M-Zoom: Fast Dense-Block Detection in Tensors with Quality Guarantees

Kijung Shin(✉), Bryan Hooi, and Christos Faloutsos

School of Computer Science, Carnegie Mellon University
kijungs@cs.cmu.edu,bhooi@andrew.cmu.edu,christos@cs.cmu.edu

Abstract. Given a large-scale and high-order tensor, how can we find dense blocks in it? Can we find them in near-linear time but with a quality guarantee? Extensive previous work has shown that dense blocks in tensors as well as graphs indicate anomalous or fraudulent behavior (e.g., lockstep behavior in social networks). However, available methods for detecting such dense blocks are not satisfactory in terms of speed, accuracy, or flexibility. In this work, we propose M-Zoom, a flexible framework for finding dense blocks in tensors, which works with a broad class of density measures. M-Zoom has the following properties: (1) **Scalable**: M-Zoom scales linearly with all aspects of tensors and is up to **114× faster** than state-of-the-art methods with similar accuracy. (2) **Provably accurate**: M-Zoom provides a guarantee on the lowest density of the blocks it finds. (3) **Flexible**: M-Zoom supports multi-block detection and size bounds as well as diverse density measures. (4) **Effective**: M-Zoom successfully detected edit wars and bot activities in Wikipedia, and spotted network attacks from a TCP dump with near-perfect accuracy (**AUC=0.98**).

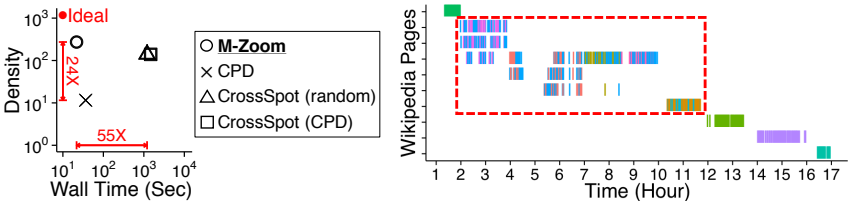
Keywords: Dense-Block Detection, Anomaly/Fraud Detection, Tensor

1 Introduction

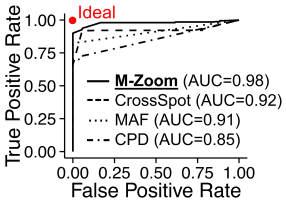
Imagine that you manage a social review site (e.g., Yelp) and have the records of which accounts wrote reviews for which restaurants. How do you detect suspicious lockstep behavior: for example, a set of accounts which give fake reviews to the same set of restaurants? What about the case where additional information is present, such as the timestamp of each review, or the keywords in each review?

Such problems of detecting suspicious lockstep behavior have been extensively studied from the perspective of dense subgraph detection. Intuitively, in the above example, highly synchronized behavior induces dense subgraphs in the bipartite review graph of accounts and restaurants. Indeed, methods which detect dense subgraphs have been successfully used to spot fraud in settings ranging from social networks [5,10,13,14], auctions [20], and search engines [8].

Additional information helps identify suspicious lockstep behavior. In the above example, the fact that reviews forming a dense subgraph were also written at about the same time, with the same keywords and number of stars, makes the reviews even more suspicious. A natural and effective way to incorporate such extra information is to model data as a tensor and find dense blocks in it [12,19].



(a) Performance Comparison (c) Edit Wars in Korean Wikipdia



(b) Intrusion Detection

(d) Bot Activities in English Wikipedia

Users	Pages
HBC AIV helperbot3	DeltaQuad/UAA/Time
HBC AIV helperbot5	DeltaQuad/UAA/Wait
HBC AIV helperbot7	Open proxy detection
DeltaQuadBot	WikiProject Spam/LinkReports
COIBot	COIBot/LinkReports
West.andrew.g	West.andrew.g/Dead links
RonaldB	Cyde/List of current proposed deletions
Cydebot	Cyde/List of requests for unblock

Fig. 1: **M-Zoom is fast, accurate, and effective. Fast:** (a) M-Zoom was 55× faster with denser blocks than CrossSpot in Korean Wikipedia Dataset. **Accurate:** (a) M-Zoom found 24× denser blocks than CPD. (b) M-Zoom identified network attacks with near-perfect accuracy (AUC=0.98). **Effective:** (c) M-Zoom spotted edit wars, during which many users (distinguished by colors) edited the same set of pages hundreds of times within several hours. (d) M-Zoom spotted bots, and pages edited hundreds of thousands of times by the bots.

However, neither existing methods for detecting dense blocks in tensors nor simple extensions of graph-based methods are satisfactory in terms of speed, accuracy, or flexibility. Especially, the types of fraud detectable by each of the methods are limited since, explicitly or implicitly, each method is based on only one density metric, which decides how dense and thus suspicious each block is.

Hence, in this work, we propose M-Zoom (Multidimensional Zoom), a general and flexible framework for detecting dense blocks in tensors. M-Zoom allows for a broad class of density metrics, in addition to having the following strengths:

- **Scalable:** M-Zoom is up to **114× faster** than state-of-the-art methods with similar accuracy (Figure 2) thanks to its linear scalability with all aspects of tensors (Figure 4).
- **Provably accurate:** M-Zoom provides a guarantee on the lowest density of blocks it finds (Theorem 4), as well as shows high accuracy similar with state-of-the-art methods in real-world datasets (Figure 1a).
- **Flexible:** M-Zoom works successfully with high-order tensors and supports various density measures, multi-block detection, and size bounds (Table 1).
- **Effective:** M-Zoom successfully detected edit wars and bot activities in Wikipedia (Figures 1c and 1d), and also detected network attacks with near-perfect accuracy (**AUC=0.98**) based on TCP dump data (Figure 1b).

Table 1: **M-Zoom is flexible.** Comparison between M-Zoom and other methods for dense-block detection. ✓represents 'supported'.

		M-Zoom	CrossSpot [12]	CPD [17]	Subgraph [16]
Data	Matrix	✓	✓	✓	✓
	Tensor	✓	✓	✓	
Density Measure	Average Mass (ρ_{ari})	✓			✓
	Average Mass (ρ_{geo})	✓			✓
	Suspiciousness	✓	✓		
Features	Accuracy Guarantee	✓			✓
	Multiple Blocks	✓	✓	✓	
	Size Bounds	✓			✓

Reproducibility: Our open-sourced code and the data we used are available at `http://www.cs.cmu.edu/~kijungs/codes/mzoom`.

Section 2 presents preliminaries and problem definitions. Our proposed M-Zoom is described in Section 3 followed by experimental results in Section 4. After discussing related work in Section 5, we draw conclusions in Section 6.

2 Preliminaries and Problem Definition

In this section, we introduce definitions and notations used in the paper. We also discuss density measures and give a formal definition of our problems.

