CatchCore: Catching Hierarchical Dense Subtensor

Wenjie Feng (✉), Shenghua Liu, and Xueqi Cheng

CAS Key Laboratory of Network Data Science & Technology,
Institute of Computing Technology, Chinese Academy of Sciences, 100190
University of Chinese Academy of Sciences (UCAS), Beijing 100049, China.
fengwenjie@ict.ac.cn, liu.shengh@gmail.com, cxq@ict.ac.cn

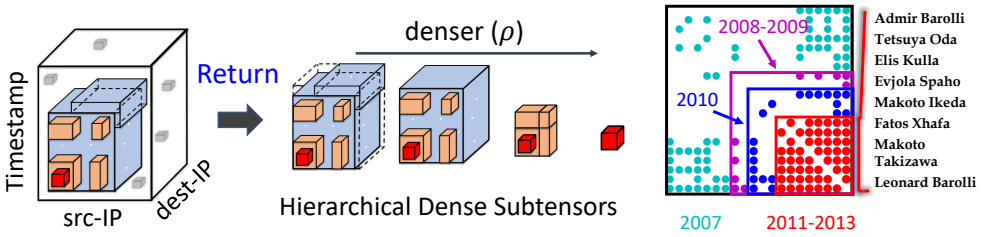
Abstract. Dense subtensor detection gains remarkable success in spotting anomaly and fraudulent behaviors for the multi-aspect data (i.e., tensors), like in social media and event streams. Existing methods detect the densest subtensors flatly and separately, with an underlying assumption that those subtensors are exclusive. However, many real-world tensors usually present hierarchical properties, e.g., the core-periphery structure or dynamic communities in networks. In this paper, we propose CatchCore, a novel framework to effectively 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. Finally, we utilize the minimum description length principle to measure the quality of detection result 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. Additionally, CatchCore successfully identified a hierarchical researcher co-authorship group with intense interactions in DBLP dataset. Meanwhile, CatchCore also scales linearly with all aspects of tensors.
Code related to this paper is available at: http://github.com/wenchieh/catchcore.

1 Introduction

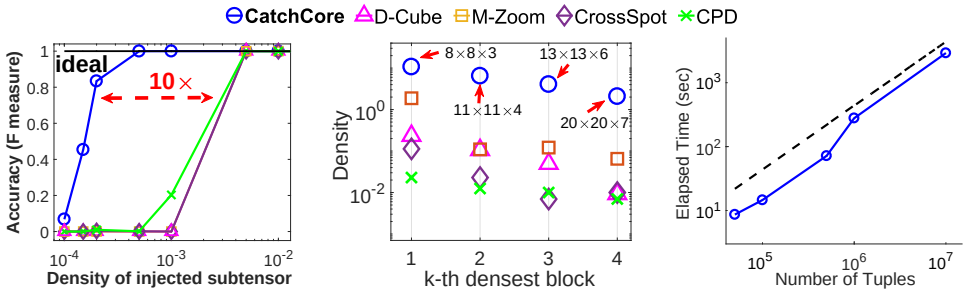
Dense subgraph and subtensor detection have been successfully used in a variety of application domains, like detecting the anomaly or fraudulent patterns (e.g. lockstep behavior, boost ratings) in social media or review sites [11, 13], identifying malicious attacks in network traffic logs or stream data [22, 24], and spotting changing gene-communities in biological networks [27], etc.

Several algorithms detect the densest subtensors or blocks in a flat manner [13, 22, 23, 24], i.e., remove-and-redetect one-by-one, with an underlying assumption that those subtensors are exclusive and separate. However, many real-world tensors usually present hierarchical properties, like the core-peripheral structure in networks and dynamic communities in social media. So it will be difficult to identify subtle structures (like multi-layer core) within the dense block and the relations (e.g., overlapping or inclusion) among different blocks. Meanwhile, other methods for community detection [5, 7, 28] and dense subgraph detection [11, 21, 30] only concentrate on the plain graph.

One challenging problem is how to efficiently detect the hierarchical dense subtensors in the multi-aspect data, and Fig. 1(a) illustrates an example for the TCP dumps scenario. The network intrusion attacks dynamically changed in interacting-intensity



(a) Pictorial depiction of hierarchical dense subtensors. (b) Researcher group in DBLP.



(c) synthetic data result. (d) Top 4 blocks in DBLP. (e) Scalability w.r.t $nnz(\mathcal{R})$.

Fig. 1. Examples and CatchCore performance overview. (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 outperforms competitors for detecting injected blocks in synthetic data, it achieves lower detection bound than others.(d) 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 as Fig. 1(b) shows. (e) CatchCore is linearly scalable w.r.t the number of tuples in tensor.

at different stages along the time and among various hosts (i.e., from `source-IP` to `destination-IP`), resulting in a multi-layer and high-density core. So, hierarchical dense subtensor detection helps to understand the process and spot such anomalies.

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, one can define distinct density measures for dense subtensor detection. The objective based on the metric can be optimized with gradient methods. By such form of metric, CatchCore can identify hierarchical dense cores through a hierarchy-wise alternative optimization with convergence guarantee. In summary, our main contributions are as follows:

- **Unified metric and algorithm**: We design a unified metric can be optimized with the gradient methods to detect dense blocks, propose CatchCore for hierarchical dense core detection with the theoretical guarantee and MDL based measurement.
- **Accuracy**: CatchCore outperforms state-of-the-art methods in accurately detecting dense blocks and hierarchical dense subtensors in both synthetic and real-world datasets (Fig. 1(c), 1(d)).

– **Effectiveness**: CatchCore successfully spotted anomaly patterns, including suspicious friend-connections, periodical network attacks, etc., and found a researcher co-authorship group with heavy interactions in a hierarchical fashion (Fig. 1(b)).
– **Scalability**: CatchCore is scalable, with linear time (Fig. 1(e)) and space complexity with all aspects of tensors (Theorem 7, 8).

