# Hierarchical Dense Pattern Detection in Tensors

WENJIE FENG, Institute of Data Science, National University of Singapore
SHENGHUA LIU and XUEQI CHENG, CAS Key Laboratory of Network Data Science and Technology, Institute of Computing Technology, Chinese Academy of Sciences

Dense subtensor detection gains remarkable success in spotting anomalies and fraudulent behaviors for multi-aspect data (i.e., tensors), like in social media and event streams. Existing methods detect the densest subtensors flatly and separately, with the underlying assumption that those subtensors are exclusive. However, many real-world tensors usually present hierarchical properties, e.g., the core-periphery structure and dynamic communities in networks. It is also unexplored how to fuse the prior knowledge into dense pattern detection to capture the local behavior.

In this article, we propose CatchCore, a novel framework to efficiently find the hierarchical dense subtensors. We first design a unified metric for dense subtensor detection, which can be optimized with gradient-based methods. With the proposed metric, CatchCore detects hierarchical dense subtensors through the hierarchy-wise alternative optimization and finds local dense patterns concerning some items in a query manner. Finally, we utilize the minimum description length principle to measure the quality of detection results and select the optimal hierarchical dense subtensors. Extensive experiments on synthetic and real-world datasets demonstrate that CatchCore outperforms the top competitors in accuracy for detecting dense subtensors and anomaly patterns, like network attacks. Additionally, CatchCore successfully identifies a hierarchical researcher co-authorship group with intense interactions in the DBLP dataset; it can also capture core collaboration and multi-hop relations around some query objects. Meanwhile, CatchCore also scales linearly with all aspects of tensors.

CCS Concepts: • **Information systems** → **Data mining**; • **Computing methodologies** → *Anomaly detection*; • **Theory of computation** → Data structures and algorithms for data management

Additional Key Words and Phrases: Tensor, dense subtensor, hierarchical structure, anomaly detection, pattern query

**ACM Reference format:**
Wenjie Feng, Shenghua Liu, and Xueqi Cheng. 2023. Hierarchical Dense Pattern Detection in Tensors. *ACM Trans. Knowl. Discov. Data.* 17, 6, Article 81 (February 2023), 29 pages.
https://doi.org/10.1145/3577022

## 1 INTRODUCTION

Dense subgraph and subtensor detection have been successfully used in a variety of application domains, such as detecting anomalies or fraudulent patterns (e.g., lockstep behavior, boosting

Authors' addresses: W. Feng (corresponding author), 1 Institute of Data Science, National University of Singapore, innovation 4.0, #04-06, 117602, Singapore; email: wenchiehfeng.us@gmail.com; S. Liu and X. Cheng, 2 CAS Key Laboratory of Network Data Science and Technology, Institute of Computing Technology, Chinese Academy of Sciences, 6 Zhongguancun South Rd, Beijing 100190, China; emails: {liushenghua, cxq}@ict.ac.cn.

1556-4681/2023/02-ART81 $15.00
https://doi.org/10.1145/3577022

ratings, and ill-gotten Page Likes) in social media or online review sites [Beutel et al. 2013; Hooi et al. 2016; Jiang et al. 2015], identifying malicious attacks in network traffic logs or streaming data [Maruhashi et al. 2011; Shin et al. 2016, 2017a], and spotting changing gene-communities in biological networks [Wong et al. 2018], and so on.

Several algorithms detected the densest subtensors or blocks flatly, i.e., detect-remove-redetect one by one, assuming that those subtensors are exclusive and separate [Duong et al. 2020; Jiang et al. 2015; Shin et al. 2016, 2018, 2017b; Yikun et al. 2019]. However, many real-world graphs and tensors usually present hierarchical properties, like the core-peripheral structure in networks [Gallagher et al. 2021; Kostoska et al. 2020], dynamic communities in social media [Chen and Saad 2010; Leskovec et al. 2008], critical disease-spreading conduits [Kitsak et al. 2010], and evolving behaviors [Rossi et al. 2020; Yang et al. 2013]. So, it will be difficult to identify the subtle structures (like a multi-layer core) within the dense block and the relations (e.g., overlapping or inclusion) among different blocks. Meanwhile, other methods for community detection [Chen and Saad 2010; Edler et al. 2017; Yang et al. 2013, 2011] and dense subgraph detection [Hooi et al. 2016; Sariyüce and Pinar 2016; Zhang et al. 2017] only concentrated on the general graph. Existing hierarchical tensor decomposition approaches [Grasedyck 2010; Song et al. 2013] can also not be applied to the dense subtensor detection scenarios.

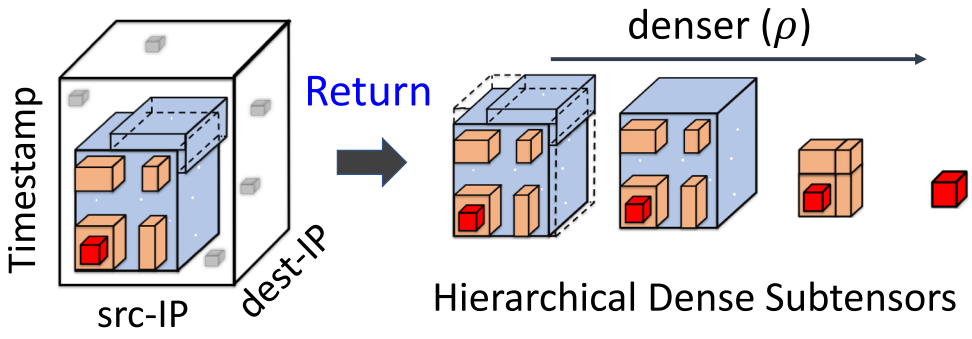
One challenging problem is how to detect the hierarchical dense subtensors efficiently in the multi-aspect data, and Figure 1(a) illustrates an example of the TCP dumps scenario. The network intrusion attacks dynamically changed in interacting intensity at different phases over time and among various hosts (i.e., from `source-IP` to `destination-IP`), leading to a multi-layer and high-density core. Hence, hierarchical dense subtensor detection aids in understanding the process and detecting such anomalies. In addition, although data query and information retrieval have been extensively studied, it remains an intriguing problem to investigate different granularity local behavior patterns of some target objects in tensors when only limited prior knowledge is available, which can be applied to crime detection, behavior recognition, and so on.

Therefore, we propose CatchCore, a novel framework to detect hierarchical dense cores in multi-aspect data (i.e., tensors). We first design a *unified metric* for defining different density measures for dense subtensor detection, where the gradient-based approaches can be utilized to optimize some derived objective functions under this metric. Relying on one type of these objective functions, CatchCore can *identify hierarchical dense cores* via a hierarchy-wise alternative optimization technique with a convergence guarantee. In addition, we present the *query-specific subtensor detection problem* to find the locally dense patterns relating to specific target objects in a query-like manner.

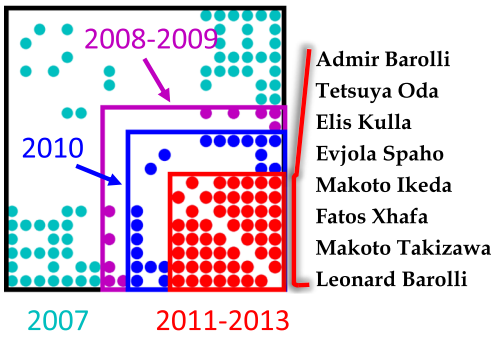
Extensive experiments show that CatchCore outperforms the baselines and discovers various interesting patterns in real-world datasets. CatchCore performs the best at dense block detection without missing detection, even the density of the injected block becomes less dense (see Figure 3). CatchCore finds hierarchical dense subtensors with a higher density than others; it captures a dense co-authorship research group for the DBLP dataset (see Figure 1(c) and (b)) and anomalies in other real-world data. For the local pattern detection, CatchCore can capture core collaboration and multi-hop relationship around some query objects. Moreover, we propose to use the *MDL principle*, which is an information-theoretic criterion, to measure the quality of detection result and help to conduct the parameter/model selection and choose the optimal hierarchical dense subtensors. Finally, our method is *scalable*, with linear runtime and space complexity, in all aspects of the tensor.

In summary, our main contributions are as follows.

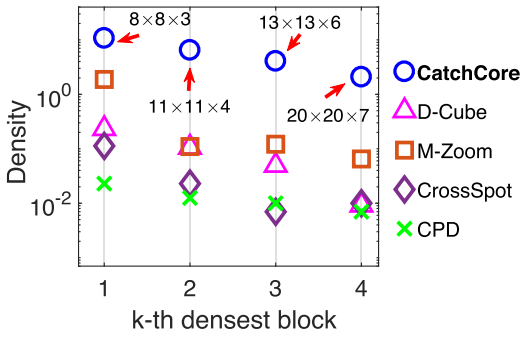
— **Unified metric and algorithm**: We design a unified metric that can be optimized with the gradient-based approaches to detect dense blocks, propose CatchCore for hierarchical




(a) Pictorial depiction of hierarchical dense subtensors.




(b) Hierarchical researcher cooperations in DBLP.




(c) Top 4 densest blocks in DBLP.

Fig. 1. **Illustrative Examples and detected patterns.** (a) Example and workflow of hierarchical dense subtensors detection. (b) shows the detected dense co-authorship researcher group (a multi-layer core) of 20 users in DBLP. The densest block (red) lasts 3 years (2011–2013) containing 8 authors as the list shows, the outer hierarchies (with different colors) include other researchers and exist in various time ranges (text labeled). (c) CatchCore detects dense subtensors with higher density compared with baselines for the top four densest blocks in DBLP. These blocks correspond to a hierarchical group in Figure 1(b).

dense core detection with a theoretical convergence guarantee and MDL-based quality measurement, and present the query-specific dense subtensor detection for some target items.

— **Accuracy**: CatchCore outperforms the state-of-the-art methods in accurately detecting dense blocks and hierarchical dense subtensors in both synthetic and real-world datasets (Figures 3 and 1(c)).

— **Effectiveness**: CatchCore successfully spotted anomalous patterns, including suspicious friend connections, periodic network attacks, and so on, found a researcher co-authorship group with heavy interactions in a hierarchical fashion (Figure 1(b)), and captured some locally dense patterns as core collaboration and multi-hop relations around some query objects.

— **Scalability**: CatchCore is scalable, with linear time and space complexity with all aspects of tensors (Theorems 5.2 and 5.3).

**Reproducibility:** Our open-sourced code and the data we used are available at http://github.com/wenchieh/catchcore.

The rest of the article is organized as follows. In Section 2, we review related work about our topics. Section 3 provides the preliminaries & notions, and Section 4 presents our unified density-metric framework and the problem formulation. In Section 5, we describe our proposed algorithm,

CatchCore, and analyze its complexity. After providing experimental results in Section 6, we conclude in Section 7.

## 2 BACKGROUND AND RELATED WORK

In this section, we present a survey for the problem, algorithms for the dense pattern and hierarchical structure detection in graph and tensor, and the important related application scenario, i.e., anomaly detection.

*Dense Subgraph/Subtensor Detection in Graph and Tensor.* The detection of dense subgraphs has been extensively studied in theory [Charikar 2000; Gibson et al. 2005; Jethava et al. 2013; Lee et al. 2010]. This task is inherently hard, and determining the densest subgraphs and subtensors is primarily an NP-hard problem [Asahiro et al. 2002]. Instead of detecting exactly the densest subgraph via max-flow-based algorithms [Goldberg 1984], greedy approximation methods give a 1/2-approximation to the optimal solutions [Chakrabarti et al. 2004; Khuller and Saha 2009]. These related algorithms were applied to mine formal concepts [Ignatov et al. 2013], detect community structure [Balalau et al. 2015; Chen and Saad 2010; Zhang et al. 2017] and anomaly [Akoglu et al. 2015; Hooi et al. 2016], and also extended to multi-aspect data (tensor) [Duong et al. 2020; Shin et al. 2016, 2018, 2017b; Yikun et al. 2019]. Another approach is CrossSpot [Jiang et al. 2015], which spotted retweet boosting in Weibo; it could find suspicious dense blocks by greedily adjusting the randomly chosen seed until the local optimum is attained. Tensor decomposition [Kolda and Bader 2009], such as HOSVD and **CP Decomposition** (**CPD**), is also used to find dense subtensors; MAF [Maruhashi et al. 2011], which is based on CPD, spots port-scanning activities, forming dense subtensor patterns in the network traffic logs. M-Zoom [Shin et al. 2016], M-Biz [Shin et al. 2018], and D-Cube [Shin et al. 2017b] utilized a greedy approximation approach to detect the estimated dense-subtensors for large-scale tensors in the in- and external-memory, distributed settings, and the density of the detected subtensors is guaranteed to be $1/N$ optimal for *arithmetic average mass* metric; they become the state-of-the-art methods due to effectiveness, flexibility, and efficiency. The extended variant, DenseAlert [Shin et al. 2017a] spotted dense subtensors for a tensor stream with an incremental algorithm. ISG+D-Spot [Yikun et al. 2019] improved the approximation ratio for the densest subtensors from $1/N$ to $1/2$ by converting an input tensor to a form of graph to reduce the number of ways and dropping all edges with weights less than a threshold; MUST [Duong et al. 2020] could estimate multiple dense subtensors at a time and provide a higher theoretical bound of the lower-bound estimated density. None of those methods, however, considers the relationships and structures of different blocks, and cannot trace the evolution of dense hierarchical patterns. The evaluation of detected subtensors is just based on a specific density measure [Jiang et al. 2015; Shin et al. 2018], which may be insufficient for some real applications.

*Hierarchical Patterns Mining.* Communities/Groups exist ubiquitously in various graphs [Chen and Saad 2010; Leskovec et al. 2008]. Their hierarchical structure and evolving behavior have also been explored in different scenes [Kumar et al. 2010; Papadimitriou et al. 2008; Rossi et al. 2020; Yang et al. 2013, 2011]; similar hierarchical patterns pertaining on topics or concepts also occur in document streams [Kleinberg 2003] and to events in temporal data [Siddique and Akhtar 2017]. [Gorovits et al. 2018] proposed a framework for joint-learning the overlapping structure and activity periods of communities. [Siddique and Akhtar 2017] detected video events with a hierarchical temporal association mining mechanism for multimedia applications. HiDDen [Zhang et al. 2017] detected hierarchical dense patterns in graph and also finds financial fraud. [Sariyüce and Pinar 2016; Sariyuce et al. 2015] used nucleus and k-core decomposition to compute and keep track of the hierarchy of dense subgraphs given by the peeling algorithm; [Sariyüce et al. 2018] presented local algorithms' framework using high-order structures during the decomposition to offer high

Table 1. Comparison of CatchCore and Related Methods (✓, △ Denote "Supported" and "Partly Supported", Respectively)

| | CatchCore | CPD [Kolda and Bader 2009] | D-Cube [Shin et al. 2017b] | M-Zoom [Shin et al. 2016] | M-Biz [Shin et al. 2018] | ISG+D-Spot [Yikun et al. 2019] | CrossSpot [Jiang et al. 2015] | MUST [Duong et al. 2020] | DSBM [Yang et al. 2011] | HiDDen [Zhang et al. 2017] | Fraudar [Hooi et al. 2016] |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Tensor data** | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | | |
| **Multiple subtensors** | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | | |
| **Hierarchical structure** | ✓ | | | | | | | | | ✓ | |
| **Detection evaluation** | ✓ | | △ | △ | | △ | | △ | ✓ | | △ |
| **Item-subtensor query** | ✓ | | | | | | | | | | |
| **Local optimality** | ✓ | | | | ✓ | | ✓ | | | ✓ | |
| **Linear scalability** | ✓ | | ✓ | ✓ | ✓ | ✓ | ✓ | | | ✓ | ✓ |

scalability for parallelization and approximations. As for the tensor, some works [Grasedyck 2010; Song et al. 2013] considered the hierarchical tensor decomposition rather than the dense structure embedded in such multidimensional arrays. In comparison, our method can deal with the multi-hierarchical structure in tensor data, provide an information-theoretical measurement for the detection result, and analyze the performance from many angles.

