Future Generation Computer Systems 115 (2021) 236–250

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

EagleMine: Vision-guided Micro-clusters recognition and collective anomaly detection

Wenjie Feng [a,b,*], Shenghua Liu [a,b,**], Christos Faloutsos [c], Bryan Hooi [d], Huawei Shen [a,b], Xueqi Cheng [a,b]

[a] CAS Key Laboratory of Network Data Science & Technology, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, 100190, China
[b] University of Chinese Academy of Sciences (UCAS), Beijing, 100049, China
[c] Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, 15213, United States of America
[d] Department of Computer Science, National University of Singapore, Singapore

ARTICLE INFO

Article history:
Received 27 May 2020
Received in revised form 13 August 2020
Accepted 18 August 2020
Available online 28 August 2020

Keywords:
Pattern recognition
Large graph mining
Micro-clusters
Anomaly detection
Histogram

ABSTRACT

Given a graph with millions of nodes, what patterns exist in the distributions of node characteristics? How can we detect them and separate anomalous nodes in a way similar to human visual perception? More generally, how can we identify micro-clusters in a histogram and spot some interesting patterns?

In this paper, we propose a vision-guided algorithm, EagleMine, to recognize and summarize node groups in a histogram constructed from some correlated features. EagleMine hierarchically discovers node groups, which form internally connected dense areas in the histogram, by utilizing a water-level tree with multiple resolutions according to the rule of the visual recognition. EagleMine uses the statistical hypothesis test to determine the optimal groups while exploring the tree and simultaneously performs vocabulary-based summarization. Moreover, EagleMine can identify anomalous micro-clusters, consisting of nodes that exhibit very similar and suspicious behavior, deviate away from the majority. Experiments on the real-world datasets show that our method can recognize intuitive node groups as human vision does; it achieves the best summarization performance compared to baselines. In terms of anomaly detection, EagleMine also outperforms the state-of-the-art graph-based methods with significantly improving accuracy in a micro-blog dataset. Moreover, EagleMine can be used for other applications, e.g., to detect the synchronized patterns in the temporal retweet event.

1. Introduction

Given real-world graphs with millions of nodes and edges, the most intuitive way to explore the graphs is to construct a correlation plot [1] based on the features of graph nodes. Usually, a heatmap of those scatter points, which is a two-dimensional histogram, is used to depict their density of aggregation [2]. In a histogram, people can visually recognize nodes gathering into separate dense areas as groups (see Fig. 1), which help to explore some interesting patterns (like communities, co-author association behaviors) and spot anomalies (e.g., fraudsters, attackers, fake-reviewers, outliers) in an interpretable way [3,4].

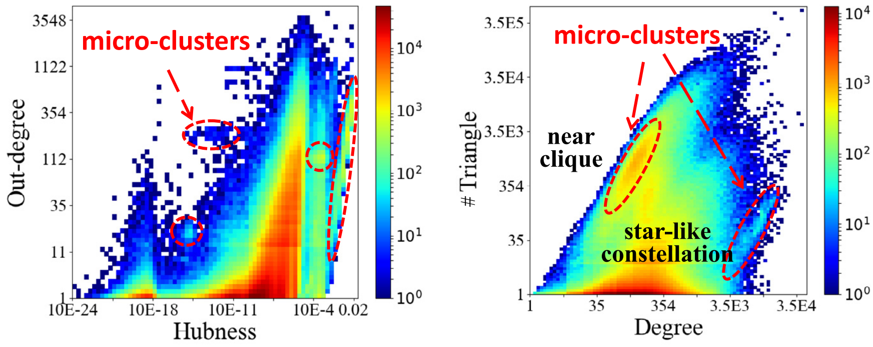
In particular, graphs can represent various relations, such as friendships on Facebook, ratings from users to items on Amazon, or retweets from users to messages on Twitter, even they are time evolving. So, a snapshot of such graphs has numerous correlated features, e.g., degree, #triangle, and PageRank, the combination of which will generate many correlation plots. However, it becomes labor-intensive to manually monitor and recognize patterns from the histogram of the snapshots for temporal graphs. Moreover, if extended to more features, the visualization and pattern-recognition become difficult for the resultant multi-dimensional histogram. This raises the following question.

Question 1. *Given a histogram of the correlation plot of graph nodes in some feature spaces, how can we design an effective algorithm to automatically*

- ***recognize*** *the node groups as human vision does?*
- ***summarize*** *the point distribution in the feature spaces?*
- ***identify*** *some suspicious micro-clusters?*

[*] Corresponding author at: CAS Key Laboratory of Network Data Science & Technology, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, 100190, China.
[**] Corresponding author.
E-mail addresses: wenchiehfeng.us@gmail.com (W. Feng), liushenghua@ict.ac.cn (S. Liu), christos@cs.cmu.edu (C. Faloutsos), bhooi@comp.nus.edu.sg (B. Hooi), shenhuawei@ict.ac.cn (H. Shen), cxq@ict.ac.cn (X. Cheng).

https://doi.org/10.1016/j.future.2020.08.033
0167-739X/© 2020 Elsevier B.V. All rights reserved.



(a) Microblog Sina Weibo

(b) Tagged

Fig. 1. Histogram of correlation plots for graph node in different applications. (a) Out-degree vs. Hubness for users in retweets relation of Sina Weibo. (b) # Triangle vs. Degree of user in friendship of Tagged [5]. (color figure online).