2.1 Definitions and Notations

Let $\mathcal{R}(A_1, A_2, ..., A_N, X)$ be a relation with N dimension attributes A_1, A_2, ..., A_N, and a nonnegative measure attribute X (see the supplementary document [1] for a running example and its pictorial description). We use $\mathcal{R}_n$ to denote the set of distinct values of A_n in $\mathcal{R}$, and use $a_n \in \mathcal{R}_n$ for a value of A_n. The value of A_n in tuple t is denoted by $t[A_n]$, and the value of X is denoted by $t[X]$. The relation $\mathcal{R}$ can be represented as an N-way tensor. In the tensor, each n-th mode has length $|\mathcal{R}_n|$, and each cell has the value of attribute X, if the corresponding tuple exists, and 0 otherwise. Let $\mathcal{B}_n$ be a subset of $\mathcal{R}_n$. Then, we define a *block* $\mathcal{B}(A_1, A_2, ..., A_N, X) = \{t \in \mathcal{R} : 1 \leq \forall n \leq N, t[A_n] \in \mathcal{B}_n\}$, the set of tuples where each dimension attribute A_n has a value in $\mathcal{B}_n$. $\mathcal{B}$ is called 'block' because it forms a subtensor where each n-th mode has length $|\mathcal{B}_n|$ in the tensor representation of $\mathcal{R}$. The set of tuples of $\mathcal{R}$ with attribute $A_n = a_n$ is denoted by $\mathcal{R}(a_n) = \{t \in \mathcal{R} : t[A_n] = a_n\}$. We define the mass of $\mathcal{R}$ as $M_{\mathcal{R}} = Mass(\mathcal{R}) = \sum_{t \in \mathcal{R}} t[X]$, the sum of the values of attribute X in $\mathcal{R}$. We also define the size of $\mathcal{R}$ as $S_{\mathcal{R}} = Size(\mathcal{R}) = \sum_{n=1}^{N} |\mathcal{R}_n|$ and the volume of $\mathcal{R}$ as $V_{\mathcal{R}} = Volume(\mathcal{R}) = \prod_{n=1}^{N} |\mathcal{R}_n|$. Lastly, we use $[x] = \{1, 2..., x\}$ for convenience. Table 2 lists frequently used symbols.

Table 2: Table of symbols.

Symbol	Definition
$\mathcal{R}(A_1, A_2, ..., A_N, X)$	a relation with N dimension attributes and a measure attribute
N	the number of dimension attributes in a relation
a_n	a value of attribute A_n
$\mathcal{R}_n$	the set of distinct values of attribute A_n in $\mathcal{R}$
$t[A_n]$ (or $t[X]$)	the value of attribute A_n (or X) in tuple t
$\mathcal{R}(a_n)$	the set of tuples with attribute $A_n = a_n$ in $\mathcal{R}$
$M_{\mathcal{R}}$ (or $Mass(\mathcal{R})$)	the mass of $\mathcal{R}$.
$S_{\mathcal{R}}$ (or $Size(\mathcal{R})$)	the size of $\mathcal{R}$.
$V_{\mathcal{R}}$ (or $Volume(\mathcal{R})$)	the volume of $\mathcal{R}$
$\rho(\mathcal{B}, \mathcal{R})$	density of block $\mathcal{B}$ in $\mathcal{R}$
k	the number of blocks we aim to find
$[x]$	$\{1, 2..., x\}$

2.2 Density Measures

In this paper, we consider three specific density measures although our method is not restricted to them. Two of the density measures (Definitions 1 and 2) are natural multi-dimensional extensions of classic density measures which have been widely used for subgraphs. The merits of the original measures are discussed in [7,15], and extensive research based on them is discussed in Section 5.

Definition 1 (Arithmetic Average Mass [7]). *The arithmetic average mass of a block $\mathcal{B}$ of a relation $\mathcal{R}$ is defined as $\rho_{ari}(\mathcal{B}, \mathcal{R}) = M_{\mathcal{B}}/(S_{\mathcal{B}}/N)$.*

Definition 2 (Geometric Average Mass [7]). *The geometric average mass of a block $\mathcal{B}$ of a relation $\mathcal{R}$ is defined as $\rho_{geo}(\mathcal{B}, \mathcal{R}) = M_{\mathcal{B}}/V_{\mathcal{B}}^{(1/N)}$.*

The other density measure (Definition 3) is the negative log likelihood of $M_{\mathcal{B}}$ on the assumption that the value on each cell (in the tensor representation) of $\mathcal{R}$ follows a Poisson distribution. This proved useful in fraud detection [12].

Definition 3 (Suspiciousness [12]). *The suspiciousness of a block $\mathcal{B}$ of a relation $\mathcal{R}$ is defined as $\rho_{susp}(\mathcal{B}, \mathcal{R}) = M_{\mathcal{B}}(\log(M_{\mathcal{B}}/M_{\mathcal{R}}) - 1) + M_{\mathcal{R}}V_{\mathcal{B}}/V_{\mathcal{R}} - M_{\mathcal{B}} \log(V_{\mathcal{B}}/V_{\mathcal{R}})$.*

Our method, however, is not restricted to the three measures mentioned above. Our method, which searches for dense blocks in a tensor, allows for any density measure ρ that satisfies Axiom 1.

Axiom 1 (Density Axiom)**.** *If two blocks of a relation have the same cardinality for every dimension attribute, the block with higher or equal mass is at least as dense as the other. Formally,*

$$M_{\mathcal{B}} \geq M_{\mathcal{B}'} \text{ and } |\mathcal{B}_n| = |\mathcal{B}'_n|, \forall n \in [N] \Rightarrow \rho(\mathcal{B}, \mathcal{R}) \geq \rho(\mathcal{B}', \mathcal{R}).$$

2.3 Problem Definition

We formally define the problem of detecting the k densest blocks in a tensor.

Problem 1 (k-Densest Blocks). **(1) Given:** a relation $\mathcal{R}$, the number of blocks k, and a density measure ρ, **(2) Find:** k distinct blocks of $\mathcal{R}$ with the highest densities in terms of ρ.

We also consider a variant of Problem 1 which incorporates lower and upper bounds on the size of the detected blocks. This is particularly useful if the unrestricted densest block is not meaningful due to being too small (e.g. a single tuple) or too large (e.g. the entire tensor).

Problem 2 (k-Densest Blocks with Size Bounds). **(1) Given:** a relation $\mathcal{R}$, the number of blocks k, a density measure ρ, lower size bound S_{min}, and upper size bound S_{max}, **(2) Find:** k distinct blocks of $\mathcal{R}$ with the highest densities in terms of ρ **(3) Among:** blocks whose sizes are at least S_{min} and at most S_{max}.

Even when we restrict our attention to a special case (N=2, k=1, ρ=ρ_{ari}, S_{min}=S_{max}), exactly solving Problems 1 and 2 takes $O(S_{\mathcal{R}}^6)$ time [9] and is NP-hard [3], resp., infeasible for large datasets. Thus, we focus on an approximation algorithm which (1) has linear scalability with all aspects of $\mathcal{R}$, (2) provides accuracy guarantees at least for some density measures, and (3) produces meaningful results in real-world datasets, as explained in detail in Section 3 and Section 4.