Reproducibility: Our open-sourced code and the data we used is available at `http://github.com/wenchieh/catchcore`, where supplementary document is also contained.

2 Notions and Concepts

Throughout the paper, vectors are denoted by boldface lowercases (e.g. $\boldsymbol{x}$), the scalars are denoted by the lowercase letters (e.g. c), and $\lfloor x \rceil \equiv \{1, \ldots, x\}$ for brevity.

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 the non-negative measure attribute $C \in \mathbb{N}^{\geq 0}$, (see the running example in supplement).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 $n \in \lfloor N \rceil$, we use $t[A_n]$ and $t[C]$ to denote the values of A_n and C respectively in t, i.e., $t[A_n] = a_n$ and $t[C] = c$. Thus, the relation $\mathcal{R}$ is actually represented as an N-way tensor of size $|\mathcal{R}_1| \times \cdots \times |\mathcal{R}_N|$, and the value of each entry in the tensor is $t[C]$, where it will be 0 if the corresponding tuple t does not exit. 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 $\mathcal{R}$ as $M_{\mathcal{B}} = \sum_{t \in \mathcal{R}} t[C]$, i.e. the sum of values of measure C 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 denoted as $D_{\mathcal{R}} = \sum_{n=1}^{N} |\mathcal{R}_n|$.

For a subtensor $\mathcal{B}$, which is composed of the subset of attributes in $\mathcal{R}$, is defined as $\mathcal{B} = \{t \in \mathcal{R};\, t[A_n] \in \mathcal{B}_n, \forall n \in \lfloor N \rceil\}$, i.e. the set of tuples where each attribute A_n has a value in $\mathcal{B}_n$. With the tensor term, the relation $\mathcal{B}$ forms a "block" of size $|\mathcal{B}_1| \times \cdots \times |\mathcal{B}_N|$ in $\mathcal{R}$. We use $\mathcal{B} \preccurlyeq \mathcal{R}$ to describe that $\mathcal{B}$ is the subtensor of $\mathcal{R}$. Mathematically, for any $n \in \lfloor N \rceil$, we can use a 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 $\mathcal{B}(n, a_n) \subseteq \mathcal{R}(n, a_n)$. Thus the inclusion relationship between $\mathcal{B}$ and $\mathcal{R}$ can be represented with an indicator vectors collection $\mathbf{X}_{\mathcal{B}} = \{\boldsymbol{x}_n \in \{0,1\}^{|\mathcal{R}_n|}; \forall n \in \lfloor N \rceil\}$. Specially, $\mathbf{X_0} = \{\mathbf{0}^{|\mathcal{R}_n|}; \forall n \in \lfloor N \rceil\}$ corresponds to **NULL** tensor ($\emptyset$), and $\mathbf{X_1} = \{\mathbf{1}^{|\mathcal{R}_n|}; \forall n \in \lfloor N \rceil\}$ corresponds to $\mathcal{R}$.

Given an indicator vector $\boldsymbol{x} \in \{0,1\}^{|\mathcal{R}_n|}$ for 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 the *n-mode product* for a tensor and a vector[1]. In addition, the subtensor $\mathcal{B}$ of $\mathcal{R}$ can be computed with corresponding indicator vectors collection $\mathbf{X}_{\mathcal{B}}$ 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.

3 Framework and Formulation

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

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

3.1 Densest Subtensor Detection Framework.

Let $\mathcal{R}$ is an N-way tensor, and $\mathcal{B}$ is the subtensor of $\mathcal{R}$ defined by the indicator vectors collection $\mathbf{X}_{\mathcal{B}}$. Then the mass $M_{\mathcal{B}}$ can be represented as,

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

where the *full-mode product* $\bar{\times}$ applies the n-mode tensor-vector product to indicator vectors along the corresponding dimension[2]. Inspired by the density function for dense subgraph [26], we propose the following unified metric,

Definition 1 (Entry-Plenum). *Assume* $\mathbf{X}$ *is an indicator vectors collection, and* $\phi > 0$ *is a constant. Given any two strictly increasing functions* g *and* h*, the entry-plenum is defined as:*

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

where the $M_{\mathbf{X}}$ is the mass and $S_{\mathbf{X}}$ is the size of subtensor defined by $\mathbf{X}$ in $\mathcal{R}$, and $S_{\mathbf{X}}$ can be $V_{\mathbf{X}}$, $D_{\mathbf{X}}$ or other forms.

Most popular existing subtensor density measures [13, 22, 23] can be subsumed into the above definition as,

- Let $g(x) = \log x, h(x) = \log \frac{x}{N}, \phi = 1$ and $S_{\mathbf{X}} = D_{\mathbf{X}}$, $f_\phi(\mathcal{B})$ is equal to the *arithmetic average mass* $\rho_{ari}(\mathcal{B}) = M_{\mathcal{B}}/(D_{\mathcal{B}}/N)$.
- Let $g(x) = h(x) = \log x$, $S_{\mathbf{X}} = V_{\mathbf{X}}$, if $\phi = 1$, then $f_\phi(\mathcal{B})$ corresponds to *volume density* $\rho_{vol}(\mathcal{B}) = M_{\mathcal{B}}/V_{\mathcal{B}}$; and if set $\phi = \frac{1}{N}$, the $f_\phi(\mathcal{B})$ comes down to the *geometric average mass* $\rho_{geo}(\mathcal{B}) = M_{\mathcal{B}}/V_{\mathcal{B}}^{1/N}$.

In principle, for the entry-plenum definition, the first term $g(M_{\mathbf{X}})$ favors subtensors with the large mass, whereas the second term $-\phi \cdot h(S_{\mathbf{X}})$ acts as regularization to penalize large-size block. Thus, detecting the densest subtensor can be rewritten as the following problem under the entry-plenum metric.