'Micro-cluster' refers to a relatively small group of nodes that exhibit very similar behavior in the feature spaces, thus, they actually form the collective anomalies [6], e.g., a group of fraudsters collude to artificially boost the reputation of mediocre services and products. Here we demonstrate some of the possible feature spaces, namely

i out-degree vs. hubness – Fig. 1a – this can be used to spot nodes with high out-degree but low hubness scores (i.e. fraudsters, which have many outgoing edges to these unimportant nodes, probably, customers that paid them) [7].
ii # triangle vs. degree – Fig. 1b – catching a near-clique group (too many triangles, for their low degree), as well as star-like constellations (too few triangles for such high degree) [8].

In this paper, we propose EagleMine, a novel tree-based mining approach to recognize and summarize the node groups in a heatmap for correlation plot of items (like graph node and behavior). EagleMine can also identify anomalous micro-clusters. Experiments show that EagleMine outperforms baselines and achieves better performance both in quantitative (i.e. the total encoded length for a compact model description) and qualitative (i.e. the consistency with vision-based judgment) comparisons. EagleMine detects a micro-cluster of hundreds of bots in a real-world micro-blog data, Sina Weibo,[1] which present strong signs of sharing unusual login-name prefixes, e.g., 'best*', 'black*' and '18-year-old*', and exhibit very similar behavior in the feature spaces (see Fig. 2b). It also can be used to detect micro-clusters in other applications beyond the graph and find some interesting patterns (see Section 5.6).

The main strengths of EagleMine are as follows:

- **Automated summarization:** EagleMine automatically summarizes the histogram derived from the correlated features and recognizes node groups forming disjoint dense areas as human vision does (see Figs. 2a and 2i).
- **Effectiveness:** EagleMine detects interpretable node groups and outperforms other baselines (clustering methods) and even those with *manually tuned parameters* in qualitative (see Figs. 2d–2i) and quantitative experiments (see Fig. 8).
- **Anomaly detection:** EagleMine can spot some explainable anomalies concerning suspicious node groups (see Fig. 2b) in real large graphs. Compared with the graph-based anomaly detection methods, EagleMine achieves higher accuracy for finding suspicious users in Sina Weibo data. EagleMine also detects micro-clusters for spotting suspicious objects in other applications beyond the graph, like temporal behavior analysis (see Fig. 9).
- **Scalability:** EagleMine is scalable, with nearly linear time in the total value of histogram, which equal to the number of nodes for the graph case, and can deal with more correlated features in multi-dimensional spaces.

Reproducibility: Our code is open-sourced at , and most of the datasets we used are publicly available online.

The rest of the paper is organized as follows. Section 2 surveys the related work. Section 3 gives the notions and our proposed model and Section 4 introduces the details of the EagleMine algorithm for recognizing and summarizing graph node groups. Experiments are given in Section 5. Finally, Section 6 concludes the paper.

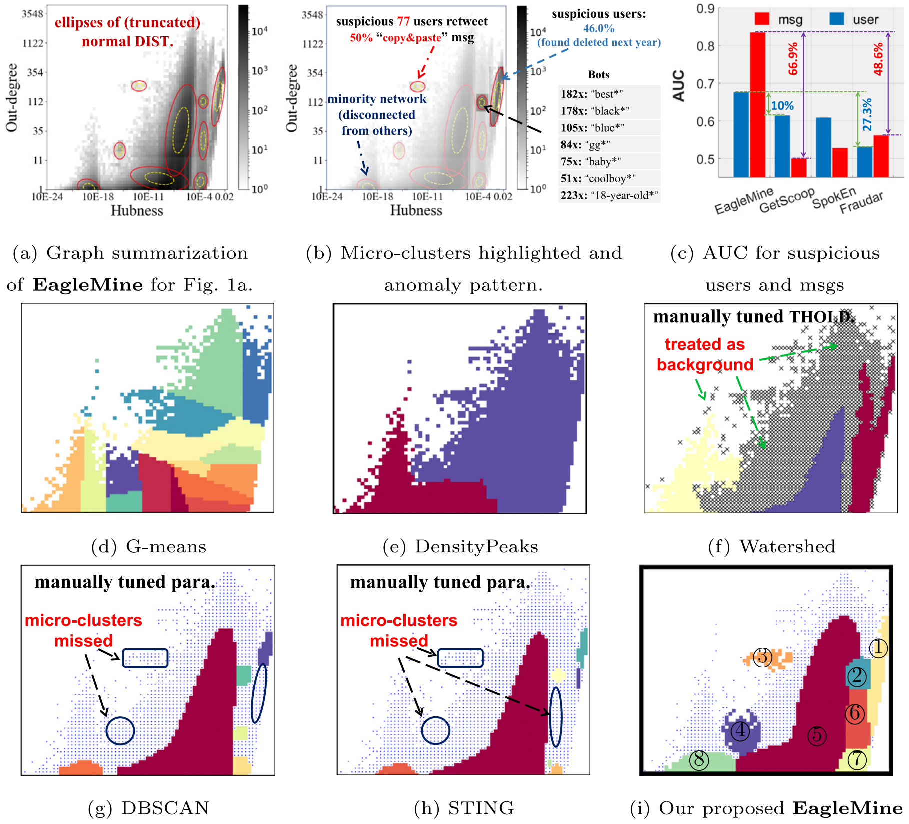
2. Related work

In this section, we review previous related work from three aspects, including the clustering method, vision-based data mining, and anomaly detection.