3 Proposed Method

In this section, we propose M-Zoom (Multidimensional Zoom), a scalable, accurate, and flexible method for finding dense blocks in a tensor. We present the details of M-Zoom in Section 3.1 and discuss its efficient implementation in Section 3.2. After analyzing the time and space complexity in Section 3.3, we prove the quality guarantees provided by M-Zoom in Section 3.4.

3.1 Algorithm

Algorithm 1 describes the outline of M-Zoom. M-Zoom first copies the given relation $\mathcal{R}$ and assigns it to $\mathcal{R}^{ori}$ (line 1). Then, M-Zoom finds k dense blocks one by one from $\mathcal{R}$ (line 4). After finding each block from $\mathcal{R}$, M-Zoom removes the tuples in the block from $\mathcal{R}$ to prevent the same block from being found again (line 5). Due to these changes in $\mathcal{R}$, a block found in $\mathcal{R}$ is not necessarily a block of the original relation $\mathcal{R}^{ori}$. Thus, instead of returning the blocks found in $\mathcal{R}$, M-Zoom returns the blocks of $\mathcal{R}^{ori}$ consisting of the same attribute values with the found blocks (lines 6-7). This also enables M-Zoom to find overlapped blocks, i.e., a tuple can be included in two or more blocks.

Algorithm 2 describes how M-Zoom finds a single dense block from the given relation $\mathcal{R}$. The block $\mathcal{B}$ is initialized to $\mathcal{R}$ (lines 1-2). From $\mathcal{B}$, M-Zoom removes attribute values one by one in a greedy way until no attribute value is

Algorithm 1: M-Zoom

Input : relation: $\mathcal{R}$, number of blocks: k, density measure: ρ, lower size bound: S_{min}, upper size bound: S_{max}
Output: k dense blocks

1 $\mathcal{R}^{ori} \leftarrow copy(\mathcal{R})$
2 $results \leftarrow \emptyset$
3 **for** $i \leftarrow 1..k$ **do**
4 $\quad \mathcal{B} \leftarrow find\_single\_block(\mathcal{R}, \rho, S_{min}, S_{max})$ ▷ see Algorithm 2
5 $\quad \mathcal{R} \leftarrow \mathcal{R} - \mathcal{B}$
6 $\quad \mathcal{B}^{ori} \leftarrow \{t \in \mathcal{R}^{ori} : \forall n \in [N], t[A_n] \in \mathcal{B}_n\}$
7 $\quad results \leftarrow results \cup \{\mathcal{B}^{ori}\}$
8 return $results$

Algorithm 2: *find_single_block in* M-Zoom

Input : relation: $\mathcal{R}$, density measure: ρ, lower size bound: S_{min}, upper size bound: S_{max}
Output: a dense block

1 $\mathcal{B} \leftarrow copy(\mathcal{R})$
2 $\mathcal{B}_n \leftarrow copy(\mathcal{R}_n)$, $\forall n \in [N]$
3 $snapshots \leftarrow \emptyset$
4 **while** $\exists n \in [N]$ *s.t.* $\mathcal{B}_n \neq \emptyset$ **do**
5 $\quad$ **if** $\mathcal{B}$ *is in size bounds* (i.e., $S_{min} \leq S_{\mathcal{B}} \leq S_{max}$) **then**
6 $\quad\quad snapshots \leftarrow snapshots \cup \{\mathcal{B}\}$
7 $\quad a_i^* \leftarrow a_i \in \bigcup_{n=1}^{N} \mathcal{B}_n$ with maximum $\rho(\mathcal{B} - \mathcal{B}(a_i), \mathcal{R})$ ▷ see Algorithm 3
8 $\quad \mathcal{B} \leftarrow \mathcal{B} - \mathcal{B}(a_i^*)$
9 $\quad \mathcal{B}_i \leftarrow \mathcal{B}_i - \{a_i^*\}$
10 return $\mathcal{B} \in snapshots$ with maximum $\rho(\mathcal{B}, \mathcal{R})$

left (line 4). Specifically, M-Zoom finds the attribute value a_i that maximizes $\rho(\mathcal{B} - \mathcal{B}(a_i), \mathcal{R})$, which corresponds to the density when tuples with $A_i = a_i$ are removed from $\mathcal{B}$ (line 7). Then, the attribute value, denoted by a_i^*, and the tuples with $A_i = a_i^*$ are removed from $\mathcal{B}_i$ and $\mathcal{B}$, respectively (lines 8-9). Before removing each attribute value, M-Zoom adds the current $\mathcal{B}$ to the snapshot list if $\mathcal{B}$ satisfies the size bound (i.e., $S_{min} \leq S_{\mathcal{B}} \leq S_{max}$) (lines 5-6). As the final step of finding a block, M-Zoom returns the block with the maximum density among those in the snapshot list (line 10).

3.2 Efficient Implementation of M-Zoom

In this section, we discuss an efficient implementation of M-Zoom focusing on the greedy attribute value selection and the densest block selection.

Attribute Value Selection Using Min-Heaps. Finding the attribute value $a_i \in \bigcup_{n=1}^{N} \mathcal{B}_n$ that maximizes $\rho(\mathcal{B} - \mathcal{B}(a_i), \mathcal{R})$ (line 7 of Algorithm 2) can be

Algorithm 3: Greedy Selection Using Min-Heap in M-Zoom

Input : current block: $\mathcal{B}$, density measure: ρ, min-heaps: $\{H_n\}_{n=1}^N$
Output: attribute value to remove

1 **for** each dimension $n \in [N]$ **do**
2 $\quad a'_n \leftarrow a_n$ with minimum key in H_n ▷ key= $M_{\mathcal{B}(a_n)}$
3 $a_i^* \leftarrow a'_i \in \{a'_n\}_{n=1}^N$ with maximum $\rho(\mathcal{B} - \mathcal{B}(a'_i), \mathcal{R})$
4 delete a_i^* from H_i
5 **for** each tuple $t \in \mathcal{B}(a_i^*)$ **do**
6 $\quad$ **for** each dimension $n \in [N] \backslash \{i\}$ **do**
7 $\quad\quad$ decrease the key of $t[A_n]$ in H_n by $t[X]$ ▷ key= $M_{\mathcal{B}(t[A_n])}$
8 return a_i^*

computationally very expensive if all possible attribute values (i.e., $\bigcup_{n=1}^N \mathcal{B}_n$) should be considered. However, due to Axiom 1, which is assumed to be satisfied by considered density measures, the number of candidates is reduced to N if $M_{\mathcal{B}(a_i)}$ is known for each attribute value a_i. Lemma 1 states this.