Problem 2 (Densest (g, h, ϕ)*-entry-plenum Subtensor).* Given an N-way tensor $\mathcal{R}$, a constant $\phi > 0$, and a pair of increasing functions g and h, find an indicator vectors collection $\mathbf{X}^*$ such that $f_\phi(\mathbf{X}^*) \geq f_\phi(\mathbf{X})$ for all feasible $\mathbf{X} = \{\boldsymbol{x}_n \in \{0,1\}^{|\mathcal{R}_n|} : \forall n \in \lfloor N \rceil\}$. The subtensor derived from $\mathbf{X}^*$ is referred to be as the ***Densest*** (g, h, ϕ)***-entry-plenum Subtensor*** of the tensor $\mathcal{R}$.

In general, finding the densest block in terms of some measure is NP-hard [2, 22], infeasible for the large dataset. Existing methods [4, 11, 22, 23] resort to greedy approximation algorithm, which iteratively selects the local optimal subtensor from candidates based on some density measure defined in the ratio form, for scalability. Instead, our framework formulates the densest subtensor detection problem in an optimization perspective as follows, it utilizes the indicator vectors collection $\mathbf{X}$, which can be treated as a block-variable, to make the above problem can be solved through *block-nonlinear Gauss-Seidel (GS) method* [10] with convergence guarantee by introducing relaxation. $M_{\mathbf{X}}$ and $S_{\mathbf{X}}$ are derivable to each indicator vector under this condition, and we can use gradient-based optimization strategy for updating as long as the g and h are differentiable functions (usually satisfied). Moreover, this process is linearly scalable as our proposed CatchCore algorithm does in Section 4.5.

[2] We use $\bar{\times}_{(-n)}$ to denote conducting full-mode product except the n-th mode.

3.2 Hierarchical Dense Subtensor Detection.

In practice, different dense blocks in $\mathcal{R}$ may be overlapping or even inclusive rather than being separate or flat as many dense-block detection methods assumed [13, 22, 23], whereas they can be described by the *hierarchical dense subtensors*. We present the following example and definition to manifest the hierarchical structure in multi-aspect data, Fig. 1(a) gives a pictorial illustration for the Example 3.

Example 3 (Network Intrusion). The DARPA dataset contains 60% records labeled as anomalous (attacks), which mostly occurred in infrequent bursts, and dominant by 9 types of attacks, like `neptune`, `smurf`, and `satan` etc.. These different attacks had various intrusion intensity and burst activities, leading to discriminative dense patterns at various time ranges.

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

Definition 4 (Hierarchical Dense Subtensors (HDS-tensors)). *For any $k \in \lfloor K \rceil$, 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, that is, for some $\eta > 1$, $\rho(\mathcal{B}^k) \geq \eta\rho(\mathcal{B}^{k-1})$.*

ii) **structure:** *subtensors in higher hierarchies are more "close-knit" (multi-layer dense core) $\mathcal{B}^k \preccurlyeq \mathcal{B}^{k-1}$, i.e., $\mathcal{B}^k_n \subseteq \mathcal{B}^{k-1}_n, \forall n \in \lfloor N \rceil$.*

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 since 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 how to design the overall objective function to be optimized by the recursive heuristic is also not clear.

Formally, with the indicator vectors collection $\mathbf{X}^k$ denoting the dense subtensor $\mathcal{B}^k$, the HDS-tensors detection problem is defined as follows.

Problem 5 (HDS-tensors Detection). **Given:** (1) the input N-way tensor $\mathcal{R}$, (2) the expected density ratio between two adjacent hierarchies η[3], (3) the maximum number of hierarchies K. **Find:** the indicator vectors collections $\{\mathbf{X}^1, \ldots, \mathbf{X}^r\}, r \leq K$ for hierarchical dense subtensors in the r significant hierarchies.

We require that $\rho(\mathbf{X}^r) \geq \eta\rho(\mathbf{X}^{r-1}) \geq \cdots \geq \eta^{r-1}\rho(\mathbf{X}^1)$, and $\mathbf{X}^r \preccurlyeq \mathbf{X}^{r-1} \preccurlyeq \cdots \preccurlyeq \mathbf{X}^1$.

In addition, we define a three-level coordinate (k, n, i) to index the indicator vectors collections, 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 $\mathbf{X}^k$ resp..

4 Proposed Method

In this section, we propose CatchCore, which is an optimization based algorithm to detect the HDS-tensors, and provide analysis for the properties of CatchCore.

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

4.1 Optimization Based Dense Subtensor Detection.

Here, we provide an **interpretable instantiation** for the entry-plenum metric based on the volume density. With the indicator vectors collection $\mathbf{X}_{\mathcal{B}}$ of 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 volume $V_{\mathcal{B}}$ is the product of the size of indicator vector for each mode, i.e, $\prod_{\boldsymbol{x}\in\mathbf{X}_{\mathcal{B}}}||\boldsymbol{x}||_1$, which equals to the total number of possible entries (including zeros).

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

Intuitively, we need to **maximize** the mass of entries while **minimize** the mass of missing entries in the block. Therefore, we propose the following optimization goal,

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

where $p > 0$ is the penalty parameter.

The rationale behind above definition is that each existing entry t in the resultant subtensor contributes $t[C]$ as itself to $\mathcal{F}(\mathbf{X})$, while each missing one $\tilde{t}$ is penalized by a value of p (i.e. $\tilde{t}[C] = -p$). In this way, the objective function maximize the total mass in the resultant subtensor while minimizing the total penalty of the missing entries. Moreover, it is also an instantiation of densest (g, h, ϕ)-entry-plenum subtensor by setting $g(x) = h(x) = x, \phi = p/(1+p)$, and $S_{\mathbf{X}} = V_X$.