2.1. Clustering methods

For the Gaussian clusters, K-means, X-means [9], G-means [10], and BIRCH [11] (which is suitable for spherical clusters) suffer from being sensitive to outliers. Those methods are distance-based and prefer to cluster the points within a short distance in feature space forming a spherical area. Density-based methods, such as DBSCAN [12] and OPTICS [13], are noise-resistant and can detect clusters of arbitrary shape and data distribution, while the clustering performance relies on the density threshold for DBSCAN, and also for OPTICS to derive clusters from reachability-plot. RIC [14] enhances other clustering algorithms as a framework, using minimum description language as the goodness criterion to select fitting distributions and separate noise. STING [15] hierarchically merges every four grids in lower layers to find clusters with a given density threshold. Density-Peaks [16] defines the cluster center as the point with high local density and away from other centers, it selects the cluster centers based on the decision graph then puts remaining points into the nearest cluster, while it needs to compute the pairwise distance of points which is not suited to large-scale data and needs to manfully determine the number of cluster centers as well. Clustering algorithms [17] derived from the watershed transformation [18] treat the pixel regions between watersheds as one cluster, they only focus on the final results and ignore the hierarchical structure of clusters. [19] compared different clustering algorithms and proposed a hierarchical clustering method, "HDBSCAN", while its complexity is prohibitive for very large datasets (like graphs) and the "outlierness" score does not line with our expectations. In

[1] One of the largest micro-blog websites in China.



(a) Graph summarization of **EagleMine** for Fig. 1a.

(b) Micro-clusters highlighted and anomaly pattern.

(c) AUC for suspicious users and msgs

(d) G-means

(e) DensityPeaks

(f) Watershed

(g) DBSCAN

(h) STING

(i) Our proposed **EagleMine**

Fig. 2. Our proposed **EagleMine** achieves effective results on micro-blog Sina Weibo data. (a) EagleMine summarizes the graph nodes in a feature space of Fig. 1a with truncated Gaussian distributions. The ellipses illustrate $1.5 \cdot \Sigma$ and $3 \cdot \Sigma$ contours of Gaussians, where Σ is the covariance matrix. (b) highlights some micro-clusters, including a disconnected small network and very suspicious micro-clusters. A user name list on the right side shows the name patterns of bots in a micro-cluster, where 182x: "best*" means 182 bots share prefix "best". (c) EagleMine achieves the best AUC \ ROC (Area Under the Receiver Operating Characteristic Curve) for detecting suspicious users and messages on Sina Weibo compared to the state-of-the-art competitors. (d)–(h) are comparison result of baselines and EagleMine for recognizing the node groups. EagleMine outperforms others by visual comparison. Watershed (with threshold THOLD. for image background), DBSCAN, and STING are manually tuned to have relatively better results. The blue scatter points in (f)–(h) denote individual outlier nodes. Even though DBSCAN and STING are extensively tuned, some micro-clusters of low density are missed. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

addition, community detection algorithms [20,21], modularity-driven clustering, and cut-based methods usually cannot handle large graphs with million nodes or fail to provide an intuitive or interpretable result when applying to graph clustering.

2.2. Vision-based data mining

Supported by human vision recognition theory, including visual saliency, color sensitivity, depth perception, and attention of vision system [22], visualization techniques [23] and HCI tools help to get insight into data [24,25]. Scagnostic [3,26] diagnoses anomalies from the plots of scattered points. Net-Ray [4] visualizes and mines adjacency matrices and scatter plots of a large graph, and discovers some interesting patterns. [27] constructs focus-plots (pairwise feature plots) from a few feature subspaces and proposes LookOut for to give pictorial explanations of outlying multi-dimensional behavior. GraphVis [25,28] introduce a flexible web-based network visual analytics platform which combines interactive visualizations with analytic techniques to revealing important patterns and decision-making etc.

2.3. Anomaly detection

As for anomaly detection, [6] gives a comprehensive survey about the different categories of anomalies, its applications, and research areas, there also have been diverse approaches for anomaly detection in graphs (see [29] for a survey); the collective anomaly has also been explored in different applications, e.g., mobile service [30], social network [31,32] and network traffic [33], etc. Recently, [34,35] find communities and suspicious clusters in a graph with spectral-subspace plots. SpokEn [34] considers the "EigenSpokes" pattern of community in the EE-plots produced by pairs of eigenvectors of a graph, which is applied to fraud detection. GetScoop [35] can find the dense subgraph in a bipartite graph with local search; VoG [36] used some predefined structures as vocabulary to summary the large graph and TimeCrunch [37] extend it by incorporating time signatures for the dynamic graph. As more recent works, dense block detection has been proposed to identify anomalous patterns and suspicious behaviors [38–41]. A representative work, Fraudar [40], proposed a densest subgraph detection method that incorporates the suspiciousness score of nodes and edges during optimization. CatchCore [41] detected hierarchical dense subtensors to catch the anomaly patterns for the multi-aspects data in a multi-resolution matter.

Besides, graph mining techniques have wide applications, including structure pattern detection [42], node classification [43,44], anomaly detection [45,46], and more. With the development of deep learning [47] and graph neural network [48], they provide useful tools for many graph-mining problems based on their powerful representation ability and can also applied to different fields, e.g., time series analysis [49], image classification [50],

Table 1
Comparison of EagleMine and relevant methods. ✓ denotes 'supported'.

	DensityPeaks [16]	DBSCAN [12]	STING [15]	Watershed [17,18]	SpokEn [34]	GetScoop [35]	Fraudar [40]	**EagleMine**
Parameter free					✓	✓	✓	✓
Non-spherical cluster	✓	✓	✓	✓				✓
Anomaly detection		✓	✓		✓	✓	✓	✓
Summarization			✓					✓
Linear in #nodes				✓			✓	✓

and knowledge graph [51], while limited to some supervised or semi-supervised tasks.