*Anomaly and Fraudulent Detection.* The survey [Akoglu et al. 2015] presents a systematic, thorough overview and summary of the methods of detecting anomalies in graphs; incorporating additional dimensions does benefit to identifying suspicious patterns in tensors with higher specificity, such as lockstep behavior and money laundering. The dense subgraphs or dense subtensors usually contain suspicious patterns, such as fraudsters in social networks [Hooi et al. 2016; Wong et al. 2018; Zhang et al. 2017], port-scanning activities in network analysis [Jiang et al. 2015; Shin et al. 2017b], as well as fake reviews [Shin et al. 2017b], vandalism and rating-attacks [Jiang et al. 2015; Shin et al. 2017b, a]. Another related line of research for discovering rare behavior patterns in graphs includes belief propagation [Akoglu et al. 2013; Eswaran et al. 2017], HITS-like methods [Ghosh et al. 2012; Pandit et al. 2007], and change detection [Gorovits et al. 2018; Wong et al. 2018]. With the great success of graph representation learning and graph neural networks, they can incorporate various types of information and have been widely used for anomaly detection [Ma et al. 2021] for both semi-supervised and unsupervised scenarios. GAL [Zhao et al. 2020] utilized the marginal loss to train GNNs for anomaly-detectable node representation and combines outliers and the dense subgraph detected by several of the aforementioned unsupervised approaches. Many proposed GNN models are applied to fake news and rumor detection [Bian et al. 2020; Wang et al. 2019b], fraud detection [Liu et al. 2021; Wang et al. 2019a], and other financial applications [Dastidar et al. 2022; Liu et al. 2020; Yang et al. 2020; Zhang et al. 2022]. There are still few works for pattern mining in tensors with GNNs.

As seen in Table 1, CatchCore can detect multiple dense subtensors with a hierarchical structure for tensors. It provides tools to evaluate the detection results based on the MDL principle and can be applied to item query scenarios; it also has a local optimality guarantee for the detection results and is linearly scalable.

Table 2. Symbols and Definitions

| Symbol | Definition |
|---|---|
| $\mathcal{R}(A_1, \ldots, A_N, C)$ | A relation with $N$ dimension attributes w.r.t. a tensor |
| $t(a_1, \ldots, a_N, c)$ | An entry (tuple) of the tensor $\mathcal{R}$ |
| $N$ | Number of modes (ways) in the tensor $\mathcal{R}$ |
| $A, a$ | Attributes and one of its value for the tensor $\mathcal{R}$ |
| $\mathcal{R}_n$ | Set of distinct values in $n$th attribute $A_n$ of $\mathcal{R}$ |
| $M_\mathcal{R}, V_\mathcal{R}, D_\mathcal{R}$ | Mass, volume, and cardinality-sum of the tensor $\mathcal{R}$ |
| $nnz(\mathcal{R})$ | Number of non-zero tuples in $\mathcal{R}$ |
| $\rho(\mathcal{R})$ | Density measure of tensor $\mathcal{R}$ |
| $\mathcal{B} \preccurlyeq \mathcal{R}$ | Subtensor relationship: $\mathcal{B}$ is a subtensor of $\mathcal{R}$ |
| $\boldsymbol{x}_n \in \{0,1\}^{\lvert\mathcal{R}_n\rvert}$ | An indicator vector for $n$th mode in $\mathcal{R}$ |
| $\mathbf{X}_\mathcal{B} = [\boldsymbol{x}_i]_{i=1}^N$ | List of indicator vectors for the subtensor $\mathcal{B}$ |
| $\mathcal{R} \bar{\times} \mathbf{X}$ | Full-mode product for $\mathcal{R}$ and $\mathbf{X}$ |
| $K$ | Maximum number of the hierarchies |
| $\lambda$ | Coefficient of the regularization term |
| $p\ (> 0)$ | Penalty for each missing (i.e., zero-valued) entry |
| $\eta\ (> 1)$ | Density ratio between two adjacent hierarchies |

## 3 NOTIONS AND DEFINITIONS

In this section, we introduce the notions and concepts used in our article. Throughout the article, vectors are denoted by boldface lower cases (e.g., $\boldsymbol{x}$), the scalars are denoted by lowercase letters (e.g., $c$), and $\lceil x \rceil \equiv \{1, \ldots, x\}$. Symbols frequently used in the article are listed in Table 2, and a toy example is given in Example 3.1.

Let $\mathcal{R}(A_1, \ldots, A_N, C)$ be a relation consisting of $N$ dimension attributes denoted by $\{A_1, \ldots, A_N\}$, and a non-negative measure attribute $C \in \mathbb{N}^{\geq 0}$, (see Example 3.1 for a running example and its pictorial illustration). We use $\mathcal{R}_n$ to denote the set of distinct values of $A_n$ whose element is $a_k \in \mathcal{R}_n$. For each entry (tuple) $t \in \mathcal{R}$ and for each $n \in \lceil N \rceil$, we use $t[A_n]$ and $t[C]$ to denote the values of $A_n$ and $C$ in $t$, respectively, i.e., $t[A_n] = a_n$ and $t[C] = c$. Thus, the relation $\mathcal{R}$ is represented as an $N$-way tensor of size $|\mathcal{R}_1| \times \cdots \times |\mathcal{R}_N|$; the value of entry $t$ of the tensor is $t[C]$, where it will be 0 if the corresponding tuple does not exist. Let $\mathcal{R}(n, a_n) = \{t \in \mathcal{R};\ t[A_n] = a_n\}$ denote all the entries of $\mathcal{R}$ where its attribute $A_n$ is fixed to be $a_n$.

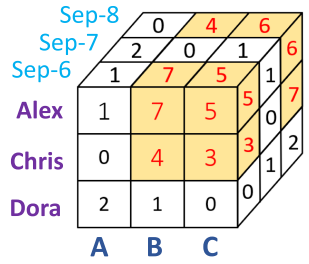
We define the mass of the tensor $\mathcal{R}$ as $M_\mathcal{R} = \sum_{t \in \mathcal{R}} t[C]$, i.e., the sum of values of all tuples in $\mathcal{R}$; the volume of $\mathcal{R}$ is defined as $V_\mathcal{R} = \prod_{n=1}^N |\mathcal{R}_n|$ and the cardinality sum of all dimensions of $\mathcal{R}$ is $D_\mathcal{R} = \sum_{n=1}^N |\mathcal{R}_n|$.

Let $\mathcal{B}_n$ be a subset of the $n$th attribute $A_n$ values of $\mathcal{R}_n$ for some $n \in \lceil N \rceil$, i.e., $\mathcal{B}_n \subseteq \mathcal{R}_n$. We define a subtensor of the tensor $\mathcal{R}$ as, $\mathcal{B} = \{t \in \mathcal{R};\ t[A_n] \in \mathcal{B}_n \subseteq \mathcal{R}_n, \forall n \in \lceil N \rceil\}$, i.e., it is the set of tuples where each attribute $A_n$ has a value in $\mathcal{B}_n$. With the tensor terminology, the relation $\mathcal{B}$ forms a "block" of size $|\mathcal{B}_1| \times \cdots \times |\mathcal{B}_N|$. We use $\mathcal{B} \preccurlyeq \mathcal{R}$ to depict that $\mathcal{B}$ is the subtensor of $\mathcal{R}$. Likewise, the mass, volume, and cardinality-sum of $\mathcal{B}$ are similar to those of $\mathcal{R}$.

Mathematically, for any $n \in \lceil N \rceil$, we can use an indicator vector $\boldsymbol{x} \in \{0,1\}^{|\mathcal{R}_n|}$ to denote whether any $a_n \in \mathcal{R}_n$ belongs to $\mathcal{B}_n$, and $\boldsymbol{x}[a_n] = 1$ iff $\mathcal{B}(n, a_n) \subseteq \mathcal{R}(n, a_n)$; we use the $\ell_1$-norm, i.e., $||\boldsymbol{x}||_1$, to denote the number of non-zero elements in the vector $\boldsymbol{x}$. Thus, the inclusion relationship between $\mathcal{B}$ and $\mathcal{R}$ can be represented with a list of indicator vectors $\mathbf{X}_\mathcal{B} = [\boldsymbol{x}_n \in \{0,1\}^{|\mathcal{R}_n|}; \forall n \in \lceil N \rceil]$. Specially, $\mathbf{X_0} = [\mathbf{0}^{|\mathcal{R}_n|}; \forall n \in \lceil N \rceil]$ corresponds to **NULL** tensor ($\emptyset$), and $\mathbf{X_1} = [\mathbf{1}^{|\mathcal{R}_n|}; \forall n \in \lceil N \rceil]$ corresponds to $\mathcal{R}$. Given an indicator vector $\boldsymbol{x} \in \{0,1\}^{|\mathcal{R}_n|}$ for a tensor $\mathcal{R}$, the subtensor, whose $n$th dimension consists of $\{a; \boldsymbol{x}[a] = 1, a \in \mathcal{R}_n\}$, can be denoted as $\mathcal{R} \times_n \boldsymbol{x}$, where "$\times_n$" is

| User | Item | Date | #Count |
|---|---|---|---|
| Alex | A | Sep-6 | 1 |
| Alex | B | Sep-6 | 7 |
| Chris | B | Sep-8 | 4 |
| Dora | A | Sep-6 | 2 |
| Alex | C | Sep-7 | 1 |
| ⋮ | ⋮ | ⋮ | ⋮ |

(a) 3 dimensions relation $\mathcal{R}$ (only a small part of colored tuples).




(b) Tensor Representation of the relation $\mathcal{R}$.

**User**: $\mathbf{x}_1 = [1, 1, 0]$
**Item**: $\mathbf{x}_2 = [0, 1, 1]$
**Date**: $\mathbf{x}_3 = [1, 0, 1]$

(c) Indicator vectors for attributes of $\mathcal{B}$.

Fig. 2. **Pictorial description of Example 3.1**. (a) Relation $\mathcal{R}$(user, item, date, #count). (b) 3-way tensor representation of $\mathcal{R}$, the colored region indicates a subtensor $\mathcal{B}$ formed by some colored tuples in relation $\mathcal{R}$. (c) The list of indicator vectors representation for subtensor $\mathcal{B}$ can be denoted as $\mathbf{X}_\mathcal{B} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3]$, and $V_\mathcal{B} = \prod_{i=1}^{N=3} ||\boldsymbol{x}_i||_1 = 2 * 2 * 2 = 8$.

the *n-mode product* for a tensor and a vector [Kolda and Bader 2009].[1] In addition, the subtensor $\mathcal{B}$ of $\mathcal{R}$ can be computed with the corresponding list of indicator vectors $\mathbf{X}_\mathcal{B}$ as: $\mathcal{B} = \mathcal{R} * (\boldsymbol{x}_1 \circ \cdots \circ \boldsymbol{x}_N)$, where "$*$" is the *Hadamard product* of tensors and "$\circ$" is the *outer product* of vectors.

*Example 3.1 (Review History).* Assume a relation $\mathcal{R}$(user, item, date, #count), its three dimension attributes are { user, item, date }, the measure attribute $C$ is the number of "count". Each tuple $t = (u, i, d, c)$ in $\mathcal{R}$ indicates that the user $u$ visited the item $i$ by date $d$ in total $c$ times.

As a toy example of relation $\mathcal{R}$ shown in Figure 2(a), $\mathcal{R}_1$ = { Alex, Chris, Dora }, $\mathcal{R}_2$ = { A, B, C }, $\mathcal{R}_3$ = { Sep-6, Sep-7, Sep-8 }. The colored regions in Figure 2(b) indicate a subtensor $\mathcal{B} \preccurlyeq \mathcal{R}$, which is composed of $\mathcal{B}_1$ = { Alex, Chris }, $\mathcal{B}_2$= { B, C }, $\mathcal{B}_3$ = { Sep-6, Sep-8 }. Taking the user dimension as an example, the indicator vector $\boldsymbol{x}_1 = [1, 1, 0]$ because $\mathcal{B}_1$ only contains "Alex" and "Chris" of $\mathcal{R}_1$ but no "Dora"; others are the same.

Many computations with respect to $\mathcal{B}$, such as its mass $M_\mathcal{B}$ and volume $V_\mathcal{B}$, are made easier with $\mathcal{R}$ and the list of indicator vectors $\mathbf{X}_\mathcal{B}$, where $\mathbf{X}_\mathcal{B} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3]$ as shown in Figure 2(c).

## 4 FRAMEWORK AND FORMULATION

In this section, we propose a unified metric, which can be optimized with the gradient-based approaches, to detect dense blocks, and then we give the formal definition of the hierarchical dense subtensors detection problem.

### 4.1 Densest Subtensor Detection Framework

Let $\mathcal{R}$ be an $N$-way tensor and $\mathcal{B}$ be a subtensor of $\mathcal{R}$, which actually corresponds to a list of indicator vectors $\mathbf{X}_\mathcal{B}$. The mass $M_\mathcal{B}$ can be computed by

$$M_\mathcal{B} = \mathcal{R} \bar{\times} \mathbf{X}_\mathcal{B} = \mathcal{R} \times_1 \boldsymbol{x}_1 \cdots \times_N \boldsymbol{x}_N, \tag{1}$$

where the *full-mode product* $\bar{\times}$ applies the $n$-mode tensor-vector product to all indicator vectors along the corresponding dimension; similarly, we use the operation $\bar{\times}_{(-n)}$ to denote conducting full-mode product, except the $n$th mode. Inspired by the density function for dense subgraph [Tsourakakis et al. 2013], we propose the following unified density metric.

[1] Entrywise, the $n$-mode product between a tensor $\mathcal{R}$ and a vector $\boldsymbol{x}$ can be denoted as: $(\mathcal{R} \times_n \boldsymbol{x})_{i_1 \ldots i_{n-1} i_{n+1} \ldots i_N} = \sum_{i_n=1}^{|\mathcal{R}_n|} t(i_1, \cdots, i_N, c) \cdot x_{i_n}$.

Table 3. Summary for Correspondence to the entry-plenum Metric

| $S_{\mathcal{X}}$ | $g(x); x = M_{\mathcal{X}}$ | $h(x); x = S_{\mathcal{X}}$ | $\phi$ | density metric |
|---|---|---|---|---|
| $D_{\mathcal{X}}$ | $\log x$ | $\log \frac{x}{N}$ | 1 | arithmetic average mass: $\rho_{ari}$ |
| $V_{\mathcal{X}}$ | $\log x$ | $\log x$ | 1 | volume density: $\rho_{vol}$ |
| | | | $\frac{1}{N}$ | geometric average mass: $\rho_{geo}$ |
| $\mathbf{X}$ | $x \cdot (\log C_1 \cdot x - 1)$ | $M_x \cdot V_x C_2 - C_1 \cdot \log V_x C_2$ | 1 | suspiciousness metric ($C_1 = \frac{1}{M_{\mathcal{R}}}, C_2 = \frac{1}{V_{\mathcal{R}}}$): $\rho_{susp}$ |

*Definition 4.1 (Entry-Plenum).* Assume $\mathbf{X}$ is a list of indicator vectors, and $\phi > 0$ is a constant. Given any two strictly increasing functions $g$ and $h$, the entry-plenum metric is defined as

$$f_\phi(\mathbf{X}; \mathcal{R}) := \begin{cases} 0 & \mathbf{X} = \mathbf{X_0} \\ g(M_{\mathcal{X}}) - \phi \cdot h(S_{\mathcal{X}}) & \text{otherwise} \end{cases}, \tag{2}$$

given the subtensor defined by $\mathbf{X}$ over $\mathcal{R}$, $M_{\mathcal{X}}$ is its mass and $S_{\mathcal{X}}$ is the property formalization related to its size. $S_{\mathcal{X}}$ can be $V_{\mathcal{X}}$, $D_{\mathcal{X}}$, or other forms.