The optimization of the objective function $\mathcal{F}(\cdot)$ is an *NP-hard* problem due to the combinatorial nature stemming from the binary constraints in Eq.(3). So, we relax these constraints from a 0-1 integer programming (IP) to a polynomial-time solvable linear programming (LP) problem, that is, $\mathbf{0}^{|\mathcal{R}_n|} \le \boldsymbol{x}_n \le \mathbf{1}^{|\mathcal{R}_n|}$. The relaxed constraint represents the probability that the slices $\mathcal{R}(n, a_n)$ belonging to the resultant dense block. Finally, only the attribute value with probability exactly 1 will be selected.

4.2 Hierarchical Dense Subtensors Detection.

Based on the optimization formulation for finding dense subtensor in one hierarchy, intuitively, we maximize the objective function in Eq.(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 aforementioned prerequisites of HDS-tensors. The ***density*** i constraint is represented as $\rho_{\mathbf{X}^{k+1}} \ge \eta\rho_{\mathbf{X}^k}$ for the k^{th} hierarchy with 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 $\mathbf{X}_{(k+1,n,\cdot)} \le \mathbf{X}_{(k,n,\cdot)} \le \mathbf{X}_{(k-1,n,\cdot)}, \forall\, n \in \lfloor N \rceil$. We assume $\mathbf{X}^0 = \mathbf{X_1}$, $\mathbf{X}^{K+1} = \mathbf{X_0}$.

Consequently, the overall optimization formulation is as follows,

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

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

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

where the parameter λ controls the importance of the regularization term.

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

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

4.3 Optimization Algorithms.

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.

Using the programming methods to solve the BMV-NLP optimization problem, the objective in Eq.(5) is a non-convex and higher-order bounded function with respect to each indicator vectors collection $\mathbf{X}_{(k,\cdot,\cdot)}$, which allows us to apply the alternative update method where we fix all variables of other hierarchies as constants except the current collections in each iteration; the similar strategy is also used to update each indicator vector $\mathbf{X}_{(k,n,\cdot)}$ alternatively.

Based on the *structure* constraint, for any dimension $n \in \lfloor N \rceil$, the feasible solution $\mathbf{X}_{(k,n,\cdot)}$ is bounded by the indicator vectors $\mathbf{X}_{(k-1,n,\cdot)}$ and $\mathbf{X}_{(k+1,n,\cdot)}$ of two adjacent hierarchies in the high-dimensional space. we relax these constraints to $\mathbf{0}^{|\mathcal{R}_n|} \leq \mathbf{X}_{(k,n,\cdot)} \leq \mathbf{X}_{(k-1,n,\cdot)}$ 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 $\{\mathbf{0}^{|\mathcal{R}_n|} \leq \mathbf{X}_{(1,n,\cdot)} \leq \mathbf{1}^{|\mathcal{R}_n|}, \forall n \in \lfloor N \rceil\}$ 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 $\{\mathbf{0}^{|\mathcal{R}_n|} \leq \mathbf{X}_{(2,n,\cdot)} \leq \mathbf{X}_{(1,n,\cdot)}, \forall n \in \lfloor N \rceil\}$ and also ignore other constraints. In this way, we can solve the K dense subtensors detection subproblems hierarchy-by-hierarchy. Technically, we adopt trust-region approach[6] to solve each nonlinear box-constrained programming subproblem. We rewrite the optimization

Algorithm 1 CatchCore for HDS-tensors detection

Input: (1) the N-way tensor: $\mathcal{R}$ (2) the maximum number of hierarchies: K
(3) the penalty value for each missing entry: p
(4) the density ratio between two adjacent hierarchies: η
(5) the regularization parameter: λ (6) maximum number of iterations: $t_{\max}$
Output: 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 $t \leq t_{\max}$ **and** Eq. (7) is not satisfied **do** ▷ stop criteria
  ▷ Gauss-Seidel method 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 $(\boldsymbol{x}^k_n)$
    update $\mathbf{X}^k$
  $t \leftarrow t + 1$
while $r \leq K$ **do** ▷ select significant subtensors
  $\mathcal{S} = \{\mathbf{X}_{(r,n,\cdot)}; \max \mathbf{X}_{(r,n,\cdot)} < 1, \forall n \in \lfloor N \rceil\}$
  if $\mathcal{S} \neq \varnothing$ **then**
    break ▷ no significant subtensors for hierarchies $> r$
  else:
    $r \leftarrow r + 1$
return the resultant r indicator vector collections $\{\mathbf{X}^1, \cdots, \mathbf{X}^r\}$.

problem in Eq.(5) as,

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

We use the alternative projected gradient descent[17, 18] method that is simple and efficient to solve the optimization problem. For any dimension n, we denote the gradient of Eq.(1) w.r.t $\boldsymbol{x}_n$ as

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

so the gradient of $f(\cdot)$ w.r.t $\boldsymbol{x}^1_n$ $(\mathbf{X}_{(1,n,\cdot)})$ and $\boldsymbol{x}^k_n$ $(\mathbf{X}_{(k,n,\cdot)}, k \geq 2)$ are

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

where $\mathbf{1}$ is a $|\mathcal{R}_n|$-dimensional all-ones vector (i.e. the same size as $\boldsymbol{x}^1_n$ and $\boldsymbol{x}^k_n$). 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 $\tilde{\boldsymbol{x}}_n = P(\boldsymbol{x}_n - \alpha \nabla_{\boldsymbol{x}_n} f)$ update rule. Here, the operator $P(\cdot)$

projects the vector back to the bounded feasible region (as above mentioned), $-\nabla_{\boldsymbol{x}_n} f$ is the gradient-related search direction, and α is a step size computed e.g. by means of an Armijo step size strategy. In the case of an Armijo's rule line search, a good step size α is chosen until the following condition is satisfied.