A comparison between EagleMine and the majority of the above methods is summarized in Table 1. Our proposed method EagleMine is the only one that matches all specifications.

3. Model

Consider a graph $\mathcal{G}$ with node set $\mathcal{V}$ and edge set $\mathcal{E}$, it can be either homogeneous (monopartite), such as friendship or following relations in the social network, or bipartite for users rating restaurants.

Which ones should be extracted to characterize the graph nodes from the possibly infinite set of features? Intuitively, we expect to select these features that (a) are fast-to-compute; (b) can describe some patterns or laws that most nodes obey, except some anomalous nodes, in a large graph. Thus, the potentially useful features include (in-, out-) degree, # triangle, coreness, spectral vectors, and PageRank, which inherently measure the importance of nodes in a graph; some time-related features can also be used in other scenarios.

As stated in the introduction, given a histogram constructed from some feature spaces, our goal is to recognize node groups that are consistent with human vision does and optimize the goodness-of-fit (GoF) of a model for the node distribution in each group. So, we map those items (e.g., graph nodes) into a histogram that may be multi-dimensional consisting of multiple features.

As the two-dimensional histogram shows (see Fig. 1), one can naturally expect that the adjacent bins with similar color should be in one group by the intuitive visual recognition. From the view of clustering, the node group with high density forms a cluster and there are distinctions between different clusters, i.e., they are separated by different low-density regions. Also, these node groups defined by different density thresholds have a nest or hierarchical structure as shown in the areas with different colors.

Consider the histogram (heatmap) $\mathcal{H}$, its dimension is denoted as F and the number of non-empty bins is $nnz(\mathcal{H})$. We use $\boldsymbol{b}$ to denote any bin in $\mathcal{H}$ and h to denote the number of nodes in a bin.

Model: To summarize the histogram $\mathcal{H}$ more succinctly corresponding to a feature space of graph nodes, we utilize some statistical distributions as the vocabulary with some characteristic parameters to describe the density and randomness properties of nodes within its group of $\mathcal{H}$. Therefore, our vocabulary-based summarization model of graph nodes consists of

- **Configurable vocabulary:** statistical distributions $\mathcal{Y}$ for describing node groups of $\mathcal{H}$ in a feature space.
- **Assignment variables:** $\mathcal{S} = \{s_1, \ldots, s_C\}$ for the distribution assignment of C node groups (clusters).

Algorithm 1 EagleMine

```
Input: Histogram $\mathcal{H}$ of a specific feature space.
Build a hierarchical tree $\mathcal{T}$ for node groups in $\mathcal{H}$ with
    WaterLevelTree algorithm.        ▷ see Section 4.1
Explore $\mathcal{T}$ and search the optimal summarization for the $\mathcal{H}$
    with TreeExplore algorithm.        ▷ see Section 4.3
```

- **Model parameters:** $\Theta = \{\theta_1, \ldots, \theta_C\}$ for distributions in each node group. E.g., the mean and variance for normal distribution.
- **Outliers:** unassigned bins $\mathcal{O}$ in $\mathcal{H}$ for some outliers.

In terms of the configurable vocabulary $\mathcal{Y}$, it can include any suitable or application-depended distribution, such as the Uniform, Gaussian, Laplace, and Exponential distributions or others that can be tailored to the data and characteristics to be described.

Moreover, based on the characteristic of node distribution in feature space, we mainly focus on some *collective anomaly patterns* [6], that is, a collection of data instance as a whole deviates significantly from the entire dataset. Follow the assumption that most of the items belong to normal data and form a large and dense cluster, while anomalies either belong to small or sparse clusters and they are corresponding to some micro-clusters in a histogram; thus, we measure their anomaly with a formulated suspicious score according to the degree of deviation to the normal.

It should be noticed that the dimension of the histogram should not be very large due to the limitation of the sparsity of node connection and the complexity of geometric properties of distributions in the high dimensional space [52,53]. So, $F \leq 5$ is suggested.

4. Proposed method

Our proposed method, EagleMine, is guided by the following traits and mechanism of human vision and cognitive system:

Trait 1. *Human vision usually detects some connected components, which can be rapidly recognized by eyes despite substantial appearance variation [54,55].*

This motivates us to identify each node group as an internally dense connected area in a histogram where different ones are disjoint from each other; it also guides the refinement steps for smoothing.

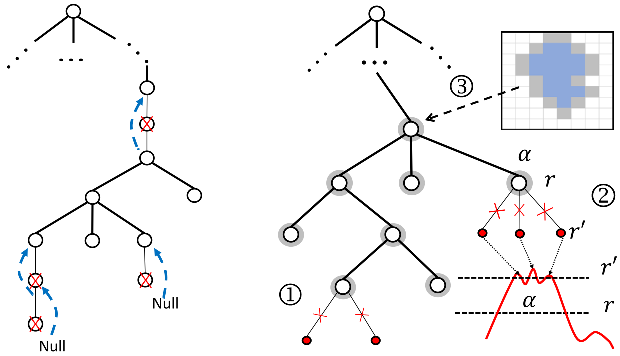
Trait 2. *Top-to-bottom recognition and hierarchical segmentation [56] characteristics. Humans organize basic elements (e.g., words, shapes, visual-areas) into higher-order groupings to generate and represent complex hierarchical structures in human cognition and visual–spatial domains.*

This suggests that organizing and exploring connected node groups should be based on a hierarchical structure, as we will do. Also, the recent work [57] uses the neural network model to study the human visual importance perception and aid data visualization and graphical design.

