor Analysis. In *ASONAM*. 203–210.

[21] Arjun Mukherjee, Vivek Venkataraman, Bing Liu, and Natalie Glance. 2013. What yelp fake review filter might be doing?. In *Proceedings of the 7th International Conference on Weblogs and Social Media, ICWSM 2013*. AAAI press, 409–418.

[22] 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*. 201–210.

[23] Evangelos E. Papalexakis, Christos Faloutsos, and Nicholas D. Sidiropoulos. 2012. ParCube: sparse parallelizable tensor decompositions. In *PKDD*. 521–536.

[24] 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*.

[25] Shebuti Rayana and Leman Akoglu. 2015. Collective Opinion Spam Detection: Bridging Review Networks and metadata. In *ACM SIGKDD*.

[26] Barna Saha, Allison Hoch, Samir Khuller, Louiqa Raschid, and Xiao Ning Zhang. 2010. *Dense Subgraphs with Restrictions and Applications to Gene Annotation Graphs*. Springer Berlin Heidelberg. 456–472 pages.

[27] Neil Shah, Alex Beutel, Brian Gallagher, and Christos Faloutsos. 2014. Spotting Suspicious Link Behavior with fBox: An Adversarial Perspective. In *IEEE ICDM*. 959–964.

[28] C. E Shannon. 1948. A mathematical theory of communication. *Bell Labs Technical Journal* 27, 4 (1948), 379–423.

[29] K. Shin, T. Eliassi-Rad, and C. Faloutsos. 2017. CoreScope: Graph Mining Using k-Core Analysis – Patterns, Anomalies and Algorithms. In *ICDM*, Vol. 00. 469–478.

[30] Kijung Shin, Bryan Hooi, and Christos Faloutsos. 2016. M-Zoom: Fast Dense-Block Detection in Tensors with Quality Guarantees. In *ECML PKDD*. 264–280.

[31] Kijung Shin, Bryan Hooi, and Christo Faloutsos. 2018. Fast, Accurate, and Flexible Algorithms for Dense Subtensor Mining. *ACM Transactions on Knowledge Discovery from Data* 12, 3 (2018), 1–30.

[32] Kijung Shin, Bryan Hooi, Jisu Kim, and Christos Faloutsos. 2017. D-Cube: Dense-Block Detection in Terabyte-Scale Tensors. In *WSDM*. 681–689.

[33] Kijung Shin and U. Kang. 2015. Distributed Methods for High-Dimensional and Large-Scale Tensor Factorization. In *IEEE ICDM*. 989–994.

[34] Ming Yang Su. 2011. Real-time anomaly detection systems for Denial-of-Service attacks by weighted k-nearest-neighbor classifiers. *Expert Systems with Applications* 38, 4 (2011), 3492–3498.

[35] Hua Tang and Zhuolin Cao. 2009. Machine Learning-based Intrusion Detection Algorithms. *Journal of Computational Information Systems* (2009), 1825–1831.

[36] Kurt Thomas, Dmytro Iatskiv, Elie Bursztein, Tadek Pietraszek, Chris Grier, and Damon McCoy. 2014. Dialing back abuse on phone verified accounts. In *Proceedings of the 2014 ACM SIGSAC*. ACM, 465–476.

[37] Yining Wang, Hsiao Yu Tung, Alexander Smola, and Animashree Anandkumar. 2015. Fast and Guaranteed Tensor Decomposition via Sketching. *NIPS*.

[38] Wanhong Xu, Wanhong Xu, Christopher Palow, Christopher Palow, and Christos Faloutsos. 2013. CopyCatch: stopping group attacks by spotting lockstep behavior in social networks. In *WWW*. 119–130.

[39] Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy Mccauley, Michael J. Franklin, Scott Shenker, and Ion Stoica. 2012. Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing. In *Usenix NSDI*. 2–2.