Most popular existing subtensor density measures [Jiang et al. 2015; Shin et al. 2016, 2018, 2017b] can be subsumed into the above definition as follows. Notice that $N$ is a constant for a given $\mathcal{R}$.

- Let $g(x) = \log x, h(x) = \log \frac{x}{N}, \phi = 1$ and $S_{\mathcal{X}} = D_{\mathcal{X}}$. Then, the metric $f_{\phi=1}(\mathcal{B}) = \log \frac{M_{\mathcal{B}}}{D_{\mathcal{B}}/N} = \log \rho_{ari}(\mathcal{B})$, where $\rho_{ari}(\mathcal{B})$ is the *arithmetic average mass* of $\mathcal{B}$;
- Let $g(x) = h(x) = \log x$, $S_{\mathcal{X}} = V_{\mathcal{X}}$. If $\phi = 1$, then $f_{\phi=1}(\mathcal{B}) = \log \frac{M_{\mathcal{B}}}{V_{\mathcal{B}}} = \log \rho_{vol}(\mathcal{B})$, where $\rho_{vol}(\mathcal{B})$ is the *volume density* of $\mathcal{B}$; and if set $\phi = \frac{1}{N}$, the metric $f_{\phi=\frac{1}{N}}(\mathcal{B}) = \log \frac{M_{\mathcal{B}}}{V_{\mathcal{B}}^{1/N}} = \log \rho_{geo}(\mathcal{B})$ where $\rho_{geo}(\mathcal{B})$ is the *geometric average mass* of $\mathcal{B}$;
- Let $g(M_{\mathcal{X}}) = M_{\mathcal{X}}(\log \frac{M_{\mathcal{X}}}{M_{\mathcal{R}}} - 1)$, $h(S_{\mathcal{X}}) = h(\mathbf{X}) = M_{\mathcal{X}} \prod_{i=1}^{N} \frac{||\boldsymbol{x}_i||_1}{|\mathcal{R}_i|} - M_{\mathcal{R}} \sum_{i=1}^{N} \log \frac{||\boldsymbol{x}_i||_1}{|\mathcal{R}_i|}$ and $\phi = 1$. Then, $f_{\phi=1}(\mathcal{B}) = \rho_{susp}(\mathcal{B})$ corresponds to the *suspiciousness* for $\mathcal{B}$ [Jiang et al. 2015].

Table 3 summarizes the setting and the corresponding of different density metrics, with respect to the entry-plenum metric. In principle, the above definition is the result of a counterbalancing of two contrasting terms: the first term $g(M_{\mathcal{X}})$ favors subtensors with the large mass, whereas the second term $-\phi \cdot h(S_{\mathcal{X}})$ acts as regularization to penalize large-size block. Thus, the framework for detecting the densest subtensor can be formalized as the following problem under the entry-plenum metric.

Problem 1 (Densest $(g, h, \phi)$-entry-plenum Subtensor). ***Given*** *an $N$-way tensor $\mathcal{R}$, a constant $\phi > 0$, and a pair of increasing functions $g$ and $h$.*

***Find****: the list of indicator vectors $\mathbf{X}^*$ such that $f_\phi(\mathbf{X}^*) \geq f_\phi(\mathbf{X})$ for all feasible $\mathbf{X} = [\boldsymbol{x}_n \in \{0,1\}^{|\mathcal{R}_n|} : \forall n \in \lceil N \rceil]$. The subtensor derived from $\mathbf{X}^*$ is referred to be as the* **Densest $(g, h, \phi)$-entry-plenum Subtensor** *of the tensor $\mathcal{R}$.*

In general, finding the densest block in terms of some measures is NP-hard [Andersen and Chellapilla 2009; Shin et al. 2016], making it infeasible for large datasets. Considering scalability, existing methods [Charikar 2000; Hooi et al. 2016; Shin et al. 2016, 2017b] resort to a greedy approximation algorithm, which iteratively selects the local optimal subtensor from candidates based on a density measure defined in ratio form. For the $\rho_{ari}$ metric, they give a 1/2-approximation solution to the optimal for dense subgraph detection and a $1/N$-approximation solution for dense block detection for $N$-way tensors. However, there is no guarantee of their performance for other density metrics, which may be arbitrarily bad.

Instead, our framework formulates the densest subtensor detection problem in an optimization perspective as follows, it utilizes the list of indicator vectors $\mathbf{X}$, which can be treated as a block-variable, to make the above problem to be solvable through the *block-nonlinear* ***Gauss–Seidel (GS)***

*method* [Grippo and Sciandrone 2000] with convergence guarantee by introducing relaxation. $M_{\mathbf{X}}$ and $S_{\mathbf{X}}$ have partial derivatives for each indicator vector under this formalization, and we can use a gradient-based approach to update $\mathbf{X}$ for optimization as long as $g$ and $h$ are differentiable functions (usually satisfied). Moreover, this process is linearly scalable, as our proposed CatchCore algorithm does in Section 5.6.

### 4.2 Hierarchical Dense Subtensor Detection and Query

In practice, dense blocks in $\mathcal{R}$ may be overlapping, even inclusive, rather than being separate or flat as many dense-block detection approaches assumed [Jiang et al. 2015; Shin et al. 2016, 2017b]; we can utilize the *hierarchical dense subtensors* structure to capture them. We present the following example and definition to manifest the hierarchical structure in multi-aspect data as follows, and Figure 1(a) also gives a pictorial illustration for such a pattern for the network intrusion case. A similar phenomenon has been explored in other applications, like event detections [Siddique and Akhtar 2017] and communities in graphs [Chen and Saad 2010; Leskovec et al. 2008; Yang et al. 2013].

*Example 4.2 (Business Advertising Behaviors).* During the time for Thanksgiving advertisement promotion on social media, the whole process may go through the following stages.

Different merchants and supermarkets will start online advertising and promotions about one month before Thanksgiving; to further broaden the target user scope, they will continue to increase their promotion efforts and publicity in the days before the holiday and they will continue to increase their promotion efforts and publicity in the days before the holiday, to further broaden the target user scope; furthermore, such activities and shopping demand may continue post-holiday and into the week after Black Friday, albeit with less intensity. Consequently, the number of people buying and merchants selling also changed. Similar phenomena will be staged many times in different shopping festivals throughout the year. Thus, it will form some kind of dense block patterns among merchants, goods, and time concerning the advertising investment or sales.

Given an $N$-way tensor $\mathcal{R}$, we want to find the dense subtensors in several hierarchies, each comprising a set of different entries. We use $\rho(\mathcal{B})$ to denote the density of subtensor $\mathcal{B}$; the dense subtensor at the $k$th-hierarchy is denoted as $\mathcal{B}^k$. In order to find some meaningful patterns and to avoid getting identical subtensors crossing distinct hierarchies, we have the following definition. Given the tensor $\mathcal{B}^0 := \mathcal{R}$ and a constant $K \in \mathbb{N}^+$,

*Definition 4.3 (Hierarchical Dense Subtensors (HDS-tensors)).* For any $k \in \lceil K \rceil$, the subtensors $\mathcal{B}^{k-1}$ and $\mathcal{B}^k$ are in two adjacent hierarchies, it is required that,

(i) **density:** the densities should be significantly different from each other, $\rho(\mathcal{B}^k) \geq \eta \cdot \rho(\mathcal{B}^{k-1})$ for some $\eta > 1$.

(ii) **structure:** subtensors in higher hierarchies are more "close-knit", leading to a multi-layer dense core, $\mathcal{B}^k \preccurlyeq \mathcal{B}^{k-1}$, subjects to, $\forall n \in \lceil N \rceil, \emptyset \neq \mathcal{B}^k_n \subseteq \mathcal{B}^{k-1}_n$ and $\exists i \in \lceil N \rceil$, $\mathcal{B}^k_i \subsetneq \mathcal{B}^{k-1}_i$.

Thus, all subtensors in $K$ hierarchies consist of **Hierarchical Dense Subtensors**.

Noteworthy is the fact that it is *not feasible* to recursively apply off-the-shelf dense subtensor detection methods to find HDS-tensors because they do not consider the relationship among different blocks, even if possible, it might return trivial results (e.g., identical dense subtensors across distinct hierarchies), and it is also unclear how to design the overall objective function to be optimized by the recursive heuristic.

Formally, using the list of indicator vectors $\mathbf{X}^k$ to denote $\mathcal{B}^k$—the subtensor of the $k$th hierarchy, the HDS-tensors detection problem is formalized as follows.

Problem 2 (HDS-tensors Detection). ***Given:*** *(1) an input $N$-way tensor $\mathcal{R}$, (2) the expected density ratio between two adjacent hierarchies $\eta > 1$,[2] and (3) the maximum number of hierarchies $K \in \mathbb{N}^+$.*

***Find:*** *a set of the list of indicator vectors $\{\mathbf{X}^1, \ldots, \mathbf{X}^r\}, r \leq K$ for hierarchical dense subtensors in $r$ significant hierarchies, satisfying that $\rho(\mathbf{X}^r) \geq \eta \cdot \rho(\mathbf{X}^{r-1}) \geq \cdots \geq \eta^{r-1} \cdot \rho(\mathbf{X}^1)$, and $\mathbf{X}^r \preccurlyeq \mathbf{X}^{r-1} \preccurlyeq \cdots \preccurlyeq \mathbf{X}^1$.*

Besides, we might also be interested in finding those HDS-tensors containing special items (with specific values for some dimensions of the tensor $\mathcal{R}$) in some applications. That is, we can detect the dense patterns containing some specific query objects, such as the traffic jam on some parts of the road net or periods, collaboration communities of some specific author, and behaviors of some pre-known suspects, which is similar to the semi-supervised cases. Therefore, we formally define such a problem for pattern query as follows.

Problem 3 (Query-specific HDS-tensors Detection). ***Given:*** *(1) an input $N$-way tensor $\mathcal{R}$, (2) the expected density ratio between two adjacent hierarchies $\eta > 1$, (3) the maximum number of hierarchies $K \in \mathbb{N}^+$, and (4) a set of the query items $\mathcal{Q} = \{(q, a_q)\}$ such that the value of $q$th dimension is $a_q$ in $\mathcal{R}$.*

***Find:*** *a set of the list of indicator vectors, $\{\mathbf{X}^1, \ldots, \mathbf{X}^r\}$, $r \leq K$, for the dense subtensors in $r$ significant hierarchies such that the query items $\mathcal{Q}$ are contained in all the results, and also satisfy the properties of HDS-tensors.*

In addition, we define a three-level coordinate $(k, n, i)$ to index the list of indicator vectors for convenience, i.e., $\mathbf{X}_{(k,n,i)}$ denotes the $i$th scalar element $x_i$ of the $n$th indicator vector $\boldsymbol{x}_n$ in $\mathbf{X}^k$. Also, $\mathbf{X}_{(k,\cdot,\cdot)}$ and $\mathbf{X}_{(k,n,\cdot)}$ represent $\mathbf{X}^k$ and indicator vector $\boldsymbol{x}_n$ of $\mathbf{X}^k$, respectively.

## 5 PROPOSED METHOD

In this section, we first give the optimization formulation for the dense subtensor detection and extend it to HDS-tensors. Then, we illustrate our proposed method, CatchCore, which is an optimization-based algorithm to solve the problem. Moreover, we design to measure the quality of detection results and perform model selection under the MDL principle. We finally provide an analysis of the properties of CatchCore.

### 5.1 Optimization Formulation for Dense Subtensor Detection

Here, we provide an **interpretable instantiation** of the entry-plenum metric based on the *volume density*. With the list of indicator vectors $\mathbf{X}_{\mathcal{B}}$ of the subtensor $\mathcal{B}$, the density is represented as $\rho_{\mathcal{B}} = \frac{M_{\mathcal{B}}}{V_{\mathcal{B}}} = \frac{M_{\mathcal{B}}}{\prod_{\boldsymbol{x} \in \mathbf{X}_{\mathcal{B}}} ||\boldsymbol{x}||_1}$, where the $V_{\mathcal{B}}$ is computed as the product of the size of the indicator vector for each tensor mode, i.e., $\prod_{\boldsymbol{x} \in \mathbf{X}_{\mathcal{B}}} ||\boldsymbol{x}||_1$, which equals the total number of possible tuples and may include some zeros.

To find the dense subtensors, if we directly maximize the density measure $\rho_{\mathcal{B}}$, however, it will lead to some trivial solutions, being the entries with maximum measure value $C$ or any single entry in binary-valued tensor; while maximizing the vector-based subtensor mass defined with Equation (1) by optimizing $\mathbf{X}_{\mathcal{B}}$ will also engender another trivial result—the $\mathcal{R}$ itself, since no size or volume limitation is taken into account.

[2] More generally, we can also set different density ratios between hierarchies rather than the fixed one parameter for specific concern.

Intuitively, we need to **maximize** the mass of entries while **minimizing** the mass of missing entries in the block. Therefore, we propose the optimization goal as $\max_{\mathbf{X}:=[\boldsymbol{x}_1,\ldots,\boldsymbol{x}_N]} \mathcal{F}(\mathbf{X})$, where

$$\mathcal{F}(\mathbf{X}) = (1+p)M_{\mathbf{X}} - p\prod_{\boldsymbol{x}_n\in\mathbf{X}} ||\boldsymbol{x}_n||_1\,, \quad \textbf{s.t. } \boldsymbol{x}_n \in \{0,1\}^{|\mathcal{R}_n|},\ \forall\, n \in \lceil N\rceil, \tag{3}$$

where $M_{\mathbf{X}} = \mathcal{R}\bar{\times}\mathbf{X}$ and $p > 0$ is the penalty parameter.

The rationale behind the above definition is that each existing entry $t$ in the resultant subtensor contributes $t[C]$ as itself to $\mathcal{F}(\mathbf{X})$, while each zero-valued entry $\tilde{t}$ is penalized by a value of $p$ (i.e., $\tilde{t}[C] = 0 - p = -p$). In this manner, the objective function maximizes the total mass in the resultant subtensor while minimizing the total penalty of the missing entries. Moreover, the objective function corresponds to the $(g,h,\phi)$-entry-plenum metric under the setting that $g(x) = h(x) = x$, $\phi = \frac{p}{1+p}$, and $S_{\mathbf{X}} = V_X$.

Generally, the optimization of the objective function $\mathcal{F}(\cdot)$ is *NP-hard* due to the combinatorial nature stemming from the binary constraints in Equation (3). So, we relax these constraints from a $0-1$ **integer programming** (**IP**) problem to a polynomial-time solvable **linear programming** (**LP**) problem, that is, $\mathbf{0}^{|\mathcal{R}_n|} \le \boldsymbol{x}_n \le \mathbf{1}^{|\mathcal{R}_n|}$. To eliminate the effect of the limited iterations or some possible unstable cases, only those attributes with a value of 1 in $\boldsymbol{x}_n$ or the rounded $\boldsymbol{x}_n$ will be selected; besides, they will be fully converged in most cases.

### 5.2 Hierarchical Dense Subtensors (HDS-tensors) Detection

Based on the formulation for finding a single dense subtensor, intuitively, we can maximize the objective function in Equation (3) for each hierarchy to detect $K$ hierarchical dense subtensors, i.e., to maximize $\sum_{k=1}^{K}\mathcal{F}(\mathbf{X}^k)$, and also consider the aforementioned prerequisites of HDS-tensors. Specifically, the *density* (i) constraint is represented as $\rho_{\mathbf{X}^{k+1}} \ge \eta\cdot\rho_{\mathbf{X}^k}$ for the $k$th hierarchy with a density increase ratio $\eta > 1$; and for the *structure* (ii) requirement ($\mathcal{B}^{k+1} \preccurlyeq \mathcal{B}^k \preccurlyeq \mathcal{B}^{k-1}$), we impose additional constraints on indicator vectors to prevent identical results, as $\mathbf{X}_{(k+1,n,\cdot)} \le \mathbf{X}_{(k,n,\cdot)} \le \mathbf{X}_{(k-1,n,\cdot)}, \forall\, n \in \lceil N\rceil$, by setting $\mathbf{X}^0 = \mathbf{X_1}$ and $\mathbf{X}^{K+1} = \mathbf{X_0}$.