The overall structure of EagleMine is described in Algorithm 1. Given a histogram $\mathcal{H}$, EagleMine first builds a hierarchical tree $\mathcal{T}$ to organize these micro-clusters (node groups) identified from $\mathcal{H}$ with WaterLevelTree, the tree $\mathcal{T}$ provides an efficient and multi-resolution tool to identify and separate clusters (node group) at different density levels; then it uses TreeExplore algorithm to explore and search the optimal summarization for $\mathcal{H}$, which consists of the model parameters Θ, assignment S for each node group, and outliers index $\mathcal{O}$. The following subsections will elaborate on each step in detail.

Algorithm 2 WaterLevelTree Algorithm

```
Input: Histogram $\mathcal{H}$.
Output: Water-Level tree $\mathcal{T}$.
1: $\mathcal{T}$ = {positive bins in $\mathcal{H}$ as root}.
   // Raw tree construction.
2: for $r = 0$ to $\log h_{max}$ by step $\rho$ do
3:     $\mathcal{H}^r$ : assign $h \in \mathcal{H}$ to zero if $\log h < r$.
4:     $\mathcal{H}^r = \mathcal{H}^r \circ \mathbf{E}$.      ▷ binary opening to smooth.
5:     islands $\mathcal{A}^r$ = {jointed bin areas in $\mathcal{H}^r$}.
6:     link islands in $\mathcal{A}^r$ to its parent at level $r_{prev}$ in $\mathcal{T}$.
7: end for
   // Tree refinement steps.
8: Contract $\mathcal{T}$: iteratively remove each single-child island and
   link its children to its parent.
9: Prune $\mathcal{T}$: heuristically remove noise nodes.
10: Expand islands in $\mathcal{T}$.
11: return $\mathcal{T}$
```



(a) Tree contract (b) Tree prune and island expand

Fig. 3. Refinement steps in the WaterLevelTree Algorithm. (a), (b) correspond to *contract, prune, and expand* for the water-level tree respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

4.1. Water level tree

In the histogram $\mathcal{H}$, we imagine an area consisting of connected positive bins ($h > 0$) as an ***island***, and the other bins as the ***water area***. Assume that we can flood those island areas, making the bins with $h < r$ to be underwater, i.e. setting hs to be 0, where r refers to some specific water level. Afterward, the remaining positive bins can form new islands in the condition of water level r.

To organize all those islands identified in different water levels, we propose a water-level tree structure $\mathcal{T}$, in which each node represents an island and the edge represents the relation that a child island at a higher water-level comes from a parent island at a lower water-level. Note that increasing r from 0 corresponds to raising the water level and flooding from the root to leaves in $\mathcal{T}$.

In a 2D histogram, the islands are candidate node groups for Trait 1; the flooding process intuitively reflects how human eyes hierarchically capture those different objects from the color histogram $\mathcal{H}$ as Trait 2. For example, the gradient color in Fig. 1 depicts groups at different water levels.

This motivates us to identify each node group as an internally connected dense area in a histogram where different ones are disjoint from each other and design refinement for smoothing. The WaterLevelTree algorithm is shown in Alg. 2. It starts from the root and raises the water level r in logarithmic scale from 0 to $\log h_{max}$ ($h_{max} = \max \mathcal{H}$) with a step size ρ,[2] to account for the power-law-like distribution of h. Here, we use the binary opening operator ($\circ$)[3] [58] to smooth each island, which can remove some small isolated bins (treated as noise), and separate weakly-connected islands with a specific structure element (Line 4). Afterward, we link each island at the current level to its parent at the lower water level r_{prev} in the tree (see Fig. 3a) (Line 6). Finally, when r reaches the maximum level $\log h_{max}$, the flooding process stops.

We propose the following steps to refine the raw tree $\mathcal{T}$:

Contract: The current tree $\mathcal{T}$ may contain many nodes with only one child, meaning that no new islands are separated, which are redundant. Hence, we *search the tree using the depth-first search (DFS); we will remove it and link its children to its parent once anyone single-child node is found.*

This process is shown in Fig. 3a, the dashed blue lines with arrow depict that the children of the single-child nodes are linked to its parent, and the gray thin links will be removed.

Prune: The purpose of pruning is to smooth away these noise peaks at the top of some nodes in $\mathcal{T}$, attributing to the fluctuations of h among neighbor bins. Hence, we *prune such child branches (including their descendants) based on a heuristic rule by their total area size: the sum of h in bins of children is no more than 95% that in the parent.*

The pruning branches are illustrated in Fig. 3b (① and ②). Taking ② as an example, the island α at the water-level r contains some fluctuation noises on its top, and these noises will separate into 3 'tiny' islands and link to their parent α in $\mathcal{T}$ when the water-level rises to r'. So, they will be removed with pruning.

Expand: We include some surrounding bins of an island (tree node) to avoid over-fitting for learning the distribution parameters and eliminate the possible effect of the uniform step ρ in the logarithmic scale. So, we *iteratively extend these positive bins around islands by a one-bin step size each time until some touches with other islands or two times the number of bins as contained in the original.*

The island-expansion of $\mathcal{T}$ is illustrated by the circles with shadow in Fig. 3b. For the node ③, the irregularly-shaped blue part depicts the original island and the outer around gray squares are expanded part at the first step; further expansion follows a similar process until it satisfies the above afore-stated constraint.