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

where $\mathbf{X}^k_{\boldsymbol{x}_n \to \tilde{\boldsymbol{x}}_n}$ means 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 σ $(0 < \sigma < 1)$ is 0.01. Thus we can alternatively update each indicator vector $\boldsymbol{x}_n$ for the current k^{th}-hierarchy, and the details are listed in OneWayOpt algorithm in supplement.

We propose CatchCore algorithm to solve the programming optimization problem in Eq.(6), First, we initialize the indicator vectors collection with rules that the probabilities of selecting the slices $\mathcal{R}(n, a_n)$ (i.e. $\mathbf{X}^{\text{init}}_{(k,n,i)}$) are 0.5 in the 1^{st} hierarchy and 0.01 in other hierarchies. In this way, we can fairly 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 criteria for the bounded-constrained optimization method besides the limitation for total iterations $t_{\max}$.

$$\left|\left|\left\{\nabla^P_{\boldsymbol{x}^k_n} f; \forall n,\, k\right\}\right|\right|_2 \leq \epsilon \left|\left|\left\{\nabla^P_{\mathbf{X}^{\text{init}}_{(k,n,\cdot)}} f; \forall n,\, k\right\}\right|\right|_2, \tag{7}$$

where $\nabla^P_{\boldsymbol{x}^k_n} f$ is the *elementwise projected gradient* defined as (for the i-th element)

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

Then CatchCore calls OneWayOpt to alternatively updating all the indicator vector for each dimension and hierarchy iteratively. In final, we only select these significant subtensors (the top r of K hierarchies) to return (Line $9-14$).

4.4 Parameter Evaluation.

The penalty value p for missing entries controls the resultant lowest density, and the ratio parameter η affects density-diversity and the number of hierarchies in final. Thus, it is a challenging problem to set them appropriately or evaluate the quality of detection result under some parameter configuration, especially in the un-supervised application. We propose to measure the result w.r.t different parameter settings based on the Minimum Description Length (MDL). In the principle manner, we compute the number of bits needed to encode the tensor $\mathcal{R}$ with detected hierarchical dense subtensors for selecting the best model (parameters), achieving the shortest code length. Intuitively, the less missing entries and the more accurate of detecting hierarchies, lead to the fewer bits needed to encode $\mathcal{R}$ in a lossless compression employing the characterization.

For the indicator vector $\mathbf{X}_{(k,n,\cdot)}$ $(k \in \lfloor K \rceil,\ n \in \lfloor N \rceil)$ we can adopt Huffman or arithmetic coding to encode the binary string, which formally can be viewed as a sequence of realizations of a binomial random variable X. Due to $\mathbf{X}_{(k,n,\cdot)} \leq \mathbf{X}_{(k-1,n,\cdot)}$, we only consider the overlapping part $\bar{\boldsymbol{x}}^k_n = \{\mathbf{X}_{(k,n,i)}; \mathbf{X}_{(k-1,n,i)} = 1, \forall i \in \lfloor |\mathcal{R}|_n \rceil\}$

to avoid redundant encoding of 0s. We denote the entropy of indicator vector $\boldsymbol{x}$ as: $H(\boldsymbol{x}) = -\sum_{q\in\{0,1\}} P(X=q)\log P(X=q)$, where $P(X=q) = n_q/\left|\left|\boldsymbol{x}\right|\right|_1$ and n_q is the number of q in $\boldsymbol{x}$. The description length for indicator vectors collection $\mathbf{X}^k$ is[4]:

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

Assume that $\mathbf{X}^{K+1} = \mathbf{X_0}$, For the dense subtensor $\mathcal{B}^k$ defined by $\mathbf{X}^k$, we only need to encode the entries in $\bar{\mathcal{B}}^k = \mathcal{B}^k - \mathcal{B}^{k+1}$ due to $\mathcal{B}^{k+1} \preccurlyeq \mathcal{B}^k$, based on some probability distribution. For the entry $t \in \bar{\mathcal{B}}^k$, specifically, if $t[C] \in \{0,1\}$, t is sampled from binomial distribution; and if $t[C] \in \mathbb{N}^{\geq 0}$, we instead model the data by using the *Poisson* distribution [13] parameterized by the density of $\bar{\mathcal{B}}^k$, i.e. $\rho_{\bar{\mathcal{B}}^k}$. Therefore the code length for encoding $\bar{\mathcal{B}}^k$ is

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

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 $\mathbf{P}$ (like the mean in Poisson).

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

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

$$L(\mathcal{R}; \mathbf{X}^1, \ldots, \mathbf{X}^K) = \log^* K + \sum_{n=1}^{N} \log^* |\mathcal{R}_n| + \sum_{k=1}^{K} L(\mathbf{X}^k) + L(\bar{\mathcal{B}}^k) + L_\epsilon. \tag{8}$$

To get the optimal parameters, we can heuristically conduct a grid search over possible values and pick the configuration that minimizes MDL. We demonstrate that the parameters according to the MDL principle results in optimal quality of detecting HDS-tensors, and the search space is limited.

4.5 Analysis.

In this section, we provide the analysis for the convergence, the time and space complexity of CatchCore. The details of proofs refer to the supplement.

Lemma 6 states the convergence properties of the gradient method for CatchCore.

Lemma 6. CatchCore *algorithm converges to a critical point.*

Theorem 7 states the time complexity of CatchCore algorithm, which is linear with K, N, and $nnz(\mathcal{R})$ — the number of non-zero entries in $\mathcal{R}$. And the space complexity is given in Theorem 8.

Theorem 7 (Worst-case Time Complexity). *Let* t_{als} *be the number of iterations for Armoji's line search used in the* OneWayOpt *Algorithm for updating any indicator vector, the worst-case time complexity of the* CatchCore *Algorithm 1 is* $O(K \cdot N \cdot t_{\max} \cdot t_{\text{als}} \cdot (nnz(\mathcal{R}) + c \cdot D_{\mathcal{R}}))$.