Consequently, the overall optimization formulation is as follows:

$$\begin{aligned}\max_{\mathbf{X}^1,\ldots,\mathbf{X}^K}\ &\sum_{k=1}^{K}\mathcal{F}(\mathbf{X}^k)\\ \text{s.t. }\ &\rho_{\mathbf{X}^{h+1}} \ge \eta\cdot\rho_{\mathbf{X}^h},\ \mathbf{X}_{(h+1,n,\cdot)} \le \mathbf{X}_{(h,n,\cdot)} \le \mathbf{X}_{(h-1,n,\cdot)}.\quad \forall\, h\in\lceil K\rceil;\ \forall n\in\lceil N\rceil.\end{aligned} \tag{4}$$

Obviously, this ***bound-constrained multi-variable nonlinear programming (BMV-NLP)*** optimization problem is non-convex and numerically intractable to solve. So, we take the following relaxation for the constraints to make it a regularization term in the objective function. Let $d^{k-1} = \eta\cdot\rho_{\mathbf{X}^{k-1}}$, which is a constant w.r.t. $\mathbf{X}^k$, the corresponding regularization term can be written as $\mathcal{G}(\mathbf{X}^k) = \mathcal{R}\bar{\times}\mathbf{X}^k - d^{k-1}\prod_{n=1}^{N}\left\|\mathbf{X}_{(k,n,\cdot)}\right\|_1$, being also the entry-plenum form. Thus, the objective function with relaxation constraints for HDS-tensors detection becomes

$$\begin{aligned}\max_{\mathbf{X}^1,\ldots,\mathbf{X}^K}\ &\sum_{k=1}^{K}\mathcal{F}(\mathbf{X}^k) + \lambda\sum_{j=2}^{K}\mathcal{G}(\mathbf{X}^j)\\ \text{s.t. }\ &\mathbf{X}_{(h+1,n,\cdot)} \le \mathbf{X}_{(h,n,\cdot)} \le \mathbf{X}_{(h-1,n,\cdot)}.\quad \forall\, h\in\lceil K\rceil;\ \forall n\in\lceil N\rceil,\end{aligned} \tag{5}$$

where the parameter $\lambda$ controls the importance of the regularization term. Thus, for $k \ge 2$, we have

$$\overline{\mathcal{F}}(\mathbf{X}^k) = (1+p+\lambda)\mathcal{R}\bar{\times}\mathbf{X}^k - (p+\lambda d^{k-1})\prod_{n=1}^{N}\left\|\mathbf{X}_{(k,n,\cdot)}\right\|_1.$$

**ALGORITHM 1:** CatchCore for HDS-tensors detection

**Require:** (1) the $N$-way tensor: $\mathcal{R}$ (2) the maximum number of hierarchies: $K$ (3) the penalty value for the missing entry: $p$ (4) the density ratio between two adjacent hierarchies: $\eta$ (5) the regularization parameter: $\lambda$ (6) the maximum number of iterations: $t_{\max}$

**Ensure:** The dense subtensors indicator vector collections: $\{\mathbf{X}^1, \cdots, \mathbf{X}^r\}$.

initialize $\mathbf{X}^1, \cdots, \mathbf{X}^K$ as $\mathbf{X}^{\text{init}}_{(k,n,\cdot)}$
$t \leftarrow 1$, $r \leftarrow 1$
**while** $t \le t_{\max}$ **and** Equation (9) is not satisfied **do** ▷ *Stop criteria*
  ▷ *Gauss-Seidel method for updating*
  **for** $k \leftarrow 1 \ldots K$ **do** ▷ *For the k-th hierarchy*
    **for** $n \leftarrow 1 \ldots N$ **do** ▷ *For the n-th dimension*
      $\boldsymbol{x}^k_n \leftarrow$ OneWayOpt $(\boldsymbol{x}^k_n)$ ▷ *Block coordinate descent*
    update $\mathbf{X}^k$
  $t \leftarrow t + 1$
**while** $r \le K$ **do** ▷ *Select significant subtensors*
  $\mathcal{S} = \{\mathbf{X}_{(r,n,\cdot)}; \max \mathbf{X}_{(r,n,\cdot)} < 1, \forall n \in \lceil N \rceil\}$
  **if** $\mathcal{S} \neq \emptyset$ **then**
    **break** ▷ *Only keep significantly dense subtensors*
  **else**:
    $r \leftarrow r + 1$
**return** The resultant $r$ indicator vector collections $\{\mathbf{X}^1, \cdots, \mathbf{X}^r\}$.

Then, the objective function in Equation (5) can be rewritten as $\mathcal{F}(\mathbf{X}^1) + \sum_{k=2}^{K} \overline{\mathcal{F}}(\mathbf{X}^k)$. We can see that in $\overline{\mathcal{F}}$, a larger penalty parameter is imposed on the missing entries of the $k$th hierarchy compared with that of the $(k-1)$-th hierarchy ($k \ge 2$), aiming at engaging the density diversity for different hierarchies.

### 5.3 Optimization Algorithms: CatchCore

In this section, we first explain the optimization techniques and then present the algorithm CatchCore to solve the problem. Algorithm 1 summarizes the complete structure of CatchCore.

When using the LP methods to solve the BMV-NLP problem, Equation (5) becomes a non-convex and higher-order bounded function with respect to each list of indicator vectors $\mathbf{X}_{(k,\cdot,\cdot)}$, it allows us to apply the alternative update method by fixing all variables for other hierarchies to be constants except for the current list in each iteration. This strategy can also be used to update each indicator vector $\mathbf{X}_{(k,n,\cdot)}$ for each hierarchy.

Based on the *structure* constraint, for any dimension $n \in \lceil N \rceil$, the feasible solution $\mathbf{X}_{(k,n,\cdot)}$ is bounded by the indicator vectors $\mathbf{X}_{(k-1,n,\cdot)}$ and $\mathbf{X}_{(k+1,n,\cdot)}$ from the two adjacent hierarchies in the high-dimensional space. We relax these constraints to $\mathbf{0}^{|\mathcal{R}_n|} \le \mathbf{X}_{(k,n,\cdot)} \le \mathbf{X}_{(k-1,n,\cdot)}$ for optimizing. Thus, we can obtain $\mathbf{X}^1, \mathbf{X}^2, \cdots, \mathbf{X}^K$ in order. That is, we first get $\mathbf{X}^1$ with the constraints $\{\mathbf{0}^{|\mathcal{R}_n|} \le \mathbf{X}_{(1,n,\cdot)} \le \mathbf{1}^{|\mathcal{R}_n|}, \forall n \in \lceil N \rceil\}$ by ignoring the constraints of other variables in other $\mathbf{X}^k$s, then we obtain $\mathbf{X}^2$ based on the result at the first step under the constraints $\{\mathbf{0}^{|\mathcal{R}_n|} \le \mathbf{X}_{(2,n,\cdot)} \le \mathbf{X}_{(1,n,\cdot)}, \forall n \in \lceil N \rceil\}$ and also ignore other constraints. In this way, we can solve the $K$ dense subtensors detection subproblems hierarchy-by-hierarchy.

Technically, we adopt the trust-region approach [Coleman and Li 1996] to solve each nonlinear box-constrained programming subproblem. We rewrite the optimization problem in Equation (5)

as $\min_{\mathbf{X}^1,\ldots,\mathbf{X}^K} f(\mathbf{X}^1,\ldots,\mathbf{X}^K)$, where

$$\begin{aligned} f(\mathbf{X}^1,\ldots,\mathbf{X}^K) = &-(1+p)\mathcal{R}\bar{\times}\mathbf{X}_{(1,\cdot,\cdot)} + p\prod_{n=1}^{N}\left\|\mathbf{X}_{(1,n,\cdot)}\right\|_1 - (1+p+\lambda)\sum_{k=2}^{K}\mathcal{R}\bar{\times}\mathbf{X}_{(k,\cdot,\cdot)} \\ &+\sum_{k=2}^{K}(p+\lambda d^{k-1})\prod_{n=1}^{N}\left\|\mathbf{X}_{(k,n,\cdot)}\right\|_1 \\ \text{s.t. } &\mathbf{X}_{(h+1,n,\cdot)} \le \mathbf{X}_{(h,n,\cdot)} \le \mathbf{X}_{(h-1,n,\cdot)}. \qquad \forall h \in \lceil K\rceil;\ \forall n \in \lceil N\rceil. \end{aligned} \tag{6}$$

Specifically, we use the alternative projected gradient descent method [Lin 2007; Lin and Moré 1999], which is simple and efficient for solving the optimization problem. For any dimension $n$, we denote the gradient of Equation (1) with respect to $\boldsymbol{x}_n$ as

$$\nabla_{\boldsymbol{x}_n} M_{\mathcal{B}} = \mathcal{R}\bar{\times}_{(-n)}\mathbf{X}_{\mathcal{B}} = \mathcal{R}\times_1 \boldsymbol{x}_1\cdots\times_{(n-1)}\boldsymbol{x}_{(n-1)}\times_{(n+1)}\boldsymbol{x}_{(n+1)}\cdots\times_N \boldsymbol{x}_N.$$

Then, the gradient of $f(\cdot)$ with respect to $\boldsymbol{x}_n^1$ ($\mathbf{X}_{(1,n,\cdot)}$) and $\boldsymbol{x}_n^k$ ($\mathbf{X}_{(k\ge 2,n,\cdot)}$) are

$$\begin{aligned} \nabla_{\boldsymbol{x}_n^1} f &= -(1+p)\nabla_{\boldsymbol{x}_n^1} M_{\mathbf{X}^1} + p\prod_{\boldsymbol{x}_n\in\mathbf{X}^1/\{\boldsymbol{x}_n^1\}} ||\boldsymbol{x}_n||_1\,\mathbf{1}, \\ \nabla_{\boldsymbol{x}_n^k} f &= -(1+p+\lambda)\nabla_{\boldsymbol{x}_n^k} M_{\mathbf{X}^k} + (p+\lambda d^{k-1})\prod_{\boldsymbol{x}_n\in\mathbf{X}^k/\{\boldsymbol{x}_n^k\}} ||\boldsymbol{x}_n||_1\,\mathbf{1}, \end{aligned} \tag{7}$$

where $\mathbf{1}$ is an all-ones vector (i.e., the same size as $\boldsymbol{x}_n^1$ and $\boldsymbol{x}_n^k$). Let $\boldsymbol{x}_n$ be the current iterator vector of any $k$th hierarchy in the projected gradient approach, the new iterator is given by the update rule $\tilde{\boldsymbol{x}}_n = P(\boldsymbol{x}_n - \alpha\nabla_{\boldsymbol{x}_n} f)$. Here, the operator $P(\cdot)$ projects the vector back to the bounded feasible region (as the above-mentioned) [Bertsekas 1997], $-\nabla_{\boldsymbol{x}_n} f$ is the gradient-related search direction, and $\alpha$ is a step size computed, e.g., utilizing the Armijo step size strategy. In the case of Armijo's rule line search, a good step size $\alpha$ is chosen until the following condition is satisfied

$$f(\cdot,\mathbf{X}^k_{\boldsymbol{x}_n\to\tilde{\boldsymbol{x}}_n},\cdot) - f(\cdot,\mathbf{X}^k,\cdot) \le \sigma\cdot(\nabla_{\boldsymbol{x}_n} f)^T(\tilde{\boldsymbol{x}}_n - \boldsymbol{x}_n), \tag{8}$$

where $\mathbf{X}^k_{\boldsymbol{x}_n\to\tilde{\boldsymbol{x}}_n}$ means only replacing the indicator vector $\boldsymbol{x}_n$ with the updated version $\tilde{\boldsymbol{x}}_n$ in $\mathbf{X}^k$, and a common choice of the parameter $\sigma$ ($0<\sigma<1$) is 0.01. Thus, we can update each indicator vector $\boldsymbol{x}_n$ for the current $k$th-hierarchy alternatively.

Searching for the step size $\alpha$ is a time-consuming process. We use a more flexible line search [Lin and Moré 1999] method to decrease the number of checks for Equation (8). The OneWayOpt described in Algorithm 2 is used to update one indicator vector $\boldsymbol{x}_n$ for $\mathbf{X}^k$. Specifically, let $\alpha$ denote the step size for updating $\boldsymbol{x}_n$; we assume it is similar to the step size for updating $\tilde{\boldsymbol{x}}_n$. To update $\tilde{\boldsymbol{x}}_n$, we use $\alpha$ as the initial guess and then either increase or decrease it with a scale factor $\beta$ ($0<\beta<1$) until finding the best step size satisfying Equation (8). The core steps for efficiently searching $\alpha$ correspond to Lines 7–10, and we use $\beta = 1/10$ in our article.

Therefore, we propose the CatchCore algorithm to solve the programming optimization problem in Equation (6). First, we use the rules to initialize the list of indicator vectors by selecting the slices $\mathcal{R}(n,a_n)$ (i.e., $\mathbf{X}^{\text{init}}_{(k,n,i)}$) with 0.5 in the 1st hierarchy, and 0.01 for the others ($2\le k\le K$). In this way, we can effectively avoid some trivial results. To make the solution to be close to the stationary point regarding convergence, we apply the following common condition as a stop criterion for the bounded-constrained optimization approach as well as the limitation for the total number of iterations, $t_{\max}$.

$$\left\|\left\{\nabla^P_{\boldsymbol{x}_n^k} f;\forall n,\ k\right\}\right\|_2 \le \epsilon\left\|\left\{\nabla^P_{\mathbf{X}^{\text{init}}_{(k,n,\cdot)}} f;\forall n,\ k\right\}\right\|_2, \tag{9}$$

**ALGORITHM 2:** OneWayOpt for updating one indicator vector

**Require:** (1) the indicator vector $\boldsymbol{x}_n$ to be updated for $\mathbf{X}^k$ (2) the ratio of decreasing the step size: $\beta \in (0, 1)$ (3) the linear search parameter: $\sigma \in (0, 1)$
**Ensure:** the newly updated indicator vector $\tilde{\boldsymbol{x}}_n$.
1: compute the gradient $\nabla_{\boldsymbol{x}_n} f$ as the Equation (7).
2: compute the objective function $f(\cdot, \mathbf{X}^k, \cdot)$ with $\boldsymbol{x}_n$ in the Equation (6)
3: initial $\alpha \leftarrow 1$ ▷ *Step size for updating*
4: **while not** satisfied the Armijo's condition **do**
5: $\tilde{\boldsymbol{x}}_n = P(\boldsymbol{x}_n - \alpha \nabla_{\boldsymbol{x}_n} f)$ ▷ *Obtain the new* $\boldsymbol{x}_n$
6: compute the objective function $f(\cdot, \mathbf{X}^k_{\boldsymbol{x}_n \to \tilde{\boldsymbol{x}}_n}, \cdot)$ with $\tilde{\boldsymbol{x}}_n$
7: adjust the step size $\alpha$ with $\beta$ based on the condition in Equation (8).
8: **return** the final $\tilde{\boldsymbol{x}}_n$.

where $\nabla^P_{\boldsymbol{x}_n^k} f$ is the *element-wise projected gradient*. For the $i$th element, it is defined as

$$(\nabla^P_{\boldsymbol{x}_n^k} f)_i = \begin{cases} \min(0, (\nabla^P_{\boldsymbol{x}_n^k} f)_i) & \text{if } \mathbf{X}_{(k,n,i)} = \mathbf{X}_{(k+1,n,i)}, \\ (\nabla^P_{\boldsymbol{x}_n^k} f)_i & \text{if } \mathbf{X}_{(k+1,n,i)} < \mathbf{X}_{(k,n,i)} < \mathbf{X}_{(k-1,n,i)}, \\ \max(0, (\nabla^P_{\boldsymbol{x}_n^k} f)_i) & \text{if } \mathbf{X}_{(k,n,i)} = \mathbf{X}_{(k-1,n,i)}. \end{cases}$$

Then, CatchCore calls OneWayOpt to alternatively update the indicator vector corresponding to each dimension and hierarchy, iteratively. In the final, we only select those significantly dense subtensors to return (Lines 9–14), i.e., the top $r$ of $K$ hierarchies.