Comparably in the Watershed formalization [18], the foreground of $\mathcal{H}$ are defined as catchment basins for clustering purposes and can capture the boundaries between clusters as segmentation. We will see in experiments (Section 5.2 and Fig. 2), the segmentation in Watershed approximates the islands in one level of $\mathcal{T}$, with a threshold parameter defined for the background. Thus, it cannot perfectly separate some islands at different levels and cannot handle multi-dimensional data. Another grid-based clustering algorithm STING [15] builds a multi-resolution tree for the histogram bins as an index for quick querying. Clusters are directly achieved by querying the tree-like index with the density threshold as a parameter, while those clusters are the islands at the same level of the water-level tree. Besides, there is no prior knowledge to set the optimal parameters in our scenario; it is also true for DBSCAN even if they are related. Also, other Gaussian-based methods, e.g., G-means and BIRCH, carry out clustering

[2] For raising water levels, it can also use some local search method to determine the optimal step size ρ^* for each step except for the fixed ρ, that is, ρ^* can separate just right the parent node in the tree $\mathcal{T}$ into at least two child nodes.

[3] Binary opening is a basic workhorse of morphological noise removal in computer vision and image processing. Here we use $\underbrace{2 \times \cdots \times 2}_{F}$ square-shape "probe".

with making an explicit assumption about the distribution of data; so they cannot be applied to situations with some complex data distribution, and the results are usually not as good as some density-based methods. However, EagleMine hierarchically identifies clusters at different density levels with no assumption for data distribution, it has no tuning parameters, and it searches the water-level tree to find the best combination of islands, which may come from different levels (see Section 4.3).

4.2. The vocabulary describing the islands

Vocabulary $\mathcal{Y}$ can contain any proper user-defined distributions. For the concerned feature spaces, we use the multivariate Gaussian as one of the vocabulary terms. Noticed that many features of graph nodes, such as degrees and # triangle, typically follow from the power-law. Hence those bins along the histogram boundary have a larger number of nodes (as the bottom in Fig. 1); thus, the truncated Gaussian distribution [59] is a proper choice for describing such truncated ellipses. Moreover, the bins in $\mathcal{H}$ are some discrete units, so the distribution terms must be discretized and it defines the probability function in each bin instead of the probability density function. Therefore, we define the *discretized, truncated, multivariate Gaussian* distribution (DTM Gaussian for short) as follows:

Definition 1 (*DTM Gaussian*)**.** The probability function in a bin $\boldsymbol{b}$ (as denote the boundary of the bin) is

$$P(\boldsymbol{b}; \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\beta}) = \int \cdots \int_{\boldsymbol{\beta}} \psi(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\beta}) d\boldsymbol{x}$$

where $\boldsymbol{x}$ is an F-dimension variable, $\boldsymbol{\beta} \in \mathbb{R}^{F\times 2}$ is the truncation bounds (lower and upper bounds for each dimension) and $\psi(\cdot)$ is the density function of a truncated normal distribution with the mean $\boldsymbol{\mu}$ and co-variance $\boldsymbol{\Sigma}$.

For the 2D histogram, $\boldsymbol{\beta} = [[0, +\infty]; [0, +\infty]]$, the DTM Gaussian is a flexible model for capturing the clusters with different shapes, like line, circle, and ellipse, and truncation. The red–yellow oval circles in Fig. 2a depict some node clusters of DTM Gaussian described with $1.5{\cdot}\Sigma$ and $3{\cdot}\Sigma$.

Observing the multi-mode distribution of islands (a skewed triangle-like island in Fig. 1a), we also add the mixture DTM Gaussian as the other vocabulary term. In our data study, this triangle-like island exists in many different histogram plots and contains the majority of graph nodes. For example, Fig. 1a depicts users' distribution over out-degree and hubness. The power-law of out-degree makes the density decrease along the vertical axis. Meanwhile, the users with similar degrees share similar hubness, forming a nearly normal distribution in horizontal. Therefore, those majority users form a triangle-like island in the feature space, and we use the mixture DTM Gaussian for it.

In general, to decide the assignment $\mathcal{S}$ of vocabulary to each island, we can use the distribution-free statistical hypothesis test, like Pearson's χ^2 test or other distribution specified approaches.

After getting the vocabulary assignment, we use maximum likelihood estimation method to learn the parameters θ_α for an island α, where $\theta_\alpha = \{\boldsymbol{\mu}_\alpha, \boldsymbol{\Sigma}_\alpha, \tilde{N}_\alpha\}$ and $\tilde{N}_\alpha = \sum_{(i_1,\ldots,i_F)\in\alpha} \log h_{i_1,\ldots,i_F}$. For denotation, we also define the function *DistributionFit*(α, s_α) as the learning process which returns θ_α.

4.3. Tree explore algorithm

With the hierarchical water-level tree $\mathcal{T}$ and the describing vocabulary $\mathcal{Y}$, we can then determine the optimal node groups and their summarization. The procedure is shown in Alg. 3, we first decide the distribution vocabulary term s_α for each island (tree node) α, search the tree with BFS, and select the optimal islands with some criteria; refine the results with stitching in final.