Lemma 1. *If we remove a value of attribute A_n from $\mathcal{B}_n$, removing $a_n \in \mathcal{B}_n$ with minimum $M_{\mathcal{B}(a_n)}$ results in the highest density. Formally,*
$M_{\mathcal{B}(a'_n)} \leq M_{\mathcal{B}(a_n)}, \forall a_n \in \mathcal{B}_n \Rightarrow \rho(\mathcal{B} - \mathcal{B}(a'_n), \mathcal{R}) \geq \rho(\mathcal{B} - \mathcal{B}(a_n), \mathcal{R}), \forall a_n \in \mathcal{B}_n$.

Proof. Let $\mathcal{B}' = \mathcal{B} - \mathcal{B}(a'_n)$ and $\mathcal{B}'' = \mathcal{B} - \mathcal{B}(a_n)$. Then, $|\mathcal{B}'_n| = |\mathcal{B}''_n|, \forall n \in [N]$. In addition, $M_{\mathcal{B}'} \geq M_{\mathcal{B}''}$ since $M_{\mathcal{B}'} = M_{\mathcal{B}} - M_{\mathcal{B}(a'_n)} \geq M_{\mathcal{B}} - M_{\mathcal{B}(a_n)} = M_{\mathcal{B}''}$. Hence, by Axiom 1, $\rho(\mathcal{B} - \mathcal{B}(a'_n), \mathcal{R}) \geq \rho(\mathcal{B} - \mathcal{B}(a_n), \mathcal{R})$. □

By Lemma 1, if we let a'_n be $a_n \in \mathcal{B}_n$ with minimum $M_{\mathcal{B}(a_n)}$, we only have to consider values in $\{a'_n\}_{n=1}^N$ instead of $\bigcup_{n=1}^N \mathcal{B}_n$ to find the attribute value maximizing density when it is removed. To exploit this, our implementation of M-Zoom maintains a min-heap for each attribute A_n where the key of each value a_n is $M_{\mathcal{B}(a_n)}$. This key is updated, which takes $O(1)$ if Fibonacci Heaps are used as min-heaps, whenever the tuples with the corresponding attribute value are removed. Algorithm 3 describes in detail how to find the attribute value to be removed based on these min-heaps, and update keys in them. Since Algorithm 3 considers all promising attribute values (i.e., $\{a'_n\}_{n=1}^N$), it is guaranteed to find the value that maximizes density when it is removed, as Theorem 1 states.

Theorem 1. *Algorithm 3 returns $a_i \in \bigcup_{n=1}^N \mathcal{B}_n$ with maximum $\rho(\mathcal{B} - \mathcal{B}(a_i), \mathcal{R})$.*

Proof. Let a_i^* be $a_i \in \bigcup_{n=1}^N \mathcal{B}_n$ with maximum $\rho(\mathcal{B} - \mathcal{B}(a_i), \mathcal{R})$. By Lemma 1, a_i^* exists among $\{a'_n\}_{n=1}^N$, all of which are considered in Algorithm 3. □

Densest Block Selection Using Attribute Value Ordering. As explained in Section 3.1, M-Zoom returns the densest block among snapshots of $\mathcal{B}$ (line 10 of Algorithm 2). Explicitly maintaining the list of snapshots, whose length is at most $S_{\mathcal{R}}$, requires $O(N|\mathcal{R}|S_{\mathcal{R}})$ computation and space for copying them. Even

maintaining only the current best (i.e., the one with the highest density so far) cannot avoid high computational cost if the current best keeps changing. Instead, our implementation maintains the order by which attribute values are removed as well as the iteration where the density was maximized, which requires only $O(S_{\mathcal{R}})$ space. From these and the original relation $\mathcal{R}$, our implementation restores the snapshot with maximum density in $O(N|\mathcal{R}| + S_{\mathcal{R}})$ time and returns it.

3.3 Complexity Analysis

The time and space complexity of M-Zoom depend on the density measure used. In this section, we assume that one of the density measures in Section 2.2, which satisfy Axiom 1, is used.

Theorem 2. *The time complexity of Algorithm 1 is $O(kN|\mathcal{R}| \log L)$ if $|\mathcal{R}_n| = L$, $\forall n \in [N]$, and $N = O(\log L)$.*

Proof. See Appendix B. □

As stated in Theorem 2, M-Zoom scales linearly or sub-linearly with all aspects of relation $\mathcal{R}$ as well as k, the number of blocks we aim to find. This result is also experimentally supported in Section 4.4. In our experiments, the actual running time scaled sub-linearly with k as well as L since the number of tuples in $\mathcal{R}$ decreases as M-Zoom finds blocks (line 5 in Algorithm 1).

Theorem 3. *The space complexity of Algorithm 1 is $O(kN|\mathcal{R}|)$.*

Proof. See the supplementary document [1]. □

M-Zoom requires up to $kN|\mathcal{R}|$ space for storing k found blocks, as stated in Theorem 3. However, since the blocks are usually far smaller than $\mathcal{R}$, as seen in Tables 4 and 5 in Section 4, actual space usage is much less than $kN|\mathcal{R}|$.

3.4 Accuracy Guarantee

In this section, we show lower bounds on the densities of the blocks found by M-Zoom on the assumption that ρ_{ari} (Definition 1) is used as the density measure. Specifically, we show that Algorithm 2 without size bounds is guaranteed to find a block with density at least $1/N$ of maximum density in the given relation (Theorem 4). This means that each n-th block returned by Algorithm 1 has density at least $1/N$ of maximum density in $\mathcal{R} - \bigcup_{i=1}^{n-1}(i\text{-th block})$.

Let $\mathcal{B}^{(r)}$ be the relation $\mathcal{B}$ at the beginning of the r-th iteration of Algorithm 2, and $a_i^{(r)} \in \mathcal{B}_i^{(r)}$ be the attribute value removed in the same iteration.

Lemma 2. *If a block $\mathcal{B}'$ satisfying $\forall a_i \in \bigcup_{n=1}^{N} \mathcal{B}'_n$, $M_{\mathcal{B}'(a_i)} \geq c$ exists, there exists $\mathcal{B}^{(r)}$ satisfying $\forall a_i \in \bigcup_{n=1}^{N} \mathcal{B}_n^{(r)}$, $M_{\mathcal{B}^{(r)}(a_i)} \geq c$.*

Proof. See Appendix C. □

Theorem 4 ($1/N$-Approximation Guarantee for Problem 1). *Given a relation $\mathcal{R}$, let $\mathcal{B}^*$ be the block $\mathcal{B} \subset \mathcal{R}$ with maximum $\rho_{ari}(\mathcal{B}, \mathcal{R})$. Let $\mathcal{B}'$ be the block obtained by Algorithm 2 without size bounds (i.e., $S_{min} = 0$ and $S_{max} = \infty$). Then, $\rho_{ari}(\mathcal{B}', \mathcal{R}) \geq \rho_{ari}(\mathcal{B}^*, \mathcal{R})/N$.*