### 5.4 Query-specified HDS-tensors Detection

Query problem 3 defines a special case for the HDS-tensors detection where the query item set $\mathcal{Q}$ should be included in each resultant subtensor, that is, $\boldsymbol{x}_q^k[a_q] = 1, \forall 1 \le k \le K$, and $(q, a_q) \in \mathcal{Q}$ in the indicator vector paradigm. However, the optimization goal in Equation (6) becomes a *mixed integer programming* problem with this treatment, which typically requires an exhaustive search to solve. To utilize the same optimization framework for Equation (6), we relax the integer combinatorial programming problem to be a small search-range by setting these entities to a number $\bar{c}$ slightly less than 1, like $\bar{c} = 0.9$. So, the lower bound for $\boldsymbol{x}_q^{K+1}$ becomes

$$\boldsymbol{x}_q^{K+1} = \begin{cases} \bar{c} & \text{if } (q, a_q) \in \mathcal{Q}, \\ 0 & \text{otherwise.} \end{cases}$$

We reuse the optimization algorithms described in Section 5.3 to solve the query problem while forcing the indicator vector to satisfy the above constraints during each update iteration, which will lead to the resultant subtensors having to contain those target query items.

Furthermore, it is worth noting that using the tensor decomposition [Kolda and Bader 2009] result, e.g., CP decomposition, as initialization for the indicator vector can speed up our algorithm due to being closer to the local optimal.

### 5.5 Parameter Evaluation and Model Selection

Regarding the dense subtensors, the penalty value $p$ controls the lowest density of the missing entries, and the ratio parameter $\eta$ affects the density-diversity and the number of hierarchies in

the final result. It is challenging to appropriately set those parameters or evaluate the quality of detection results of some parameter configurations, especially in unsupervised applications.

We propose to measure the quality of detection results of different parameter settings based on the ***Minimum Description Length*** (**MDL**) [Adriaans and Vitanyi 2009; Rissanen 1978] where the shortest description of the data is the best model. MDL methods learn through a data compression perspective and are sometimes described as mathematical applications of Occam's razor. In such a principled manner, we compute the number of bits needed to encode the tensor $\mathcal{R}$ with the detected hierarchical dense subtensors; and select the best model (parameters) to achieve the shortest code length, where the shorter the MDL, the better the model. Intuitively, the fewer missing entries and the more accurate when detecting different hierarchies, leading to the fewer bits needed to encode $\mathcal{R}$ in a lossless compression employing the characterization.

For the indicator vector $\mathbf{X}_{(k,n,\cdot)}$ ($k \in \lceil K \rceil$, $n \in \lceil N \rceil$) we can adopt the Huffman or arithmetic coding schema to encode the binary string, which formally can be viewed as a sequence of realizations of a binomial random variable $X$. Considering $\mathbf{X}_{(k,n,\cdot)} \leq \mathbf{X}_{(k-1,n,\cdot)}$, we can only focus on the overlapping part $\overline{\boldsymbol{x}}_n^k = \{\mathbf{X}_{(k,n,i)}; \mathbf{X}_{(k-1,n,i)} = 1, \forall i \in \lceil |\mathcal{R}|_n \rceil\}$ to avoid the redundant encoding of 0s. We denote the entropy of the indicator vector $\boldsymbol{x}$ as

$$H(\boldsymbol{x}) = -\sum_{q \in \{0,1\}} P(X = q) \log P(X = q),$$

where $P(X = q) = n_q / ||\boldsymbol{x}||_1$ and $n_q$ is the number of $q$ in $\boldsymbol{x}$.

The description length of the list of indicator vectors $\mathbf{X}^k$ is

$$L(\mathbf{X}^k) = \sum_{n=1}^{N} \left( \log^* \left\| \mathbf{X}_{(k,n,\cdot)} \right\|_1 + \left\| \mathbf{X}_{(k-1,n,\cdot)} \right\|_1 \cdot H(\overline{\boldsymbol{x}}_n^k) \right),$$

where $\log^* x$ is the universal code length for an integer $x$ [Rissanen 1983].

Assuming $\mathbf{X}^{K+1} = \mathbf{X_0}$, for the dense subtensor $\mathcal{B}^k$ defined by $\mathbf{X}^k$, we only need to encode the entries in $\overline{\mathcal{B}}^k = \mathcal{B}^k - \mathcal{B}^{k+1}$ due to $\mathcal{B}^{k+1} \preccurlyeq \mathcal{B}^k$, based on some probability distribution model. For the tuple $t \in \overline{\mathcal{B}}^k$, specifically, if $t[C] \in \{0, 1\}$, $t$ is sampled from the binomial distribution; and if $t[C] \in \mathbb{N}^{\geq 0}$, we instead model the data by using the *Poisson* distribution [Jiang et al. 2015] parameterized by the density of $\overline{\mathcal{B}}^k$, i.e., $\rho_{\overline{\mathcal{B}}^k}$. Therefore, the code length for encoding $\overline{\mathcal{B}}^k$ is

$$L(\overline{\mathcal{B}}^k) = -\sum_{q \in \{t[C]; t \in \overline{\mathcal{B}}^k\}} n_q \cdot \log P(X = q) + C_{para},$$

where $P(X = q)$ is the probability of $q$ in the probability distribution function $\mathbf{P}$, and $C_{para}$ is for encoding the parameters of $\mathbf{P}$, e.g., the mean in a Poisson distribution.

As for the residual part $\overline{\mathcal{R}} = \mathcal{R} - \mathcal{B}^1$, we use Huffman coding to encode its entries, considering the sparsity and discrete properties, and the code length is denoted as $L_\epsilon$.

Putting all these parts together, we can write the total code length for representing the tensor $\mathcal{R}$ with the resultant list of indicator vectors for $K$ hierarchies as

$$L(\mathcal{R}; \mathbf{X}^1, \ldots, \mathbf{X}^K) = \log^* K + \sum_{n=1}^{N} \log^* |\mathcal{R}_n| + \sum_{k=1}^{K} L(\mathbf{X}^k) + L(\bar{\mathcal{B}}^k) + L_\epsilon. \tag{10}$$

To get a better parameter setting, we can perform a grid search over the parameter space heuristically and pick the configuration that minimizes the MDL cost. In the experiment, we demonstrated that the parameters determined by the MDL principle lead to the optimal quality for detecting the HDS-tensors while the search space is also limited.

### 5.6 Algorithm Analysis

Here, we provide the convergence analysis and time-space complexity of CatchCore.

Lemma 5.1 states the convergence properties of the gradient method for CatchCore.

Lemma 5.1 (Convergence). *CatchCore algorithm converges to a critical point (Local Optimality).*

Proof. Considering the subtensor detection as Equation (3) defines, the alternative updating steps in CatchCore come down to the block nonlinear Gauss–Seidel method.

Based on the convergence conclusion in [Grippo and Sciandrone 2000] for the quasi-convex objective function, the indicator vectors in each dimension are bounded by a closed convex set. The projected gradient algorithm that uses the Armijo procedure converges to a stationary point [Calamai and Moré 1987], where the Armijo-type **line search** (**LS**) adopted in Algorithm 2 generates a sequence of points for any dimension, and there exists at least one limit point $\tilde{\boldsymbol{x}}$, we have $(\nabla_{\boldsymbol{x}} f)^T(\tilde{\boldsymbol{x}} - \boldsymbol{x}) \geq 0$, i.e., $\tilde{\boldsymbol{x}}$ is a critical (stationary) point. This convergence result is still satisfied without any convexity assumption on the objective function.

To detect hierarchical subtensors, CatchCore updates indicator vectors hierarchy-by-hierarchy, and the convergence property is maintained since the subproblem for the subtensor detection in one hierarchy is the same as in Equation (3). Therefore, the CatchCore algorithm converges to a critical point for each limit point. □

Theorem 5.2 states the time complexity of the CatchCore algorithm, which is linear with $K$, $N$, and $nnz(\mathcal{R})$—the number of non-zero entries in $\mathcal{R}$. The space complexity is given in Theorem 5.3.

Theorem 5.2 (Worst-case Time Complexity). *Let $t_{\text{als}}$ be the number of iterations for Armoji's line search used in the OneWayOpt Algorithm for updating any indicator vector. Then, the worst-case time complexity of the CatchCore Algorithm is $O(K \cdot N \cdot t_{\max} \cdot t_{\text{als}} \cdot (nnz(\mathcal{R}) + c \cdot D_{\mathcal{R}}))$.*

Proof. Regarding the computational complexity for $M_{\mathbf{X}^k}$ and $\nabla_{\boldsymbol{x}_n} f$ in OneWayOpt Algorithm 2, they rely on the full-mode product $\bar{\times}$ or $\bar{\times}_{(-n)}$, which takes at most $O(nnz(\mathcal{R}))$ by conducting an element-wise $n$-mode product for tensor and all vectors. Computing the norm of indicator vectors takes $O(\sum_{n=1}^{N} |\mathcal{R}_n|)$, which is included in the objective function $f$, vector gradient of $f$, and stop-criteria. Thus, for $t_{\text{als}}$ searching iterations, the time complexity of OneWayOpt is $O(t_{\text{als}} \cdot (nnz(\mathcal{R}) + c \cdot D_{\mathcal{R}}))$.

The alternative updating for all variables in the CatchCore algorithm runs $t_{\max}$ times at most, so there will be $K \cdot N \cdot t_{\max}$ iterations to call OneWayOpt in total. To keep the significant subtensors, all indicator vectors in each of the $K$ hierarchies will be checked at most once, for a total of $O(K \cdot \sum_{n=1}^{N} |\mathcal{R}_n|)$ times. Hence, the worst-case time complexity of the CatchCore algorithm is $O(K \cdot N \cdot t_{\max} \cdot t_{\text{als}} \cdot (nnz(\mathcal{R}) + c \cdot D_{\mathcal{R}}))$. □

Theorem 5.3 (Memory Requirements). *The amount of memory space required by CatchCore is*

$$O(nnz(\mathcal{R}) + 2K \cdot D_{\mathcal{R}}). \tag{11}$$

Proof. CatchCore stores the tensor $\mathcal{R}$, the indicator vectors and gradient vectors when computing $M_{\mathbf{X}^k}$ and $\nabla_{\boldsymbol{x}_n^k} f$ for a list of indicator vectors $\mathbf{X}^k$. So, the memory requirement is $O(nnz(\mathcal{R}) + 2K \cdot D_{\mathcal{R}})$. □

**Parameter Analysis**: For the maximum number of significant hierarchies $K_{\max} = \log_{\eta}(\frac{\max(\mathcal{R})}{\rho_{\mathcal{R}}})$, where $\max(\mathcal{R})$ is the maximum value of the measure attributes of $\mathcal{R}$. In practice, we have the following observations, which ensure the efficiency of CatchCore,

Table 4. Summary of Real-world Datasets

| Name | Volume | $card(\mathcal{R})$ | $\mathbf{nnz}(\mathcal{R})$ |
|---|---|---|---|
| **Ratings: users × items × timestamps × rating → #reviews** | | | |
| Android | 1.32M × 61.3K × 1.28K × 5 | 1.38M | 2.23M |
| BeerAdvocate | 26.5K × 50.8K × 1,472 × 1 | 78.7K | 1.07M |
| StackOverflow | 545K × 96.7K × 1,154 × 1 | 643K | 1.30M |
| **Social network: users × users × timestamps → #interactions** | | | |
| DBLP | 1.31M × 1.31M × 72 | 2.63M | 18.9M |
| YouTube | 3.22M × 3.22M × 203 | 6.45M | 18.5M |
| **TCP dumps: IPs × IPs × timestamps → #connections** | | | |
| DARPA | 9.48K × 23.4K × 46.6K | 79.5K | 522K |
| **TCP dumps: protocol × service × flag ×⋯ → #connections** | | | |
| AirForce | 3 × 70 × 11 × 7.20K × 21.5K × 512 × 512 | 39.7K | 648K |

- $nnz(\mathcal{R}) \gg D_{\mathcal{R}}$, and $K \ll K_{\max}$, i.e., there are only a few significant hierarchies;
- $t < t_{\max}$, i.e., early stopping for iterations;
- a small $t_{\text{als}}$, i.e., few iterations for searching the step size.

The dimension-update (Line 5) could be solved separately, a situation suitable for a parallel environment.

## 6 EXPERIMENTS

We design experiments to answer the following questions:

- **Q1. Accuracy**: How accurately does CatchCore detect HDS-tensors in synthetic and real-world datasets? Does the MDL evaluation select the optimal parameters?
- **Q2. Pattern detection and anomaly detection**: What patterns does CatchCore detect in real-world data? What is the behavior of the detected anomalies?
- **Q3. Query-based pattern**: How about the behavior of specific items? Can CatchCore capture other interesting patterns in terms of query-specific HDS-tensors?
- **Q4. Scalability**: Does the CatchCore scale linearly with all aspects of data?

### 6.1 Experimental Settings

**Baselines:** We selected several state-of-the-art methods for dense-block detection as the baselines, including D-Cube [Shin et al. 2017b], M-Zoom [Shin et al. 2016], CrossSpot [Jiang et al. 2015], and CPD [Kolda and Bader 2009]. In all experiments, a sparse tensor format was used for efficient memory usage, and the $\rho_{ari}$ and $\rho_{geo}$ were used for D-Cube and M-Zoom (report the best result); we used a variant of CrossSpot which maximizes the same density metrics and used the CPD result for seed selection as did in [Shin et al. 2016].

**Data:** Table 4 summarizes the real-world datasets in our experiments. In the *Rating* category, data are 4-way tensors *(user, item, timestamp, rating)*, where the entry values are the number of reviews. In *Social network*, data are 3-way tensors *(user, user, timestamps)*, where the entry values are the number of interactions (co-authorship/favorite), and its first two attributes refer to the same set of users. *DARPA* is the TCP dumps represented with a 3-way tensor *(source IP, destination IP, timestamps)*, where the entry values are the number of connections between two IP addresses (hosts); 60% of records are labeled as anomalous (in total 89 different sorts of attacks), which are dominated by nine types of attacks, such as Neptune, Smurf, Satan, and so on. *AirForce* is also a network intrusion dataset for a typical U.S. Air Force LAN, which is represented as a 7-way tensor *(protocol, service, src_bytes, dst_bytes, flag, host_count, src_count)*, and it contains 22 types of attacks.

The TCP dumps datasets contain the label of each record, including normal or different types of attack. Timestamps are in minutes for DARPA, in dates for Ratings and YouTube, and in years for DBLP.

### 6.2 Q1. Accuracy

We compare how accurately each method detects the injected dense subtensors in the synthetic and dense patterns in real-world datasets.

#### 6.2.1 *Injected Dense Blocks Detection.*

*Single dense block.* We randomly and uniformly generated a $5K \times 5K \times 2K$ 3-way tensor $\mathcal{R}$ with a density of $3E-6$. Into $\mathcal{R}$, one $200 \times 300 \times 100$ block is injected with distinct density, for testing the detection bound of each method. Figure 3 demonstrated that CatchCore effectively detects blocks as low as a tenth of the density that the best baselines detect. In a real scenario, the synchronized behavior of fraudsters often use many resources, including fake accounts, IP addresses, and time, to conduct the same fraud to achieve the "economy of scale"; thus, the lower detection density for dense blocks means that our method can spot such fraudsters with less adversarial effort and detect they as early as possible. The result of real-world data is omitted since CatchCore had similar behavior to other competitors.

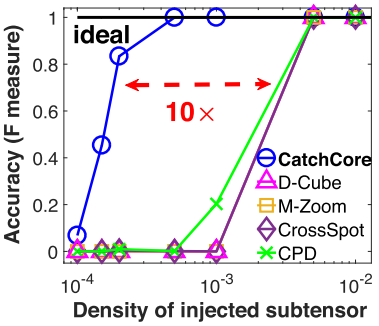



Fig. 3. **Injected dense blocks detection.** CatchCore outperforms competitors for detecting injected blocks in synthetic data, it achieves lower (10× less density) detection bound than others.