[4] $\log^* x$ is the universal code length for an integer x.[20].

Table 1. Summary of real-world datasets.

Name	Size	$\lvert card(\mathcal{R})\rvert$	$\mathbf{nnz}(\mathcal{R})$
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: duration × protocol × service × ··· → #connections			
AirForce	3 × 70 × 11 × 7.20K × 21.5K × 512 × 512	39.7K	863K

Theorem 8 (Memory Requirements). *The amount of memory space required by* CatchCore *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 measure attributes of $\mathcal{R}$. In practice, we have following observations, which ensure the efficiency of CatchCore,

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

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

5 Experiments

We design experiments to answer the following questions:

- **Q1. Accuracy**: How accurately does CatchCore detect HDS-tensors in synthetic and real datasets? Does the MDL evaluation select the optimal parameters?
- **Q2. Pattern and anomaly detection**: What patterns does CatchCore detect in real-world data? What is behavior of the detected anomalies?
- **Q3. Scalability**: Does the CatchCore scale linearly with all aspects of data?

5.1 Experimental settings.

Baselines: We selected several state-of-the-art methods for dense-block detection as the baselines, including D-Cube [23], M-Zoom [22], CrossSpot [13], and CP Decomposition (CPD) [14]. In all experiments, a sparse tensor format was used for efficient memory usage, and the ρ_{ari} and ρ_{geo} were used for D-Cube and M-Zoom; we used a variant of CrossSpot which maximizes the same density metrics and used the CPD result for seed selection as did in [22].

Data: Table 1 summarizes the real-world datasets in our experiments. In *Rating* category, data are 4-way tensors *(user ,item, timestamp, rating)*, where entry values are

Table 2. Hierarchical subtensors detection results 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					
	$0.2 + 0.1$	**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			–		
	$1 + 0.05$	**1**	0	0.784	0.784	**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: $100 \times 80 \times 1$.

the number of reviews. In *Social network*, data are 3-way tensors *(user, user, timestamps)*, where entry values are the number of interactions (co-authorship / favorite). *DARPA* is the TCP dumps represented with a 3-way tensor *(source IP, destination IP, timestamps)*, where entry values are the number of connections between two IP addresses (hosts). *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, #connections)*. Timestamps are in minutes for DARPA, in dates for Ratings and Youtube, and in years for DBLP.

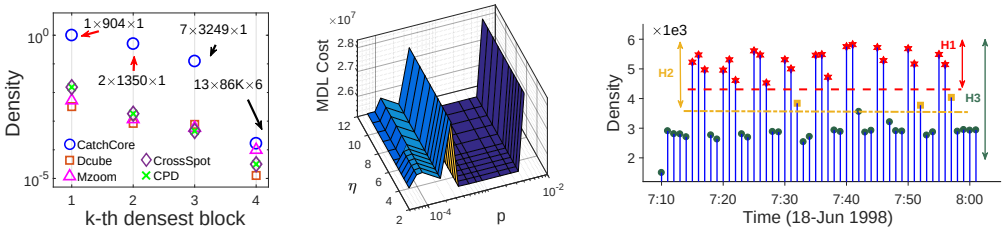
5.2 Q1. Accuracy.

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

We randomly and uniformly generated a $5K \times 5K \times 2K$ 3-way tensor $\boldsymbol{\mathcal{R}}$ with a density of $3 \cdot 10^{-6}$. Into $\boldsymbol{\mathcal{R}}$, one $200 \times 300 \times 100$ block is injected with distinct density, for testing the detection bound of each method. Fig. 1(c) demonstrated that CatchCore effectively detects block as low as a tenth of the density that the best baselines detect, which means that our method can spot such fraudsters with more adversarial effort.

We then injected K subtensors with different size and density into BeerAdvocate and synthetic tensor $\boldsymbol{\mathcal{R}}$ in a hierarchical manner. Table 2 lists the result for former (the result for synthetic data is listed in the supplement), where H1 is the densest or the first subtensor detected by methods, and the density (order) decreases (increases) from H1 to H3, and the information of injected blocks is listed at the bottom. CatchCore can accurately detect all injected subtensors in various hierarchies, size and density diversity, and consistently outperforms other baselines which fail to accurately detect or even miss at least one block. D-Cube and M-Zoom have similar accuracy (except some special cases as highlighted), they can not identify the structure of dense blocks, leading to some of the sparsest or densest injected blocks are missed or overwhelmed by large-volume result. CrossSpot and CPD also do not find hierarchical dense subtensors accurately. Similar conclusions can be drawn for the synthetic dataset.

Moreover, CatchCore can also accurately detect non-overlapping blocks and catch higher density subtensors for the dense blocks with some blanks (or hollows). These cases are verified by different injection schema and results are omitted for space.



(a) Top 4 densest blocks in Youtube

(b) MDL Cost w.r.t parameter p and η

(c) Network Attack Pattern

Fig. 2. (a) CatchCore detects higher-density blocks, which contain suspicious behaviors that users in the top 2 blocks create abnormal large friendship with others within an hour, and outperforms other competitors. (d) The optimal hierarchies achieve the lowest MDL cost w.r.t the parameter p and η. CatchCore can obtain optimal hierarchical dense subtensors for a wide parameters range to some extend. (c) Hierarchical network intrusion behavior between a pair of IPs on June 18, 1998 in DARPA dataset.

Dense blocks in real data: We apply CatchCore for various real-world datasets, and measure the density instead of the mass to avoid the trivial results since that the blocks with higher density contain interesting patterns w.h.p. Fig. 1(d), 2(a) only show the densities of top four densest blocks found by the methods for DBLP and Youtube dataset. CatchCore spots denser blocks for each data, where it consistently outperforms the competitors for all blocks, whose patterns are analyzed in Section 5.3.