Proof. $\forall a_i \in \bigcup_{n=1}^{N} \mathcal{B}_n^*$, $M_{\mathcal{B}^*(a_i)} \geq M_{\mathcal{B}^*}/S_{\mathcal{B}^*}$. Otherwise, a contradiction would result since for a_i with $M_{\mathcal{B}^*(a_i)} < M_{\mathcal{B}^*}/S_{\mathcal{B}^*}$,

$$\rho_{ari}(\mathcal{B}^* - \mathcal{B}^*(a_i), \mathcal{R}) = \frac{M_{\mathcal{B}^*} - M_{\mathcal{B}^*(a_i)}}{(S_{\mathcal{B}^*} - 1)/N} > \frac{M_{\mathcal{B}^*} - M_{\mathcal{B}^*}/S_{\mathcal{B}^*}}{(S_{\mathcal{B}^*} - 1)/N} = \rho_{ari}(\mathcal{B}^*, \mathcal{R}).$$

Consider $\mathcal{B}^{(r)}$ where $\forall a_i \in \bigcup_{n=1}^{N} \mathcal{B}_n^{(r)}$, $M_{\mathcal{B}^{(r)}(a_i)} \geq M_{\mathcal{B}^*}/S_{\mathcal{B}^*}$. Such $\mathcal{B}^{(r)}$ exists by Lemma 2. $M_{\mathcal{B}^{(r)}} \geq (S_{\mathcal{B}^{(r)}}/N)\ (M_{\mathcal{B}^*}/S_{\mathcal{B}^*}) = (S_{\mathcal{B}^{(r)}}/N)(\rho_{ari}(\mathcal{B}^*, \mathcal{R})/N)$. Hence, $\rho_{ari}(\mathcal{B}', \mathcal{R}) \geq \rho_{ari}(\mathcal{B}^{(r)}, \mathcal{R}) = M_{\mathcal{B}^{(r)}}/(S_{\mathcal{B}^{(r)}}/N) \geq \rho_{ari}(\mathcal{B}^*, \mathcal{R})/N$. □

Theorem 4 can be extended to cases where a lower bound exists. In these cases, the approximate factor is $1/(N+1)$, as stated in Theorem 5.

Theorem 5 ($1/(N+1)$-Approximation Guarantee for Problem 2). *Given a relation $\mathcal{R}$, let $\mathcal{B}^*$ be the block $\mathcal{B} \subset \mathcal{R}$ with maximum $\rho_{ari}(\mathcal{B}, \mathcal{R})$ among blocks with size at least S_{min}. Let $\mathcal{B}'$ be the block obtained by Algorithm 2 with lower size bound (i.e., $1 \leq S_{min} \leq S_{\mathcal{R}}$ and $S_{max} = \infty$). Then, $\rho_{ari}(\mathcal{B}', \mathcal{R}) \geq \rho_{ari}(\mathcal{B}^*, \mathcal{R})/(N+1)$.*

Proof. See the supplementary document [1]. □

4 Experiments

We designed and performed experiments to answer the following questions:

- **Q1.** How fast and accurately does M-Zoom detect dense blocks in real data?
- **Q2.** Does M-Zoom find many different dense blocks in real data?
- **Q3.** Does M-Zoom scale linearly with all aspects of data?
- **Q4.** Which anomalies or fraud does M-Zoom spot in real data?

4.1 Experimental Settings

All experiments were conducted on a machine with 2.67 GHz Intel Xeon E7-8837 CPUs and 1TB RAM. We compared M-Zoom with CrossSpot [12], CP Decomposition (CPD) [17] (see Appendix A for details), and MultiAspectForensics (MAF) [19]. M-Zoom and CrossSpot[1] were implemented in Java, and Tensor Toolbox [4] was used for CPD and MAF. Although CrossSpot was originally designed to maximize ρ_{susp}, it can be extended to other density measures. These variants were used depending on the density measure compared in each experiment. In addition, we used CPD as a seed selection method of CrossSpot, which outperformed HOSVD used in [12] in terms of both speed and accuracy. We used diverse real-world datasets, grouped as follows:

[1] We referred the open-sourced implementation at `http://github.com/mjiang89/CrossSpot`.

Table 3: Summary of real-world datasets.

	StackO.	Youtube	KoWiki	EnWiki	Yelp	Netflix	YahooM.	AirForce
N	3	3	3	3	4	4	4	7
$\lvert\mathcal{R}_1\rvert$	545K	3.22M	470K	44.1M	552K	480K	1.00M	3
$\lvert\mathcal{R}_2\rvert$	96.7K	3.22M	1.18M	38.5M	77.1K	17.8K	625K	70
$\lvert\mathcal{R}_3\rvert$	1.15K	203	101K	129K	3.80K	2.18K	84.4K	11
$\lvert\mathcal{R}_4\rvert$	-	-	-	-	5	5	101	7.20K*
$\lvert\mathcal{R}\rvert$	1.30M	18.7M	11.0M	483M	2.23M	99.1M	253M	648K

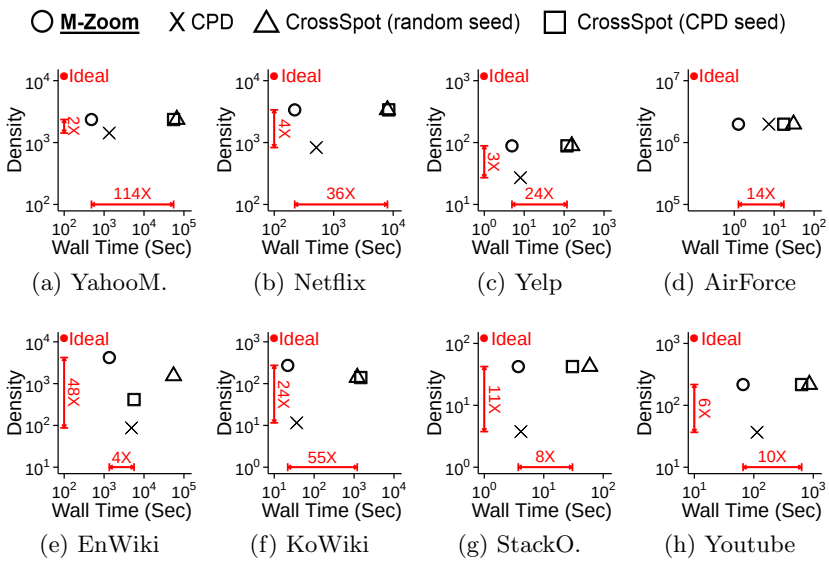
$*$ $|\mathcal{R}_5|$=21.5K, $|\mathcal{R}_6|$=512, $|\mathcal{R}_7|$=512

- **User behavior logs:** *StackO.(user,post,timestamp,1)* represents who marked which post as a favorite when on Stack Overflow. *Youtube(user,user,date,1)* represents who became a friend of whom when on Youtube. *KoWiki(user,page, timestamp,#revisions)* and *EnWiki(user,page,timestamp,#revisions)* represent who revised which page when how many times on Korean Wikipedia and English Wikipedia, respectively.
- **User reviews:** *Yelp(user,business,date,score,1)*, *Netflix(user,movie,date,score,1)*, and *YahooM.(user,item,timestamp,score,1)* represent who gave which score when to which business, movie, and item on Yelp, Netflix, and Yahoo Music, respectively.
- **TCP dumps:** From TCP dump data for a typical U.S. Air Force LAN, we created a relation *AirForce(protocol,service,src_bytes,dst_bytes,flag,host_count ,src_count,#connections)*. See the supplementary document [1] for the description of each attribute.