Algorithm 3 TreeExplore Algorithm

```
Input: WaterLevelTree $\mathcal{T}$
Output: summarization $\{\mathcal{S}, \Theta, \mathcal{O}\}$.
 1: $\Theta = \emptyset$.
 2: $\mathcal{S}$ = decide the distribution type $s_\alpha$ from vocabulary $\mathcal{Y}$ for
    each island in $\mathcal{T}$.
 3: Queue $\boldsymbol{Q}$ = root node of $\mathcal{T}$.
 4: while $\boldsymbol{Q} \neq \emptyset$ do                         ▷ breadth-first search.
 5:     $\alpha \leftarrow$ dequeue of $\boldsymbol{Q}$.
 6:     $\theta_\alpha$ = DistributionFit$(\alpha, s_\alpha)$            ▷ determine param.
 7:     Hypothesis test $\boldsymbol{H}_0$ = bins of island $\alpha$ come from
        distribution $s_\alpha$.
 8:     if $\boldsymbol{H}_0$ is rejected then
 9:         enqueue all the children of $\alpha$ into $\boldsymbol{Q}$.
10:         $\mathcal{S} = \mathcal{S} \setminus \{s_\alpha\}$
11:     else
12:         $\Theta = \Theta + \{\theta_\alpha\}$
13:     end if
14: end while
15: Stitch and replace the promising distributions in $\mathcal{S}$, then
    update $\Theta$.
16: Decide outliers $\mathcal{O}$ deviating from the recognized groups.
17: return summarization $\{\mathcal{S}, \Theta, \mathcal{O}\}$.
```

To determine the assignment of the distribution term, we heuristically assign the mixture DTM Gaussian to the island containing the largest number of graph nodes at each tree level, and the DTM Gaussian to other islands for simplicity.

BFS search and Selection criteria: Afterward, we search along the tree $\mathcal{T}$ with BFS to select the optimal combination of node groups (see steps 4 to 14). Starting from the root, we can determine whether to explore the children of a node according to some criteria.

In statistics and machine learning, AIC and BIC are criteria used for model selection among a finite set of models in the regularization framework; the model with the lowest score is preferred [60]. Thus we can use those criteria to determine the target island: if the score of a parent node is less than the sum of scores of its children, will we stop searching and select this parent island, otherwise, we will continue to search.

Another criterion — the statistical hypothesis tests select models by measuring the statistically significant of idealized null hypothesis [10,61], such as the distribution-free tests Pearson's χ^2 test and K–S test, Anderson–Darling test for Gaussian distribution, and also others. The null hypothesis for searching the children of the island α in $\mathcal{T}$ is:

$\boldsymbol{H}_0$: the bins of island α come from distribution s_α.

If $\boldsymbol{H}_0$ is not rejected, we stop searching for the children of the island α; otherwise, we further explore its children.

We utilize the Anderson–Darling statistic test, compared with Pearson's χ^2 test, AIC, and BIC criteria being unstable due to the extreme variability of the value in histogram bins. Thus, we test an island based on its binary image by focusing on whether the island's shape looks like a truncated Gaussian or mixture. We simplify the hypothesis test by projecting the data of bins into some dimensions and apply the hypothesis test according to projection pursuit [62] and G-means [10]. We implement the Quadratic class 'upper tail' Anderson–Darling Statistic test[4] [63]

[4] It measures the GoF of the left-truncated Gaussian distribution.

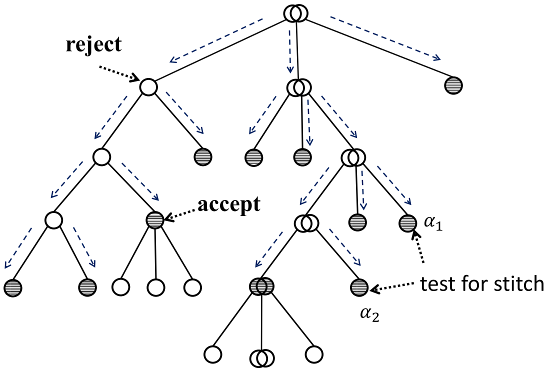


Fig. 4. The optimal island candidates search and the stitching post-process in TreeExplore Algorithm. The dashed lines with an arrow along with the edges of the tree denote BFS search, the dashed circles denote the final optimal candidates of node groups selected by the statistical hypothesis test.

(with the 1% significant level) due to the truncation. And we accept the null hypothesis $\boldsymbol{H}_0$ only when the test is true for all dimension projections; will $\boldsymbol{H}_0$ be rejected once one of them is rejected. Finally, we get the node group candidates to summarize the histogram when the BFS stops. The dashed lines with an arrow in Fig. 4 demonstrate the search trace and the shaded circles are the final optimal candidates of islands.

Stitch: Furthermore, some islands from different parents, e.g., α_1 and α_2 in Fig. 4, are physically close to each other. In such a case, those islands can probably come from the same distribution. Therefore, we use the *stitching* process in step 15 to merge them by the hypothesis test as well. The stitching process stops until no changes occur. When there are many pairs of islands can be merged at the same time, we choose the one with the least average log-likelihood reduction after stitching:

$$(\alpha_{i^*},\alpha_{j^*}) = \arg\min_{i,j} \frac{\mathcal{L}_i+\mathcal{L}_j-\mathcal{L}_{ij}}{\#\text{points of } \alpha_i \text{ and } \alpha_j} \qquad (1)$$

where α_i and α_j are the pair of islands to be merged, $\mathcal{L}_{(\cdot)}$ is log-likelihood of a island, and $\mathcal{L}_{ij}$ is the log-likelihood of the merged island.

Outliers and Suspiciousness score: The outlier comprises of those bins far away from any distribution of the identified node groups (i.e., with the probability less than $\delta = 10^{-4}$).

Besides, the major island containing the most nodes is normal based on the implicit assumption that the normal instances are far more frequent than anomalies in data for the unsupervised mode [6]. So, we define the weighted KL-divergence of an island to the major one as its suspiciousness score:

Definition 2 (*Suspiciousness*)**.** Given the parameter θ_m for the major island, the suspiciousness of the island α_i is described by a distribution with the parameter θ_i is:

$$\kappa(\theta_i) = \log \bar{d}_i \cdot \sum_{\boldsymbol{b}\in\alpha_i} N_i \cdot \mathrm{KL}(\, P(\boldsymbol{b} \mid \theta_i) \parallel P(\boldsymbol{b} \mid \theta_m)\,) \qquad (2)$$

where $P(\boldsymbol{b} \mid \theta)$ is the probability of bin $\boldsymbol{b}$ for a distribution with θ as the parameter, N_i is the number of nodes in the island i, and we use the logarithmic $\bar{d}_i$, the average degree of all the graph nodes in i, as a weight, based on the domain knowledge that higher-degree nodes are more suspicious if all else being equal.[5]

Table 2
Summary of the real world dataset.