MDL-based Evaluation We evaluate the utility of our MDL criterion for measuring the quality of hierarchies returned by CatchCore under different parameter configurations. In this experiment, the BeerAdvocate data with 3 hierarchical injected dense blocks (the second case in Table 2 when $K = 3$) is used, we computed the MDL cost for the detection result by varying a pair of parameters p and η, the result is shown in Fig. 2. The optimal hierarchies achieve the lowest MDL cost w.r.t p and η. In addition, our model can obtain optimal results for a wide range of parameters.

5.3 Q2. Pattern and anomaly detection.

Anomaly detection. CatchCore detected network intrusion in TCP dumps with high accuracy and identified attack patterns for DARPA and AirForce dataset, where each intrusion connection are labeled. Table 3 compares the accuracy of each method. CatchCore outperformed competitors for DARPA data, and also spotted the hierarchical behavior pattern of `Neptune` attack in H1 - H3, which are composed of the connections in different time for a pair of IPs. Fig. 2(c) shows the attack pattern snippet occurred during 7am - 8am on June 18, 1998. The densities (attack intensity) vary greatly over different hierarchies, i.e. the density in H1 is about 5K, while it is only about 3K for remain parts in H3. And the intense attacks represented cyclic patterns in 5 minutes. Although, the hierarchical structure of all subtensors include almost malicious connections (with recall = 98%) with the cost of containing some false positive samples, CatchCore achieves comparable performance for AirForce dataset.

CatchCore also discerned denser and more anomaly subtensors. For the Youtube dataset in Fig. 2(a), these dense blocks are missed by other competitors. Especially, the block with highest-density (H1) represents one user became friend with 904 different

Table 3. CatchCore identifies network attacks from the real-world TCP Dumps dataset with best or comparable performance in F-measure (Left); it spots different hierarchical dense blocks with interesting patterns for DARPA (Right).

	DARPA	AirForce
CPD	0.167	0.785
CrossSpot	0.822	0.785
M-Zoom	0.826	0.906
D-Cube	0.856	**0.940**
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%

users in one day, the other user in H2 also created connections with 475 users at the same time. So, these two users more likely a fraudulent or bot accounts. The densest block in StackOverflow data shows one user marked 257 posts as the favorite in one day, which is too much than the normality. CatchCore detects holistically optimal multi-layers dense subtensors and the densest one is only part of it rather than our direct target. The volume density metric tends to non-empty blocks and may result in some locally 1D slices (may not the densest slices within the whole tensor) in the highest-density layer. Using other density metrics could eliminate this issue.

Evolving community pattern. As the Fig. 1(d), 1(b) show the evolving co-authorship structure of dense subtensors in the top 4 densest hierarchies for DBLP dataset, corresponding to the interaction between 20 researchers during 2007 to 2013, and Fig. 1(d) also presents their densities. The block in H1 with the size $8 \times 8 \times 3$, consists of research cooperation between `Leonard Barolli, Makoto Takizawa`, and `Fatos Xhafa`, etc. during 2011 to 2013 in 'Algorithm and Distributed System' field and the average connection between them is more than 10.7 in each 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 less connections than H1, but the density of blocks in H4 is also more than 2. Therefore, CatchCore can cater to detect evolving community structures at different scales in a hierarchical fashion.

5.4 Q3. Scalability.

Empirically, we show that CatchCore scales (sub-) linearly with every aspect of input, i.e., the number of tuples, the number of dimension attributes, and the cardinality of dimension attributes we aim to find. To measure the scalability with each factor, we started with finding the injected subtensor with two hierarchies, which are $100 \times 100 \times 2$ with density 0.2 and $50 \times 50 \times 1$ with density 1.0, in a randomly generated tensor $\mathcal{R}$ which contains 1 millions tuples with three attributes whose total cardinality is 100K. Then, we measured the running time as we changed one factor at a time. As seen in Fig. 1(e) and result in supplement, CatchCore scales linearly with the number of tuples and the number of attributes, it also scales sub-linearly with the cardinalities of attributes, which illustrates the complexity of CatchCore in Theorem 7 is too pessimistic.

6 Related work

Dense Subgraph / Subtensor Detection in Tensor. The detection of dense-subgraph has been extensively studied [4, 8, 12], and many algorithms for the NP-hard

problem are applied to detect community structure [3, 5, 30] and anomaly [1, 11], or extend to multi-aspect data [22, 23]. CrossSpot [13] finds suspicious dense blocks by adjusting the seed in a greedy way until converges to a local optimum. Tensor decomposition such as HOSVD and CP decomposition [14] are also used to spot dense subtensors. M-Zoom [22] and D-Cube [23] adopt greedy approximation algorithms with quality guarantees to detect dense-subtensors for large-scale tensors. [24] spots dense subtensors for tensor stream with an incremental algorithm. None of these methods consider the relationship and structures of different blocks, and can not trace the evolving of dense subtensors or the hierarchical patterns.

Hierarchical Patterns Mining. Communities exit ubiquitously in various graphs [5, 16], their evolving behavior and hierarchical structure also have been explored in different scenes [15, 19, 28, 29]. [9] proposed a framework for joint learning the overlapping structure and activity period of communities. [25] detected video event with hierarchical temporal association mining mechanism for multimedia applications. HiDDen [30] detects hierarchical dense patterns on graph and also finds financial frauds. [21] uses k-core decomposition to compute the hierarchy of dense subgraphs given by peeling algorithm. Our method can deal with multi-aspect data, provide an information-theoretical measurement for the result, and advanced analyze for the performance.