*Hierarchical dense blocks.* We injected $K$ subtensors with different sizes and densities into the BeerAdvocate and a synthetic tensor $\mathcal{R}$ in a hierarchical manner.

Tables 5 and 6 list the detection results, where $H1$ is the densest or the first subtensor detected by the methods, and density (the order of detection) decreases (increases) from $H1$ to $H3$. The information about the injected blocks is given at the bottom of the tables.

As we can see, in both the BeerAdvocate and synthetic cases, CatchCore can detect all injected subtensors in various hierarchies, sizes, and densities. It consistently outperforms other baselines, which fail to capture them accurately or miss at least one block. D-Cube and M-Zoom have similar accuracy except for some special cases as highlighted; they cannot identify the structure among dense blocks, resulting in missing some of the sparsest or densest injected blocks or being overwhelmed by some large volume blocks. CrossSpot and CPD also do not find those hierarchical dense subtensors accurately.

Moreover, CatchCore can accurately detect those non-overlapping blocks and catch higher density subtensors for dense blocks with some blank spaces or hollow. We also verified such cases via different injection strategies.

#### 6.2.2 *Dense Blocks Detection in Real World Datasets.*

We apply CatchCore to various real-world datasets and report the volume density $\rho_{vol}$ instead of mass to avoid trivial results and reflects the fillrate in the blocks for 0-1 tensors. Those blocks with a higher density contain interesting patterns with a high probability.

Figure 1(c) and Figure 4 show the densities of the top four densest blocks found by different methods for the DBLP, YouTube, StackOverflow, and Android datasets. As the figures show, for DBLP and YouTube data, CatchCore spots the denser blocks and consistently outperforms the

Table 5. Accuracy of Hierarchical Subtensors Detection for BeerAdvocate Dataset.

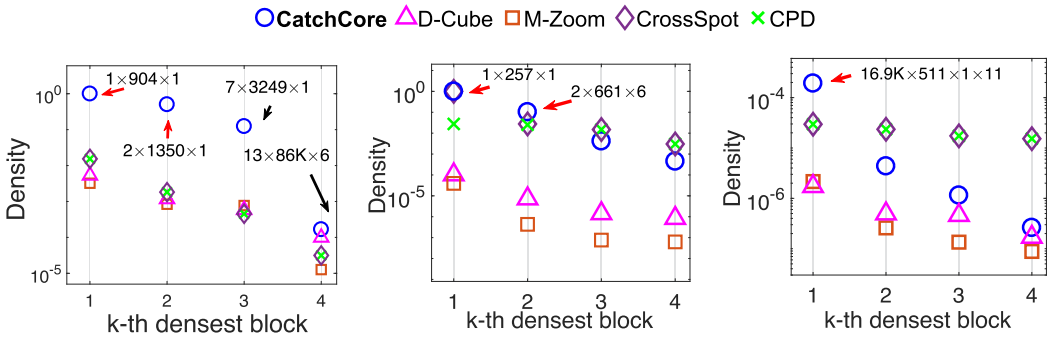
| K | Injected Densities | H1 | | | | H2 | | | | | H3 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **CC** | D/M | CS | CPD | **CC** | D | M | CS | CPD | **CC** | D | M | CS | CPD |
| 2 | 0.05 + 0.01 | 1 | 1 | 1 | 0.999 | 1 | 0.121 | 0.041 | 0 | 0.080 | | | | | |
| | 0.05 + 0.01 | **1** | 1 | 1 | 0.999 | **1** | 0.121 | 0.041 | 0 | 0.080 | | | | | |
| | 0.2 + 0.1 | **1** | 0 | 0.236 | 0.236 | **1** | 1 | 1 | 1 | 0.997 | | | | | |
| | 0.7 + 0.1 | **1** | 0 | 0.325 | 0.325 | **1** | 1 | 1 | 1 | 0.519 | | | - | | |
| | 0.7 + 0.2 | **1** | 0 | 0.959 | 0.959 | **1** | 1 | 1 | 1 | 0.998 | | | | | |
| | 1 + 0.05 | **1** | 0 | 0.784 | 0.784 | **1** | 1 | 1 | 1 | 0.998 | | | | | |
| | 1 + 0.1 | **1** | 0 | 0.751 | 0.751 | **1** | 1 | 1 | 1 | 0.998 | | | | | |
| | 1 + 0.2 | **1** | 0 | 0.795 | 0.795 | **1** | 1 | 1 | 1 | 0.999 | | | | | |
| 3 | 0.2 + 0.1 + 0.05 | **1** | 0 | 0.265 | 0.265 | **1** | 0 | 0 | 0 | 0 | **1** | 1 | 1 | 1 | 0.980 |
| | 1 + 0.2 + 0.01 | **1** | 0 | 0 | 0 | **1** | 0.999 | 1.0 | 1.0 | 0.223 | **1** | 0.444 | 0.431 | 0.746 | 0.850 |
| | 1 + 0.7 + 0.2 | **1** | 0 | 0 | 0 | **1** | 0 | 0 | 0.981 | 0.981 | **1** | 1 | 1 | 1 | 0.999 |

The algorithm abbreviations mean that CC: CatchCore, D: D-Cube, M: M-Zoom, CS: CrossSpot.
The injected shape w.r.t density is: 0.01: $1K \times 800 \times 15$, 0.05: $800 \times 600 \times 10$, 0.1: $500 \times 500 \times 5$, 0.2: $300 \times 300 \times 5$, 0.7: $200 \times 100 \times 2$, 1: $100 \times 80 \times 1$.

Table 6. Accuracy of Hierarchical Subtensors Detection for Synthetic Dataset

| K | Injected Densities | H1 | | | | H2 | | | | H3 | | | | H4 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **CC** | D/M | CS | CPD | **CC** | D/M | CS | CPD | **CC** | D/M | CS | CPD | **CC** | D/M | CS | CPD |
| 2 | 0.01 + 0.001 | 1 | 1 | 0.14 | 0.14 | 1 | 0.183 | 0.89 | 0.89 | | | | | | | | |
| | 0.1 + 0.01 | 1 | 1 | 0.25 | 0.25 | 1 | 1 | 0.89 | 0.89 | | | – | | | | | |
| | 0.25 + 0.1 | 1 | 1 | 0.35 | 0.35 | 1 | 0.257 | 1 | 1 | | | | | | | – | |
| 3 | 0.1 + 0.01 + 0.001 | 1 | 1 | 0.17 | 0.17 | 1 | 1 | 0.20 | 0.20 | 1 | 0.321 | 0.74 | 0.87 | | | | |
| | 0.25 + 0.1 + 0.01 | 1 | 0 | 0.98 | 0.98 | 1 | 1 | 0.20 | 0.20 | 1 | 1 | 0.85 | 0.98 | | | | |
| 4 | 0.25 + 0.1 + 0.01 + 0.001 | 1 | 0 | 0.96 | 0.96 | 1 | 1 | 0.51 | 0.51 | 1 | 1 | 0.19 | 0.19 | 1 | 0.359 | 0.79 | 0.95 |

*The algorithm abbreviations mean that CC: CatchCore, D: D-Cube, M: M-Zoom, CS: CrossSpot.
*The shape corresponding to different injected density is: 0.001: $200 \times 250 \times 100$, 0.01: $200 \times 100 \times 60$, 0.1:$80 \times 80 \times 30$, 0.25: $50 \times 60 \times 10$.




(a) Youtube (b) StackOverflow (c) Android

Fig. 4. **The subtensors detected by CatchCore for the real-world datasets achieve higher densities.** In each sub-figure, points represent the density of $k$th of the top 4 blocks found by the different methods. CatchCore outperforms the competitors, especially for the YouTube dataset, which contains suspicious behaviors that users in the top 2 blocks create abnormally large friendship connections with others within an hour.

competitors for all blocks; it also achieves a similar best result for StackOverflow as CrossSpot. The density of the densest blocks is 1.0 for YouTube and StackOverflow, which means that they are fully populated and correspond to cohesive connections. As Figure 4(c) for Android shows, CatchCore captures the densest block with a higher density than the result from other methods; except for the first block, CPD and CrossSpot had the same performance, which means that CrossSpot did not get an improvement over CPD, and were better than other methods. This phenomenon comes from the fact that the results obtained based on the other metrics (like $\rho_{ari}$) are not necessarily optimal or even good at the $\rho_{vol}$ density metric, for which the results of those methods adopting greedy approximation might be arbitrarily bad. Even if their densities are already larger than those of taking $\rho_{geo}$ metric, the dense blocks detected by M-Zoom and D-Cube still *contain more zeros* leading to a small density value of $\rho_{vol}$ but the highest values under $\rho_{ari}$ based on our scrutinization.

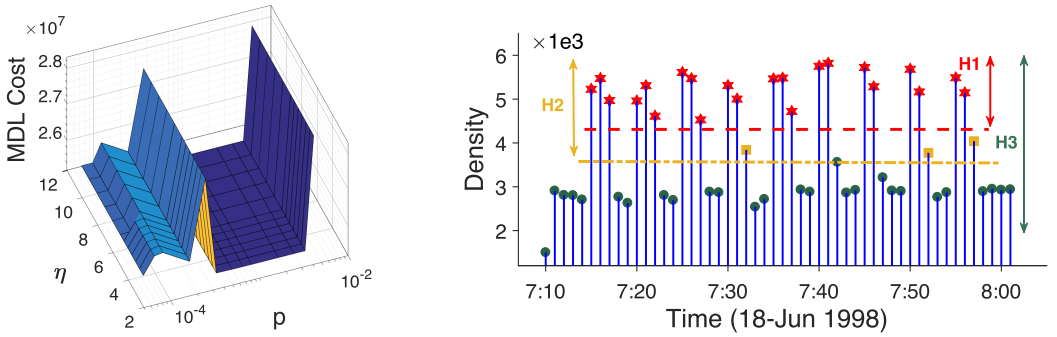
*6.2.3 MDL-based Evaluation.* We evaluate the utility of our MDL criteria for measuring the quality of hierarchies returned by CatchCore under different parameter configurations.

In this experiment, we used the BeerAdvocate data with three hierarchical injected dense blocks, i.e., the second case with $K = 3$ in Table 5. We computed the MDL cost of the detection result by varying a pair of parameters, $p$ and $\eta$. As we can see from the results in Figure 5(a), the optimal hierarchies achieve the shortest MDL with respect to $p$ and $\eta$, thus, our designed MDL measure indeed aids to select the best model (parameter configuration). In addition, our model is insensitive to parameters to some extent; that is, it can obtain optimal results for a wide range of parameters.

### 6.3 Q2. Anomaly Detection and Community Pattern

*6.3.1 Anomaly Detection.* CatchCore detected network intrusion in TCP dumps with high accuracy and identified attack patterns for the DARPA and AirForce datasets, where each intrusion connection is labeled.

Table 7 compares the F-score of the top 5 densest blocks detected by each method. CatchCore outperformed competitors for DARPA data and had a comparable performance to M-Zoom for AirForce data. CPD had the weakest results due to including more false-positive results by only relying on the tensor decomposition components; CrossSpot might achieve more refined and




(a) MDL Cost w.r.t parameter $p$ and $\eta$

(b) Network Attack Pattern

Fig. 5. (a) The optimal hierarchies achieve the shortest MDL w.r.t. the parameters $p$ and $\eta$. CatchCore can obtain optimal hierarchical dense subtensors for a wide range of parameters to some extent. (b) Hierarchical network intrusion behavior between a pair of IPs on June 18, 1998, in DARPA dataset.

Table 7. CatchCore Identifies Network Attacks from the Real-world TCP Dumps Datasets with the Best or Comparable Performance in F-score (Left)

| | DARPA | AirForce |
|---|---|---|
| CPD | 0.167 | 0.785 |
| CrossSpot | 0.822 | 0.785 |
| M-Zoom | 0.826 | 0.906 |
| D-Cube | 0.856 | **0.942** |
| CatchCore | **0.877** | 0.902 |

| H | Subtensor Shape | Anomaly Ratio |
|---|---|---|
| 1 | $1 \times 1 \times 96$ | 100% |
| 2 | $1 \times 1 \times 100$ | 100% |
| 3 | $1 \times 1 \times 274$ | 100% |
| 4 | $16 \times 5 \times 24.7K$ | 87.0% |
| 5 | $171 \times 15 \times 29.2K$ | 85.4% |

It spots different hierarchical dense blocks with interesting patterns for DARPA (Right).

better results via a local search on the basis of CPD's results. M-Zoom and D-Cube always had better performance than the previous two methods, since they directly detect the densest blocks, but they may have missed some items on the periphery of the densest blocks, resulting in poor performance. Our approach can find more subtle structures within those results and sometimes achieve a better result if such characters are significant, like in DARPA.

For the DARPA dataset, CatchCore also spotted the hierarchical behavior pattern of the `Neptune` attack in $H1$–$H3$, which is composed of the connections at different times for a pair of IPs; Figure 5(b) shows the attack pattern snippet that occurred from 7 a.m. to 8 a.m. on June 18, 1998. The densities (attack intensities) vary greatly over different hierarchies, i.e., the density in $H1$ is about $5K$, while it is only about 3K for the remaining parts in $H3$; the intense attacks represented cyclic patterns in 5 minutes. These attacks had various levels of intrusion intensity and burst activity, resulting in discriminatively dense patterns over different time periods. For the AirForce dataset, although the hierarchical structure of all subtensors includes almost malicious connections (with a recall of 98%) at the cost of containing some false-positive samples, CatchCore achieves comparable performance to baselines, while D-Cube is the best one.

Provided the labeled AirForce dataset, we formalized the anomaly detection here as a binary classification problem (the normal vs. attack) and compared to some neural network models, i.e., the unsupervised outlier/anomaly models, including **AutoEncoder** (**AE**) [Sakurada and Yairi 2014], **Variational AutoEncoder** (**VAE**) [Kingma and Welling 2014], $\beta$-VAE [Burgess et al. 2018], and DeepSVDD [Ruff et al. 2018]; MLP as a supervised classification model. We ignored the comparison of DARPA with any GNN models since few of them can deal with the edge attributes (i.e., timestamps) and mixed edge-labels.

Table 8. Performance of Unsupervised and Supervised Neural Network Models for Anomaly Detection Over AirForce

| | VAE | $\beta$-VAE | DeepSVDD | AutoEncoder | MLP |
|---|---|---|---|---|---|
| F1 score | 0.419 | 0.541 | 0.575 | 0.667 | 0.996 |

We utilized the implementations in the popular Anomaly Detection library PyOD [Zhao et al. 2019] and used 70% data for training and the remaining 30% as the testing set. The parameter settings are listed as follows (the *contamination* parameter is set to be the proportion of the normal to make it in the range [0.0.5], if it is needed).

— VAE: the dimensions of hidden neurons are 32 and 16;
— $\beta$-VAE: coefficient of $\beta$-VAE regime as 2.5 and the maximum capacity of a loss bottleneck as 15.0, others keep the same as VAE;
— AutoEncoder: the dimensions of hidden neurons are 64 and 32;
— DeepSVDD: it trains an AutoEncoder while minimizing the volume of a hypersphere that encloses the network representations of the data, forcing the network to extract the common factors of variation. The dimensions of hidden neurons are the same as the AutoEncoder and with a dropout rate of 0.1;
— MLP: three layers with 32 hidden neurons, the regularization term as $1E-5$.

We optimize by Adam optimizer with the learning rate of $1E-4$ with a batch size of 128.