Timestamps are in hours in all the datasets. Table 3 summarizes all the datasets.

4.2 Q1. Running Time and Accuracy of M-Zoom

We compare the speed of different methods and the densities of the blocks found by the methods in real-world datasets. Specifically, we measured time taken to find three blocks and the maximum density among the three blocks. Figure 2 shows the result when ρ_{ari} was used as the density measure. M-Zoom clearly provided the best trade-off between speed and accuracy in all datasets. For example, in YahooM. Dataset, M-Zoom was 114 times faster than CrossSpot, while detecting blocks with similar densities. Compared with CPD, M-Zoom detected two times denser blocks 2.8 times faster. Although the results are not included in Figure 2, MAF found several orders of magnitude sparser blocks than the other methods, with speed similar to that of CPD. M-Zoom also gave the best trade-off between speed and accuracy when ρ_{geo} or ρ_{susp} was used instead of ρ_{ari} (see the supplementary document [1]).



(a) YahooM. (b) Netflix (c) Yelp (d) AirForce

(e) EnWiki (f) KoWiki (g) StackO. (h) Youtube

Fig. 2: **Only M-Zoom achieves both speed and accuracy.** In each plot, points represent the speed of different methods and the highest density (ρ_{ari}) of three blocks found by the methods. Upper-left region indicates better performance. M-Zoom gives the best trade-off between speed and density. Specifically, M-Zoom is up to **114× faster** than CrossSpot with similarly dense blocks.

4.3 Q2. Diversity of Blocks Found by M-Zoom

We compare the diversity of dense blocks found by each method. Ability to detect many different dense blocks is useful since distinct blocks may indicate different anomalies or fraud. We define the diversity as the average dissimilarity between the pairs of blocks, and the dissimilarity of two blocks is defined as $dissimilarity(\mathcal{B}, \mathcal{B}') = 1 - \frac{|(\bigcup_{n=1}^{N} \mathcal{B}_n) \cap (\bigcup_{n=1}^{N} \mathcal{B}'_n)|}{|(\bigcup_{n=1}^{N} \mathcal{B}_n) \cup (\bigcup_{n=1}^{N} \mathcal{B}'_n)|}$. Diversities were measured among three blocks found by each method using ρ_{ari} as the density metric.

As seen in Figure 3, in all datasets, M-Zoom and CPD successfully detected distinct dense blocks. CrossSpot, however, found the same block repeatedly or blocks with slight difference, even when it started from different seed blocks. Although using CPD for seed-block selection in CrossSpot improved the diversity, the effect was limited in most datasets. Similar results were obtained when ρ_{geo} or ρ_{susp} was used instead of ρ_{ari} (see the supplementary document [1]).

4.4 Q3. Scalability of M-Zoom

We empirically demonstrate the scalability of M-Zoom, mathematically analyzed in Theorem 2. Specifically, we measured the scalability of M-Zoom with regard to the number of tuples, the number of attributes, the cardinalities of attributes, and the number of blocks we aim to find. We started with finding

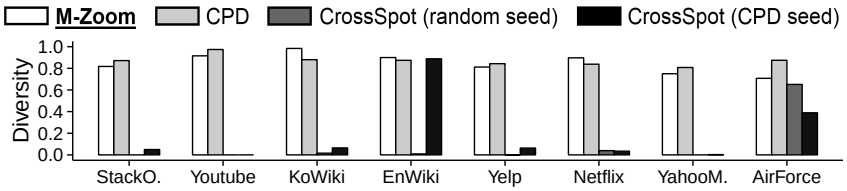
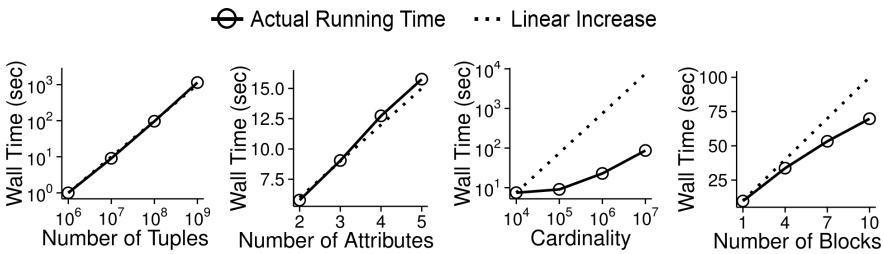


Fig. 3: **M-Zoom detects many different dense blocks.** The dense blocks found by M-Zoom and CPD have high diversity, while the dense blocks found by CrossSpot tend to be almost same.



(a) Number of Tuples ($|\mathcal{R}|$) (b) Number of Attributes (N) (c) Cardinality of Attributes ($|\mathcal{R}_n|$) (d) Number of Blocks to Find (k)

Fig. 4: **M-Zoom is scalable.** (a) (b) M-Zoom scales linearly with the number of tuples and the number of attributes. (c) (d) M-Zoom scales sub-linearly with the cardinalities of attributes and the number of blocks we aim to find.

one block in a randomly generated 10 millions tuples with three attributes each of whose cardinality is 100K. Then, we measured the running time by changing one factor at a time while fixing the others. As seen in Figure 4, M-Zoom scaled linearly with the number of tuples and the number of attributes. Moreover, M-Zoom scaled sub-linearly with the number of blocks we aim to find as well as the cardinalities of attributes due to the reason explained in Section 3.3. These results held regardless of the density measure used.

4.5 Q4. Anomaly/Fraud Detection by M-Zoom in Real Data

We demonstrate the effectiveness of M-Zoom for anomaly and fraud detection by analyzing dense blocks detected by M-Zoom in real-world datasets.

M-Zoom spots edit wars and bot activities in Wikipedia. Table 4 lists the first three dense blocks found by M-Zoom in EnWiki and KoWiki Datasets. As seen in the third dense block visualized in Figure 1c, the dense blocks detected in KoWiki Dataset indicate edit wars. That is, users with conflicting opinions revised the same set of pages hundreds of times within several hours. On the other hand, the dense blocks detected in EnWiki Dataset indicate the activities of bots, which changed the same pages hundreds of thousands of times. Figure 1d lists the bots and pages corresponding to the second found block.

Table 4: **M-Zoom detects anomalous behaviors in Wikipedia.** The tables list the first three blocks detected by M-Zoom in KoWiki and EnWiki Datasets, which correspond to edit wars and bot activities, respectively.