Anomaly and Fraudulent Detection. The survey [1] provides a structured overview and summary for the methods of detection anomaly in graphs. The dense-subgraphs or dense-subtensors usually contain suspicious patterns, such as fraudsters in social network [11, 27, 30], port-scanning activities in network analysis [13, 23], and lockstep behaviors or vandalism [13, 23, 24].

7 Conclusions

In this work, we propose CatchCore algorithm to detect the hierarchical dense subtensors with gradient optimization strategy, based on an novel and interpretable uniform framework for dense block detection, in large tensor. CatchCore accurately detects dense blocks and hierarchical dense subtensors for the synthetic and real data, and outperforms state-of-the-art baseline methods, it can identify anomaly behaviors and interesting patterns, like periodic attack and dynamic researcher group. In addition, CatchCore also scales up linearly in term of all aspects of tensor.

Acknowledgments

This material is based upon work supported by the Strategic Priority Research Program of CAS (XDA19020400), NSF of China (61772498, 61425016, 61872206), and the Beijing NSF (4172059).

References

1. Akoglu, L., Tong, H., Koutra, D.: Graph based anomaly detection and description: a survey. Data mining and knowledge discovery (2015)
2. Andersen, R., Chellapilla, K.: Finding dense subgraphs with size bounds. In: WAW'09
3. Balalau, O.D., Bonchi, F., Chan, T.H.H., Gullo, F., Sozio, M.: Finding subgraphs with maximum total density and limited overlap. In: WSDM'15 (2015)

4. Charikar, M.: Greedy approximation algorithms for finding dense components in a graph. In: APPROX'00 (2000)
5. Chen, J., Saad, Y.: Dense subgraph extraction with application to community detection. IEEE Trans. on knowledge and Data Engineering (2010)
6. Coleman, T.F., Li, Y.: An interior trust region approach for nonlinear minimization subject to bounds. SIAM Journal on optimization (1996)
7. Edler, D., Bohlin, L., et al.: Mapping higher-order network flows in memory and multilayer networks with infomap. Algorithms **10**(4), 112 (2017)
8. Gibson, D., Kumar, R., Tomkins, A.: Discovering large dense subgraphs in massive graphs. In: VLDB'05. VLDB Endowment (2005)
9. Gorovits, A., Gujral, E., Papalexakis, E.E., Bogdanov, P.: Larc: Learning activity-regularized overlapping communities across time. In: SIGKDD'18. ACM (2018)
10. Grippo, L., Sciandrone, M.: On the convergence of the block nonlinear gauss-seidel method under convex constraints. Oper. Res. Lett. (2000)
11. Hooi, B., Song, H.A., Beutel, A., Shah, N., Shin, K., Faloutsos, C.: Fraudar: Bounding graph fraud in the face of camouflage. In: SIGKDD'16. pp. 895–904 (2016)
12. Jethava, V., Martinsson, A., Bhattacharyya, C., Dubhashi, D.: Lovász ϑ function, svms and finding dense subgraphs. The Journal of Machine Learning Research (2013)
13. Jiang, M., Beutel, A., Cui, P., Hooi, B., Yang, S., Faloutsos, C.: A general suspiciousness metric for dense blocks in multimodal data. In: ICDM'15 (2015)
14. Kolda, T.G., Bader, B.W.: Tensor decompositions and applications. SIAM review (2009)
15. Kumar, R., Novak, J., Tomkins, A.: Structure and evolution of online social networks. In: Link mining: models, algorithms, and applications (2010)
16. Leskovec, J., Lang, K.J., Dasgupta, A., Mahoney, M.W.: Statistical properties of community structure in large social and information networks. In: WWW '08 (2008)
17. Lin, C.J.: Projected gradient methods for nonnegative matrix factorization (2007)
18. Lin, C.J., Moré, J.J.: Newton's method for large bound-constrained optimization problems. SIAM J. on Optimization (1999)
19. Papadimitriou, S., Sun, J., Faloutsos, C., Philip, S.Y.: Hierarchical, parameter-free community discovery. In: ECML-PKDD'08. Springer (2008)
20. Rissanen, J.: A universal prior for integers and estimation by minimum description length. The Annals of Statistics (1983)
21. Sariyüce, A.E., Pinar, A.: Fast hierarchy construction for dense subgraphs. VLDB'16
22. Shin, K., Hooi, B., Faloutsos, C.: M-zoom: Fast dense-block detection in tensors with quality guarantees. In: ECML-PKDD'16 (2016)
23. Shin, K., Hooi, B., Kim, J., Faloutsos, C.: D-cube: Dense-block detection in terabyte-scale tensors. In: WSDM'17. ACM (2017)
24. Shin, K., Hooi, B., Kim, J., Faloutsos, C.: Densealert: Incremental dense-subtensor detection in tensor streams (2017)
25. Siddique, B., Akhtar, N.: Temporal hierarchical event detection of timestamped data. ICCCA'17 (2017)
26. Tsourakakis, C., Bonchi, F., Gionis, A., Gullo, F., Tsiarli, M.: Denser than the densest subgraph: extracting optimal quasi-cliques with quality guarantees. In: SIGKDD'13. ACM
27. Wong, S.W., Pastrello, C., Kotlyar, M., Faloutsos, C., Jurisica, I.: Sdregion: Fast spotting of changing communities in biological networks. In: SIGKDD'18 (2018)
28. Yang, B., Di, J., Liu, J., Liu, D.: Hierarchical community detection with applications to real-world network analysis. DKE (2013)
29. Yang, T., Chi, Y., Zhu, S., Gong, Y., Jin, R.: Detecting communities and their evolutions in dynamic social networks–a bayesian approach. Mach. Learn. (2011)
30. Zhang, S., Zhou, D., Yildirim, M.Y., Alcorn, S., He, J., Davulcu, H., Tong, H.: Hidden: hierarchical dense subgraph detection with application to financial fraud detection. In: SDM'17. SIAM (2017)