Table 8 shows the F1 score of those neural network based models. The performances of those unsupervised models are significantly worse than the results of dense subtensor detection methods in Table 7; they are usually sensitive to the model parameters and the variances are also large, e.g., the highest score of $\beta$-VAE is 0.663 and the lowest is just 0.418. Surprisingly, the MLP model achieves the best performance, and even significant gain compared with Table 7 after using some label information. In contrast, dense subtensor detection methods can easily handle both the relational (i.e., graph or multi-correlation tensor) and tabular data in an unsupervised manner, and find some interesting and unexpected behavior patterns, which may be beyond the ability of some deep models.

CatchCore also discerned denser and more anomalous subtensors. For the YouTube dataset in Figure 4(a), these dense blocks are missed by other baselines. In particular, the block with the highest density ($H1$) represents one user who became friends with 904 different users in one day, while the other one in $H2$ also created connections with 475 users at the same time. So, these two users are more likely to be fraudulent or bot accounts. The densest block in the StackOverflow data shows one user marking 257 posts as their favorite in one day, which is also most likely to be an anomaly.

Compared to the baselines, CatchCore detects holistically optimal multi-layer dense subtensors while the densest one is only part of it, which is not our direct target. Besides, the volume density metric tends to be non-empty blocks and may result in some locally dense one-dimensional slices in the highest density layer, which may not be the densest slices within the whole tensor. Using other density metrics could eliminate this issue.

*6.3.2 Evolving Community Pattern.* Figure 1(b) and (c) shows the evolving co-authorship structure of dense subtensors in the top four densest hierarchies for the DBLP dataset; they correspond to the interaction among 20 researchers from 2007 to 2013; Figure 1(c) also presents their densities. The block in $H1$ with a size of $8 \times 8 \times 3$, consists of research cooperation between `Leonard Barolli, Makoto Takizawa, Fatos Xhafa`, and so on. from 2011 to 2013 in the "Algorithm and Distributed System" field and the average connection between them is more than 10.7 in each

Table 9. **Query-specific HDS-tensors Detection Examples**

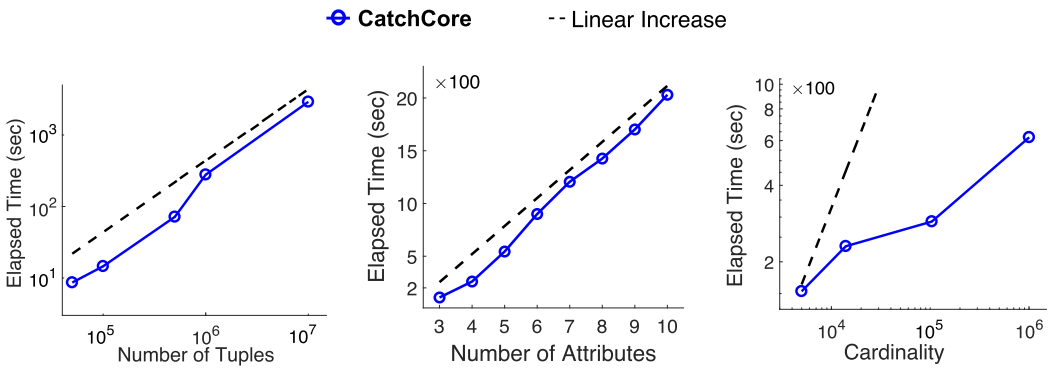
| Query Author | Hs | Volume | Density | *nnz* | Return Items |
|---|---|---|---|---|---|
| Jiawei Han | $H1$ | $7 \times 7 \times 5$ | 2.52 | 132 | { Jing Gao, Jiawei Han, Yizhou Sun, Latifur Khan, Mohammad M. Masud, Murat Kantarcioglu, Bhavani M. Thuraisingham }, { 2008 - 2012 } |
| | $H2$ | $9 \times 9 \times 6$ | 1.79 | 210 | + { Philip S. Yu, Charu C. Aggarwal }, + { 2007 } |
| | $H3$ | $11 \times 11 \times 7$ | 1.39 | 342 | + { Wei Fan, Xifeng Yan }, + { 2013 } |
| | $H4$ | $18 \times 18 \times 7$ | 0.78 | 528 | + { Jiebo Luo, Yun Fu, Thomas S. Huang, Xin Jin, Hong Cheng, Liangliang Cao, Shuicheng Yan } |
| Christos Faloutsos | $H1$ | $8 \times 8 \times 5$ | 1.21 | 158 | { Christos Faloutsos, Lei Li, U. Kang, Leman Akoglu, Hanghang Tong, B. Aditya Prakash, Duen Horng Chau, Tina Eliassi-Rad }, { 2008, 2010 - 2013 } |
| | $H2$ | $16 \times 16 \times 5$ | 0.75 | 358 | + { Wei Fan, Jing Gao, Philip S. Yu, Yizhou Sun, Charu C. Aggarwal, Jimeng Sun, Xifeng Yan } |
| | $H3$ | $19 \times 19 \times 7$ | 0.56 | 536 | + { Hong Cheng, Spiros Papadimitriou, Xin Jin }, + { 2007, 2009} |
| | $H4$ | $70 \times 70 \times 7$ | 0.26 | 2,659 | + { RiChang Hong, Yun Fu, Yue Gao, Deng Cai, Yan Song, Thomas S. Huang, Xiuqing Wu, Xuelong Li, Jingdong Wang, Jiebo Luo, Xindong Wu, Liangliang Cao, Hongjiang Zhang Tat-Seng Chua, Jiajun Bu, Xiaoou Tang, $\cdots$ } |
| Vipin Kumar | $H1$ | $7 \times 7 \times 1$ | 4.24 | 42 | { Vipin Kumar, Shyam Boriah, Michael Steinbach, Varun Mithal, Christopher Potter, Ashish Garg, Steven A. Klooster }, { 2011 } |
| | $H2$ | $11 \times 11 \times 2$ | 1.95 | 170 | + { Juan Carlos Castilla-Rubio, Vikrant Krishna, Yashu Chamber, Ivan Brugere }, + { 2012 } |
| | $H3$ | $17 \times 17 \times 2$ | 1.18 | 342 | + { Anuj Karpatne, Michael Lau, Alok N.Choudhary, J. D. Stanley, Teji Abraham, Karsten Steinhaeuser } |
| | $H4$ | $32 \times 32 \times 3$ | 0.57 | 767 | + { Fredrick H. M. Semazzi, Yu Cheng, Xi C. Chen, Nagiza F. Samatova, William Hendrix, Daniel Honbo, Md. Mostofa Ali Patwary, Stefan Liess, Yusheng Xie, Zhengzhang Chen, Ankit Agrawal, Jay Kawale, Wei-keng Liao, Kunpeng Zhang }, + { 2013 } |

Here we only list the top-4 densest hierarchical dense subtensors resulting from three query authors ("Jiawei Han", "Christos Faloutsos", and "Vipin Kumar") for the DBLP coauthor dataset. The items following "+" denote the added authors and/or years compared to the previous hierarchies.

year, forming a high-density clique. Also, the subtensors in other hierarchies are extended with their other common co-authors and years and contain relatively fewer connections than $H1$, but the density of blocks in $H4$ is also more than 2.0. Therefore, CatchCore can cater to detecting evolving community structures at different scales in a hierarchical fashion.

### 6.4 Q3. Query-specific HDS-tensors Patterns

Considering the item query situation, we performed some case studies of the detected HDS-tensors based on the algorithm described in Section 5.4. Here, we utilized the DBLP coauthor dataset and chose some target authors to find their closely collaborated authors and active time. Considering




(a) Number of Tuples ($nnz(\mathcal{R})$) (b) Number of Attributes ($N$) (c) Cardinalty of Attributes ($V_{\mathcal{R}}$)

Fig. 6. **CatchCore is scalable with all the aspects of tensors.** (a) (b) CatchCore scales linearly with the number of tuples and the number of attributes of $\mathcal{R}$. (c) CatchCore scales sub-linearly with the cardinalities of attributes of $\mathcal{R}$.

three well-known researchers in the Data Mining field, i.e., Jiawei Han, Christos Faloutsos, and Vipin Kumar, we use those specific authors as the seeds[3] and query the hierarchical dense subtensors. Table 9 lists the top 4 densest HDS-tensors query results.

As we can see, for each query item, the sizes and densities of different hierarchies vary greatly, resulting from including different researchers and years of varying intensities of cooperation. These results are also consistent with what we find in Google Scholar and web search. For Jiawei Han, his close collaborators in $H1$ consisted of Ph.D. students he previously supervised, e.g., Jing Gao and Yizhou Sun, and other scholars in similar research fields but from different universities, like Latifur Khan and Bhavani M. Thuraisingham; other two famous scholars, Philip S. Yu and Charu C. Aggarwal, are also included in $H2$. It is worth noting that scholars in different fields, i.e., Multimedia, Computer Vision, Machine Learning, are included in $H4$, such as Jiebo Luo, Thomas S. Huang, and Shuicheng Yan. This reflects the relationship established through the multi-hop collaboration. For Christos Faloutsos, those Ph.D. students he supervised made up his core collaborators in $H1$, e.g., U. Kang, Hanghang Tong, Leman Akoglu, and so on. There are some overlapping with the results from Jiawei Han after $H2$, via some common collaborators, being the bridge connects two groups, including Philip S. Yu, and Xifeng Yan. In contrast, the coauthor network of Vipin Kumar is different from the first two; and the density of $H1$ is higher, that is, 7 researchers around Vipin Kumar published 42 articles in 2011; his active times are also more concentrated, which just in 3 years, 2011–2013.

Therefore, these interesting findings and patterns confirm that the query-specific HDS-tensors detection method can offer an effective way to explore the surrounding relationship of specific objects and provide the possibility to capture anomalies and clues, which is helpful to make full use of the limited prior knowledge.

### 6.5 Q4. Scalability

Empirically, we show that CatchCore scales (sub-) linearly with every aspect of the input, i.e., the number of tuples, the number of dimension attributes, and the cardinality of dimension attributes we aim to find, as mathematically analyzed in Theorem 5.2.

[3]As for the key parameters, including the penalty $p$ and the regularization balance factor $\lambda$, they depend on the local density of the targets, for which we have no prior knowledge; so we choose the proper setting leading to better results. More specifically, $p = 2E-3$, $\lambda = 20$ for Jiawei Han, $p = 1E-3$, $\lambda = 20$ for Christos Faloutsos, and $p = 7E-4$, $\lambda = 1,000$ for Vipin Kumar.

To measure the scalability with each factor of the input, we started by finding the injected subtensor with two hierarchies, i.e., $100 \times 100 \times 2$ with density 0.2 and $50 \times 50 \times 1$ with density 1.0, in a randomly generated tensor $\mathcal{R}$, which contains one million tuples with three attributes whose total cardinality is 100 K. Then, we measured the running time as we changed one of the factors at a time. Specifically, for the scalability of # of tuples, $\mathcal{R}$ has three attributes and a cardinality of $10K$ in different densities, varying from $5E-8$ to $1E-5$. For the # of attributes, we changed the dimension from 3 to 10 for $\mathcal{R}$ with $1M$ tuples and the same cardinality. For the cardinality, the $D_{\mathcal{R}}$ of a 3-way tensor $\mathcal{R}$ with $1M$ tuples varies from $5E3$ to $1E6$.

As we can see in Figure 6, CatchCore scales linearly with the number of tuples and attributes; it also scales sub-linearly with the cardinalities of attributes, which illustrates the complexity analysis in Theorem 5.2 seems too pessimistic.

## 7 CONCLUSIONS

In this article, we propose the CatchCore algorithm to detect the hierarchical dense subtensors with a gradient optimization strategy in large tensors, based on a novel and interpretable uniform framework for dense block detection; and it can also be applied to the query-specific dense subtensor detection scenario. CatchCore accurately detects dense blocks and hierarchical dense subtensors for synthetic and real-world data, and outperforms state-of-the-art baseline methods. It can identify various local dense patterns indicating attack and anomalous behaviors, like periodic attacks and dynamic groups of researchers, and capture core collaboration and multi-hop relations around query items. In addition, CatchCore also scales up linearly in terms of all aspects of the tensor. Our contributions are as follows:

- **Unified metric and algorithm**: We design a unified metric that can be optimized in gradient techniques to detect dense subtensors and propose CatchCore for hierarchical dense subtensor detection with a convergence guarantee. We also propose query-specific dense subtensor detection for some target items.
- **Accuracy**: CatchCore accurately detects dense blocks and hierarchical dense subtensors for synthetic and real-world datasets, and it outperforms the other baseline methods. We designed to measure the quality of detection results in the MDL principle and help select the optimal model.
- **Effectiveness**: CatchCore spotted various anomalous patterns in real-world datasets, like network intrusion and social bots. It also found hierarchical dense research co-authorship groups for the DBLP and captured some local-dense patterns as core collaboration and multi-hop relations of query objects.
- **Scalability**: CatchCore runs linearly in terms of all aspects of tensors, and takes linear space in the number of tuples.

**Reproducibility:** The sourced code and datasets used in this article are available at http://github.com/wenchieh/catchcore.

*Discussion.* In this article, we assume the tensor data can fit in the memory for optimization. Thus, our method cannot be directly applied to the tensors in a streaming format or stored on disk for real applications, such as social media and the web. Exploring other distribution or block optimization techniques while maintaining good convergence properties might be promising for this problem. Another trivial problem found in the experiment is that it requires a larger value of parameter $K$ for detecting dense enough blocks due to the limitations of the gradient-based optimization process and the quality of local optimal. We expect progress in optimization theory and algorithms will help address it. Indeed, our work merely explores some interesting properties of real-world behavioral patterns, relying on researchers' insights and precise definitions. It is

worthy of further study on how to design a way to find or learn similar patterns automatically in complex data.

## REFERENCES


Pieter Adriaans and Paul M. B. Vitanyi. 2009. Approximation of the two-part MDL code. *IEEE Transactions on Information Theory* 55, 1 (2009), 444–457. DOI: https://doi.org/10.1109/TIT.2008.2008152

Leman Akoglu, Rishi Chandy, and Christos Faloutsos. 2013. Opinion fraud detection in online reviews by network effects. In *Proceedings of the 7th International Conference on Weblogs and Social Media*. The AAAI. Retrieved from http://www.aaai.org/ocs/index.php/ICWSM/ICWSM13/paper/view/5981.

Leman Akoglu, Hanghang Tong, and Danai Koutra. 2015. Graph based anomaly detection and description: A survey. *Data Mining and Knowledge Discovery* 29, 3 (2015), 626–688.

Reid Andersen and Kumar Chellapilla. 2009. Finding dense subgraphs with size bounds. In *Proceedings of the Algorithms and Models for the Web-Graph, 6th International Workshop*. Springer, 25–37. DOI: https://doi.org/10.1007/978-3-540-95995-3_3

Yuichi Asahiro, Refael Hassin, and Kazuo Iwama. 2002. Complexity of finding dense subgraphs. *Discrete Applied Mathematics* 121, 1–3 (2002), 15–26.

Oana Denisa Balalau, Francesco Bonchi, T. H. Hubert Chan, Francesco Gullo, and Mauro Sozio. 2015. Finding subgraphs with maximum total density and limited overlap. In *Proceedings of the 8th ACM International Conference on Web Search and Data Mining*. ACM, 379–388. DOI: https://doi.org/10.1145/2684822.2685298

Dimitri P. Bertsekas. 1997. Nonlinear programming. *Journal of the Operational Research Society* 48, 3 (1997), 334–334.

Alex Beutel, Wanhong Xu, Venkatesan Guruswami, Christopher Palow, and Christos Faloutsos. 2013. Copycatch: Stopping group attacks by spotting lockstep behavior in social networks. In *Proceedings of the 22nd International World Wide Web Conference*. International World Wide Web Conferences Steering Committee/ACM, 119–130. DOI: https://doi.org/10.1145/2488388.2488400

Tian Bian, Xi Xiao, Tingyang Xu, Peilin Zhao, Wenbing Huang, Yu Rong, and Junzhou Huang. 2020. Rumor detection on social media with bi-directional graph convolutional networks. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence, AAAI 2020, The 32nd Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The 10th AAAI Symposium on Educational Advances in Artificial Intelligence*. AAAI , 549–556.