Korean Wikipedia (KoWiki)				English Wikipedia (EnWiki)			
#	Volume	Mass	Density (ρ_{ari})	#	Volume	Mass	Density (ρ_{geo})
1	2×2×2	546	273	1	1×1,585×6,733	1.93M	8,772
2	2×2×3	574	246	2	8×12×67.9K	2.43M	13.0K
3	11×10×16	2,305	187	3	1×1×90	17.6K	3,933

Table 5: **M-Zoom identifies network attacks with near-perfect accuracy.** The first three blocks found by M-Zoom in AirForce Dataset consist of attacks.

#	Volume	Density (ρ_{geo})	# Connections	# Attacks (Ratio)
1	2	2,050,505	2,263,941	2,263,941 (100%)
2	1	263,295	263,295	263,295 (100%)
3	8,100	263,072	952,383	952,382 (99.9%)

M-Zoom spots network intrusions. Table 5 lists the first three blocks found by M-Zoom in AirForce Dataset. Based on the provided ground truth labels, all of the about 3 millions connections composing the blocks were attacks except only one normal connection. This indicates that malicious connections form dense blocks due to the similarity in their behaviors. Based on this observation, we could accurately separate normal connections and attacks based on the densities of blocks they belong (i.e., the denser block a connection belongs, the more suspicious it is). Especially, we got the highest AUC (Area under the curve) 0.98 with M-Zoom, as shown in Figure 1b, because M-Zoom detects many different dense blocks accurately, as shown in previous experiments. For each method, we used the best density measure that leads to the highest AUC.

5 Related Work

Dense Subgraph/Submatrix/Subtensor Detection. The *densest subgraph problem*, the problem of finding the subgraph which maximizes ρ_{ari} or ρ_{geo} (see Definitions 1 and 2), has been extensively studied in theory (see [18] for surveys). The two major directions are max-flow based exact algorithms [9,16] and greedy algorithms [7,16] giving a 1/2-approximation to the densest subgraph. Variants allow for size restrictions [3], providing a 1/3-approximation to the densest subgraph for the lower bound case. Another related line of research deals with dense blocks in binary matrices or tensors where the definition of density is designed for the purpose of frequent itemset mining [22] or formal concept mining [6,11].

Anomaly/Fraud Detection based on Dense Subgraphs. Spectral approaches make use of eigendecomposition or SVD of the adjacency matrix for

dense-block detection. Such approaches have been used to spot anomalous pattens in a patent graph [21], lockstep followers in a social network [14], and stealthy or small-scale attacks in social networks [23]. Other approaches include NetProbe [20], which used belief propagation to detect fraud-accomplice bipartite cores in an auction network, and CopyCatch [5], which used one-class clustering and sub-space clustering to identify "Like" boosting in Facebook. In addition, OddBall [2] spotted near-cliques in links among posts in blogs based on egonet features. Recently, Fraudar [10], which generalizes densest subgraph-detection methods so that the suspiciousness of nodes and edges can be incorporated, spotted follower-buying services in Twitter.

Anomaly/Fraud Detection based on Dense Subtensors. Spectral methods for dense subgraphs can be extended to tensors where tensor decomposition, such as CP Decomposition and HOSVD [17], is used to spot dense subtensors. MAF [19], which is based on CP Decomposition, detected dense blocks corresponding to port-scanning activities based on network traffic logs. Another approach is CrossSpot [12], which finds dense blocks by starting from seed blocks and growing them in a greedy way until ρ_{susp} (see Definition 3) converges. CrossSpot spotted retweet boosting in Weibo, outperforming HOSVD.

Our M-Zoom non-trivially generalizes theoretical results regarding the densest subgraph problem, especially [3], for supporting tensors, various density measures, and multi-block detection. As seen in Table 1, M-Zoom provides more flexibility than other methods for dense-block detection.

6 Conclusion

In this work, we propose M-Zoom, a flexible framework for finding dense blocks in tensors, which has the following advantages over state-of-the-art methods:

- **Scalable:** M-Zoom is up to **114× faster** than competitors with similar accuracy due to its linear scalability with all input factors (Figures 2 and 4).
- **Provably accurate:** M-Zoom provides lower bounds on the densities of the blocks it finds (Theorem 4) as well as high accuracy in real data (Figure 2).
- **Flexible:** M-Zoom supports high-order tensors, various density measures, multi-block detection, and size bounds (Table 1).
- **Effective:** M-Zoom successfully detected fraud based on a TCP dump with near-perfect accuracy (**AUC=0.98**), and anomalies in Wikipedia (Figure 1).

Reproducibility: Our open-sourced code and the data we used are at `http://www.cs.cmu.edu/~kijungs/codes/mzoom`.

Acknowledgments. This material is based upon work supported by the National Science Foundation under Grant No. CNS-1314632 and IIS-1408924. Research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-09-2-0053. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science

Foundation, or other funding parties. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

1. Supplementary document (examples, proofs, and additional experiments). Available online, http://www.cs.cmu.edu/~kijungs/codes/mzoom/supple.pdf
2. Akoglu, L., McGlohon, M., Faloutsos, C.: Oddball: Spotting anomalies in weighted graphs. In: PAKDD (2010)
3. Andersen, R., Chellapilla, K.: Finding dense subgraphs with size bounds. In: WAW (2009)
4. Bader, B.W., Kolda, T.G., et al.: Matlab tensor toolbox version 2.6. Available online, http://www.sandia.gov/~tgkolda/TensorToolbox/
5. Beutel, A., Xu, W., Guruswami, V., Palow, C., Faloutsos, C.: Copycatch: stopping group attacks by spotting lockstep behavior in social networks. In: WWW (2013)
6. Cerf, L., Besson, J., Robardet, C., Boulicaut, J.F.: Data peeler: Contraint-based closed pattern mining in n-ary relations. In: SDM (2008)
7. Charikar, M.: Greedy approximation algorithms for finding dense components in a graph. In: APPROX (2000)
8. Gibson, D., Kumar, R., Tomkins, A.: Discovering large dense subgraphs in massive graphs. In: VLDB (2005)
9. Goldberg, A.V.: Finding a maximum density subgraph. Technical Report (1984)
10. Hooi, B., Song, H.A., Beutel, A., Shah, N., Shin, K., Faloutsos, C.: Fraudar: Bounding graph fraud in the face of camouflage. In: KDD (2016)
11. Ignatov, D.I., Kuznetsov, S.O., Poelmans, J., Zhukov, L.E.: Can triconcepts become triclusters? Int. J. General Systems 42(6), 572–593 (2013)
12. Jiang, M., Beutel, A., Cui, P., Hooi, B., Yang, S., Faloutsos, C.: A general suspiciousness metric for dense blocks in multimodal data. In: ICDM (2015)
13. Jiang, M., Cui, P., Beutel, A., Faloutsos, C., Yang, S.: Catchsync: catching synchronized behavior in large directed graphs. In: KDD (2014)
14. Jiang, M., Cui, P., Beutel, A., Faloutsos, C., Yang, S.: Inferring strange behavior from connectivity pattern in social networks. In: PAKDD (2014)
15. Kannan, R., Vinay, V.: Analyzing the structure of large graphs. Technical Report (1999)
16. Khuller, S., Saha, B.: On finding dense subgraphs. In: ICALP, pp. 597–608 (2009)
17. Kolda, T.G., Bader, B.W.: Tensor decompositions and applications. SIAM review 51(3), 455–500 (2009)
18. Lee, V.E., Ruan, N., Jin, R., Aggarwal, C.: A survey of algorithms for dense subgraph discovery. In: Managing and Mining Graph Data, pp. 303–336 (2010)
19. Maruhashi, K., Guo, F., Faloutsos, C.: Multiaspectforensics: Pattern mining on large-scale heterogeneous networks with tensor analysis. In: ASONAM (2011)
20. Pandit, S., Chau, D.H., Wang, S., Faloutsos, C.: Netprobe: a fast and scalable system for fraud detection in online auction networks. In: WWW (2007)
21. Prakash, B., Seshadri, M., Sridharan, A., Machiraju, S., Faloutsos, C.: Eigenspokes: Surprising patterns and community structure in large graphs. PAKDD (2010)
22. Seppänen, J.K., Mannila, H.: Dense itemsets. In: KDD (2004)
23. Shah, N., Beutel, A., Gallagher, B., Faloutsos, C.: Spotting suspicious link behavior with fbox: An adversarial perspective. In: ICDM (2014)