Christopher P. Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. 2018. Understanding disentangling in $\beta$-VAE. arXiv:1804.03599. Retrieved from https://arxiv.org/abs/1804.03599.

Paul H. Calamai and Jorge J. Moré. 1987. Projected gradient methods for linearly constrained problems. *Mathematical Programming* 39, 1 (1987), 93–116.

Deepayan Chakrabarti, Spiros Papadimitriou, Dharmendra S. Modha, and Christos Faloutsos. 2004. Fully automatic cross-associations. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 79–88. DOI: https://doi.org/10.1145/1014052.1014064

Moses Charikar. 2000. Greedy approximation algorithms for finding dense components in a graph. In *Proceedings of the 3rd International Workshop on Approximation Algorithms for Combinatorial Optimization*. Springer, 84–95. DOI: https://doi.org/10.1007/3-540-44436-X_10

Jie Chen and Yousef Saad. 2010. Dense subgraph extraction with application to community detection. *IEEE Transactions on Knowledge and Data Engineering* 24, 7 (2010), 1216–1230.

Thomas F. Coleman and Yuying Li. 1996. An interior trust region approach for nonlinear minimization subject to bounds. *SIAM Journal on Optimization* 6, 2 (1996), 418–445.

Kanishka Ghosh Dastidar, Johannes Jurgovsky, Wissam Siblini, Liyun He-Guelton, and Michael Granitzer. 2022. NAG: Neural feature aggregation framework for credit card fraud detection. *Knowledge and Information Systems* 64, 3, 831–858. DOI: https://doi.org/10.1007/s10115-022-01653-0

Quang-Huy Duong, Heri Ramampiaro, and Kjetil Nørvåg. 2020. Multiple dense subtensor estimation with high density guarantee. In *Proceedings of the 36th IEEE International Conference on Data Engineering*. IEEE, 637–648. DOI: https://doi.org/10.1109/ICDE48307.2020.00061

Daniel Edler, Ludvig Bohlin, and Martin Rosvall. 2017. Mapping higher-order network flows in memory and multilayer networks with infomap. *Algorithms* 10, 4 (2017), 112.

Dhivya Eswaran, Stephan Günnemann, Christos Faloutsos, Disha Makhija, and Mohit Kumar. 2017. Zoobp: Belief propagation for heterogeneous networks. *Proceedings of the VLDB Endowment* 10, 5 (2017), 625–636.

Ryan J Gallagher, Jean-Gabriel Young, and Brooke Foucault Welles. 2021. A clarified typology of core-periphery structure in networks. *Science Advances* 7, 12 (2021), eabc9800.

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 *Proceedings of the 21st World Wide Web Conference 2012*. ACM, 61–70. DOI: https://doi.org/10.1145/2187836.2187846

David Gibson, Ravi Kumar, and Andrew Tomkins. 2005. Discovering large dense subgraphs in massive graphs. In *Proceedings of the 31st International Conference on Very Large Data Bases*. ACM, 721–732. Retrieved from http://www.vldb.org/archives/website/2005/program/paper/thu/p721-gibson.pdf.

Andrew V. Goldberg. 1984. *Finding a Maximum Density Subgraph*. University of California at Berkeley.

Alexander Gorovits, Ekta Gujral, Evangelos E Papalexakis, and Petko Bogdanov. 2018. Larc: Learning activity-regularized overlapping communities across time. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1465–1474. DOI: https://doi.org/10.1145/3219819.3220118

Lars Grasedyck. 2010. Hierarchical singular value decomposition of tensors. *SIAM Journal on Matrix Analysis and Applications* 31, 4 (2010), 2029–2054.

Luigi Grippo and Marco Sciandrone. 2000. On the convergence of the block nonlinear gauss–seidel method under convex constraints. *Operations Research Letters* 26, 3 (2000), 127–136.

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 *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 895–904. DOI: https://doi.org/10.1145/2939672.2939747

Dmitry I. Ignatov, Sergei O. Kuznetsov, Jonas Poelmans, and Leonid E. Zhukov. 2013. Can triconcepts become triclusters? *International Journal of General Systems* 42, 6 (2013), 572–593.

Vinay Jethava, Anders Martinsson, Chiranjib Bhattacharyya, and Devdatt Dubhashi. 2013. Lovász $\vartheta$ function, SVMs and finding dense subgraphs. *Journal of Machine Learning Research* 14, 1 (2013), 3495–3536. DOI: https://doi.org/10.5555/2567709.2627669

Meng Jiang, Alex Beutel, Peng Cui, Bryan Hooi, Shiqiang Yang, and Christos Faloutsos. 2015. A general suspiciousness metric for dense blocks in multimodal data. In *Proceedings of the 2015 IEEE International Conference on Data Mining*. IEEE Computer Society, 781–786. DOI: https://doi.org/10.1109/ICDM.2015.61

Samir Khuller and Barna Saha. 2009. On finding dense subgraphs. In *Proceedings of the 36th International Colloquium on Automata, Languages, and Programming, ICALP 2009* . Springer, 597–608. DOI: https://doi.org/10.1007/978-3-642-02927-1_50

Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations (ICLR'14, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings)*, Yoshua Bengio and Yann LeCun (Ed.).

Maksim Kitsak, Lazaros K. Gallos, Shlomo Havlin, Fredrik Liljeros, Lev Muchnik, H. Eugene Stanley, and Hernán A. Makse. 2010. Identification of influential spreaders in complex networks. *Nature Physics* 6, 11 (2010), 888–893.

Jon Kleinberg. 2003. Bursty and hierarchical structure in streams. *Data Mining and Knowledge Discovery* 7, 4 (2003), 373–397.

Tamara G. Kolda and Brett W. Bader. 2009. Tensor decompositions and applications. *SIAM Review* 51, 3 (2009), 455–500.

Olivera Kostoska, Sonja Mitikj, Petar Jovanovski, and Ljupco Kocarev. 2020. Core-periphery structure in sectoral international trade networks: A new approach to an old theory. *PloS One* 15, 4 (2020), e0229547.

Ravi Kumar, Jasmine Novak, and Andrew Tomkins. 2010. Structure and evolution of online social networks. In *Proceedings of the Link Mining: Models, Algorithms, and Applications*. Springer, 337–357. DOI: https://doi.org/10.1007/978-1-4419-6515-8_13

Victor E Lee, Ning Ruan, Ruoming Jin, and Charu Aggarwal. 2010. A survey of algorithms for dense subgraph discovery. In *Proceedings of the Managing and Mining Graph Data*. Springer, 303–336. DOI: https://doi.org/10.1007/978-1-4419-6045-0_10

Jure Leskovec, Kevin J. Lang, Anirban Dasgupta, and Michael W. Mahoney. 2008. Statistical properties of community structure in large social and information networks. In *Proceedings of the 17th International Conference on World Wide Web*. ACM, 695–704. DOI: https://doi.org/10.1145/1367497.1367591

Chih-Jen Lin. 2007. Projected gradient methods for nonnegative matrix factorization. *Neural Computation* 19, 10 (2007), 2756–2779.

Chih-Jen Lin and Jorge J. Moré. 1999. Newton's method for large bound-constrained optimization problems. *SIAM Journal on Optimization* 9, 4 (1999), 1100–1127.

Yang Liu, Xiang Ao, Qiwei Zhong, Jinghua Feng, Jiayu Tang, and Qing He. 2020. Alike and unlike: Resolving class imbalance problem in financial credit risk assessment. In *Proceedings of the CIKM'20: The 29th ACM International Conference on Information and Knowledge Management*. ACM, 2125–2128. DOI: https://doi.org/10.1145/3340531.3412111

Yixin Liu, Shirui Pan, Yu Guang Wang, Fei Xiong, Liang Wang, Qingfeng Chen, and Vincent CS Lee. 2021. Anomaly detection in dynamic graphs via transformer. *IEEE Transactions on Knowledge and Data Engineering* 01 (2021), 1–1.

Xiaoxiao Ma, Jia Wu, Shan Xue, Jian Yang, Chuan Zhou, Quan Z. Sheng, Hui Xiong, and Leman Akoglu. 2021. A comprehensive survey on graph anomaly detection with deep learning. *IEEE Transactions on Knowledge and Data Engineering* (2021), 1–1. DOI: https://doi.org/10.1109/TKDE.2021.3118815

Koji Maruhashi, Fan Guo, and Christos Faloutsos. 2011. Multiaspectforensics: Pattern mining on large-scale heterogeneous networks with tensor analysis. In *Proceedings of the International Conference on Advances in Social Networks Analysis and Mining*. IEEE Computer Society, 203–210. DOI: https://doi.org/10.1109/ASONAM.2011.80

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 *Proceedings of the 16th International Conference on World Wide Web*. ACM, 201–210. DOI: https://doi.org/10.1145/1242572.1242600

Spiros Papadimitriou, Jimeng Sun, Christos Faloutsos, and Philip S. Yu. 2008. Hierarchical, parameter-free community discovery. In *Proceedings of the Machine Learning and Knowledge Discovery in Databases, European Conference*. Springer, 170–187.

Jorma Rissanen. 1978. Modeling by shortest data description. *Automatica* 14, 5 (1978), 465–471.

Jorma Rissanen. 1983. A universal prior for integers and estimation by minimum description length. *The Annals of Statistics* 11, 2 (1983), 416–431.

Ryan A. Rossi, Nesreen K. Ahmed, Eunyee Koh, and Sungchul Kim. 2020. Fast hierarchical graph clustering in linear-time. *Companion Proceedings of the Web Conference (WWW'20)*, Association for Computing Machinery, New York, NY, 10–12. DOI: https://doi.org/10.1145/3366424.3382673

Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. 2018. Deep one-class classification. In *Proceedings of the 35th International Conference on Machine Learning*. PMLR, 4390–4399. Retrieved from http://proceedings.mlr.press/v80/ruff18a.html.

Mayu Sakurada and Takehisa Yairi. 2014. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*. ACM, 4. DOI: https://doi.org/10.1145/2689746.2689747

Ahmet Erdem Sariyüce and Ali Pinar. 2016. Fast hierarchy construction for dense subgraphs. *Proceedings of the VLDB Endowment* 10, 3 (2016), 97–108.

Ahmet Erdem Sariyüce, C. Seshadhri, and Ali Pinar. 2018. Local algorithms for hierarchical dense subgraph discovery. *Proceedings of the VLDB Endowment* 12, 1 (2018), 43–56.

Ahmet Erdem Sariyuce, C. Seshadhri, Ali Pinar, and Umit V. Catalyurek. 2015. Finding the hierarchy of dense subgraphs using nucleus decompositions. In *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 927–937.

Kijung Shin, Bryan Hooi, and Christos Faloutsos. 2016. M-zoom: Fast dense-block detection in tensors with quality guarantees. In *Proceedings of the Machine Learning and Knowledge Discovery in Databases - European Conference*. Springer, 264–280. DOI: https://doi.org/10.1007/978-3-319-46128-1_17

Kijung Shin, Bryan Hooi, and Christos Faloutsos. 2018. Fast, accurate, and flexible algorithms for dense subtensor mining. *ACM Transactions on Knowledge Discovery from Data* 12, 3 (2018), 1–30.

Kijung Shin, Bryan Hooi, Jisu Kim, and Christos Faloutsos. 2017b. D-cube: Dense-block detection in terabyte-scale tensors. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1057–1066. DOI: https://doi.org/10.1145/3097983.3098087

Kijung Shin, Bryan Hooi, Jisu Kim, and Christos Faloutsos. 2017a. Densealert: Incremental dense-subtensor detection in tensor streams. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1057–1066.

Bushra Siddique and Nadeem Akhtar. 2017. Temporal hierarchical event detection of timestamped data. In *Proceedings of the 2017 International Conference on Computing, Communication, and Automation*. 783–788. DOI: https://doi.org/10.1109/CCAA.2017.8229902

Le Song, Mariya Ishteva, Ankur Parikh, Eric Xing, and Haesun Park. 2013. Hierarchical tensor decomposition of latent tree graphical models. In *Proceedings of the 30th International Conference on Machine Learning*. JMLR.org, 334–342. Retrieved from http://proceedings.mlr.press/v28/song13.html.

Charalampos Tsourakakis, Francesco Bonchi, Aristides Gionis, Francesco Gullo, and Maria Tsiarli. 2013. Denser than the densest subgraph: Extracting optimal quasi-cliques with quality guarantees. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 104–112. DOI: https://doi.org/10.1145/2487575.2487645

Daixin Wang, Jianbin Lin, Peng Cui, Quanhui Jia, Zhen Wang, Yanming Fang, Quan Yu, Jun Zhou, Shuang Yang, and Yuan Qi. 2019a. A semi-supervised graph attentive network for financial fraud detection. In *Proceedings of the 2019 IEEE International Conference on Data Mining*. IEEE, 598–607. DOI: https://doi.org/10.1109/ICDM.2019.00070

Jianyu Wang, Rui Wen, Chunming Wu, Yu Huang, and Jian Xion. 2019b. Fdgars: Fraudster detection via graph convolutional networks in online app review system. In *Proceedings of the 2019 World Wide Web Conference*. ACM, 310–316. DOI: https://doi.org/10.1145/3308560.3316586

Serene WH Wong, Chiara Pastrello, Max Kotlyar, Christos Faloutsos, and Igor Jurisica. 2018. Sdregion: Fast spotting of changing communities in biological networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 867–875. DOI: https://doi.org/10.1145/3219819.3219854

Bo Yang, Jin Di, Jiming Liu, and Dayou Liu. 2013. Hierarchical community detection with applications to real-world network analysis. *Data and Knowledge Engineering* 83 (2013), 20–38.

Shuo Yang, Zhiqiang Zhang, Jun Zhou, Yang Wang, Wang Sun, Xingyu Zhong, Yanming Fang, Quan Yu, and Yuan Qi. 2020. Financial risk analysis for SMEs with graph-based supply chain mining. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence, IJCAI 2020*. ijcai.org, 4661–4667. DOI: https://doi.org/10.24963/ijcai.2020/643

Tianbao Yang, Yun Chi, Shenghuo Zhu, Yihong Gong, and Rong Jin. 2011. Detecting communities and their evolutions in dynamic social networks-a bayesian approach. *Machine Learning* 82, 2 (2011), 157–189.

Ban Yikun, Liu Xin, Huang Ling, Duan Yitao, Liu Xue, and Xu Wei. 2019. No place to hide: Catching fraudulent entities in tensors. In *Proceedings of the World Wide Web Conference*. ACM, 83–93. DOI: https://doi.org/10.1145/3308558.3313403

Ge Zhang, Zhao Li, Jiaming Huang, Jia Wu, Chuan Zhou, Jian Yang, and Jianliang Gao. 2022. Efraudcom: An e-commerce fraud detection system via competitive graph neural networks. *ACM Transactions on Information Systems* 40, 3 (2022), 1–29.

Si Zhang, Dawei Zhou, Mehmet Yigit Yildirim, Scott Alcorn, Jingrui He, Hasan Davulcu, and Hanghang Tong. 2017. HiDDen: Hierarchical dense subgraph detection with application to financial fraud detection. In *Proceedings of the 2017 SIAM International Conference on Data Mining*. SIAM, 570–578. DOI: https://doi.org/10.1137/1.9781611974973.64

Tong Zhao, Chuchen Deng, Kaifeng Yu, Tianwen Jiang, Daheng Wang, and Meng Jiang. 2020. Error-bounded graph anomaly loss for GNNs. In *Proceedings of the CIKM'20: The 29th ACM International Conference on Information and Knowledge Management*. ACM, 1873–1882. DOI: https://doi.org/10.1145/3340531.3411979

Yue Zhao, Zain Nasrullah, and Zheng Li. 2019. PyOD: A python toolbox for scalable outlier detection. *Journal of Machine Learning Research* 20, 96 (2019), 1–7. Retrieved from http://jmlr.org/papers/v20/19-011.html.

Received 15 February 2022; revised 12 November 2022; accepted 13 December 2022