A CP Decomposition (CPD)

In a graph, dense subgraphs lead to high singular values of the adjacency matrix [23]. The singular vectors corresponding to the high singular values roughly indicate which nodes form dense blocks. This idea can be extended to tensors, where dense blocks are captured by components in CP Decomposition [17]. Let $\mathbf{A}^{(1)} \in \mathbb{R}^{|\mathcal{R}_1| \times k}$, $\mathbf{A}^{(2)} \in \mathbb{R}^{|\mathcal{R}_2| \times k}$, ..., $\mathbf{A}^{(N)} \in \mathbb{R}^{|\mathcal{R}_N| \times k}$ be the factor matrices obtained by the rank-k CP Decomposition of $\mathcal{R}$. For each $i \in [k]$, we form a block with every attribute value a_n whose corresponding element in the i-th column of $\mathbf{A}^{(n)}$ is at least $1/\sqrt{|\mathcal{R}_n|}$.

B Proof of Theorem 2

Proof. In Algorithm 3, lines 1-3 take $O(N)$ for all the density measures considered (i.e., ρ_{ari}, ρ_{geo}, and ρ_{susp}) if we maintain and update aggregated values (e.g., M_B, S_B, and V_B) instead of computing $\rho(\mathcal{B} - \mathcal{B}(a_i'), \mathcal{R})$ from scratch every time. In addition, line 4 takes $O(\log |\mathcal{R}_n|)$ and lines 5-7 take $O(N|\mathcal{B}(a_i^*)|)$ if we use Fibonacci heaps. Algorithm 2, whose computational bottleneck is line 7, has time complexity $O(N|\mathcal{R}| + N\sum_{n=1}^{N} |\mathcal{R}_n| + \sum_{n=1}^{N} |\mathcal{R}_n| \log |\mathcal{R}_n|))$ since lines 1-4 of Algorithm 3 are executed $S_{\mathcal{R}} = \sum_{n=1}^{N} |\mathcal{R}_n|$ times, and line 7 is executed $N|\mathcal{R}|$ times. Algorithm 1, whose computational bottleneck is line 4, has time complexity $O(kN|\mathcal{R}| + kN\sum_{n=1}^{N} |\mathcal{R}_n| + k\sum_{n=1}^{N} |\mathcal{R}_n| \log |\mathcal{R}_n|))$ since Algorithm 2 is executed k times.

Assume $|\mathcal{R}_n| = L, \forall n \in [N]$, and $N = O(\log L)$. The time complexity of Algorithm 1 becomes $O(kN(|\mathcal{R}| + NL + L \log L))$. Since $N = O(\log L)$, by assumption, and $L \leq |\mathcal{R}|$, there exists a constant c such that $|\mathcal{R}| + NL + L \log L \leq c|\mathcal{R}| \log L = O(|\mathcal{R}| \log L)$. Thus, the time complexity of Algorithm 1 is $O(kN|\mathcal{R}| \log L)$. □

C Proof of Lemma 2

Lemma 3. *$a_i^{(r)}$ minimizes $Mass(\mathcal{B}^{(r)}(a_j))$ among $a_j \in \bigcup_{n=1}^{N} \mathcal{B}_n^{(r)}$.*

Proof. From Theorem 1, $\rho_{ari}(\mathcal{B}^{(r)} - \mathcal{B}^{(r)}(a_i^{(r)}), \mathcal{R}) \geq \rho_{ari}(\mathcal{B}^{(r)} - \mathcal{B}^{(r)}(a_j), \mathcal{R})$, $\forall a_j \in \bigcup_{n=1}^{N} \mathcal{B}_n^{(r)}$. Thus, $Mass(\mathcal{B}^{(r)} - \mathcal{B}^{(r)}(a_i^{(r)})) = \rho_{ari}(\mathcal{B}^{(r)} - \mathcal{B}^{(r)}(a_i^{(r)}), \mathcal{R})$ $(Size(\mathcal{B}^{(r)}) - 1)/N \geq \rho_{ari}(\mathcal{B}^{(r)} - \mathcal{B}^{(r)}(a_j), \mathcal{R})(Size(\mathcal{B}^{(r)}) - 1)/N = Mass(\mathcal{B}^{(r)} - \mathcal{B}^{(r)}(a_j))$. Then, $Mass(\mathcal{B}^{(r)}(a_i^{(r)})) = Mass(\mathcal{B}^{(r)}) - Mass(\mathcal{B}^{(r)} - \mathcal{B}^{(r)}(a_i^{(r)})) \leq Mass(\mathcal{B}^{(r)}) - Mass(\mathcal{B}^{(r)} - \mathcal{B}^{(r)}(a_j)) = Mass(\mathcal{B}^{(r)}(a_j))$, $\forall a_j \in \bigcup_{n=1}^{N} \mathcal{B}_n^{(r)}$. □

Proof of Lemma 2.

Proof. Let r be the first iteration in Algorithm 2 where $a_i^{(r)} \in \bigcup_{n=1}^{N} \mathcal{B}_n'$. Since $\mathcal{B}^{(r)} \supset \mathcal{B}'$, $Mass(\mathcal{B}^{(r)}(a_i^{(r)})) \geq Mass(\mathcal{B}'(a_i^{(r)})) \geq c$. By Lemma 3, $\forall a_j \in \bigcup_{n=1}^{N} \mathcal{B}_n^{(r)}$, $Mass(\mathcal{B}^{(r)}(a_j)) \geq Mass(\mathcal{B}^{(r)}(a_i^{(r)})) \geq c$. □