	# of nodes	# of edges	Content/Relation
Amazon	(2.14M, 1.23M)	5.84M	Rate
Android	(1.32M, 61.27K)	2.64M	Rate
BeerAdvocate	(33.37K, 65.91K)	1.57M	Rate
Yelp	(686K, 85.54K)	2.68M	Rate
Flickr	(2.30M, 2.30M)	33.14M	User to group
Tagged	(2.73M, 4.65M)	150.8M	Anonymized links
Youtube	(3.22M, 3.22M)	9.37M	Who-follow-who
Sina Weibo	(2.75M, 8.08M)	50.1M	User-retweet-msg

4.4. Time complexity

Given the features associated with nodes $\mathcal{V}$, the time complexity for generating histogram $\mathcal{H}$ is $O(|\mathcal{V}|)$.

Let $\mathcal{C}$ be the number of clusters, assume that T is the number of iterations for learning the parameters in *DistributionFit*$(\cdot)$ by gradient-descent of Alg. 3, it is related to the differences between initial and optimal objective values. With $h_{max} = \max \mathcal{H}$ as defined before, then we have:

Theorem 1 (*Time Complexity*)**.** *The time complexity of EagleMine is*

$$O(\frac{\log h_{max}}{\rho} \cdot nnz(\mathcal{H}) + \mathcal{C} \cdot T \cdot nnz(\mathcal{H})). \qquad (3)$$

Proof. In EagleMine, to build $\mathcal{T}$, WaterLevelTree compares all $nnz(\mathcal{H})$ non-empty bins with water level r in Line 3, and then do binary opening; to remove small blobs (noise) by checking non-empty ones in Line 4, both of which cost $O(nnz(\mathcal{H}))$. From Line 5 to 6, for each island, we connect its children (# of islands $< nnz(\mathcal{H})$) to it, so the time cost equals to the number of links, i.e. $O(nnz(\mathcal{H}))$. Hence the whole iteration from Line 2 to 7 takes $O(\tau \cdot nnz(\mathcal{H}))$, where $\tau = {}^{\log h_{\max}}/_{\rho}$. As a result, we get a tree $\mathcal{T}$, whose height is τ and width is at most $nnz(\mathcal{H})$. The total number of links in that tree are less than $\tau \cdot nnz(\mathcal{H})$). Afterward, the operation of contracting takes $O(\tau \cdot nnz(\mathcal{H}))$. In each tree level, the summation of bins in islands is less than $nnz(\mathcal{H})$, so the complexity of both pruning and expanding process is also $O(\tau \cdot nnz(\mathcal{H}))$. Consequently, the costs of constructing the water-level tree $\mathcal{T}$ is $O(\tau \cdot nnz(\mathcal{H}))$.

During the search of the optimal node groups in the TreeExplore algorithm, the cost of *DistributionFit*$(\cdot)$ is $O(T \cdot nnz(\mathcal{H}))$. Since our algorithm finds C micro-clusters when stops, the subtree with visited nodes by BFS search on $\mathcal{T}$ has C leaves. Due to the contraction of WaterLevelTree at Line 8, each non-leaf node in the subtree has at least two children, hence the subtree has at most $2 \cdot C$ nodes, which means the steps from Line 4 to 14 have at most $2 \cdot C$ times of choosing the largest island, conducting *DistributionFit*$(\cdot)$, and applying hypothesis tests. The cost of the statistical hypothesis test on each node (island) is linear to the number of bins in the island, which is less than $nnz(\mathcal{H})$. During stitching, we only test those islands close to each other in a plane, which costs less than the above process on tree $\mathcal{T}$.

Therefore, the time complexity of EagleMine is

$$\begin{aligned} & O(\tau \cdot nnz(\mathcal{H}) + 2C \cdot (T \cdot nnz(\mathcal{H}) + nnz(\mathcal{H}))) \\ = \; & O(\frac{\log h_{max}}{\rho} \cdot nnz(\mathcal{H}) + C \cdot T \cdot nnz(\mathcal{H})) \end{aligned}$$

where $C \ll nnz(\mathcal{H})$. ■

5. Experiments

We designed and conducted experiments to answer the following questions:

[5] In general cases, we can use the sum of the average deviation of each feature with normalization to be the weight as a contribution to the suspiciousness.

- **Q1. Qualitative interpretation (vision-based):** Does EagleMine accurately identify micro-clusters that agree with human vision recognition?
- **Q2. Quantitative evaluation:** Does EagleMine give a significant improvement in concisely summarizing the graph?
- **Q3. Anomaly detection accuracy:** How does EagleMine's performance on anomaly detection in real-world graphs, compared with the state-of-the-art methods? How much improvement does the visual-inspired information bring?
- **Q4. Generalization:** Does EagleMine can be generalized to mine interesting patterns beyond the graph data?
- **Q5. Scalability:** Is EagleMine scalable with regard to the data size? Dose EagleMine scale out?

5.1. Experiment settings

5.1.1. Datasets

The dataset information used in our experiments is summarized in Table 2. The Amazon [64], Android [65], BeerAdvocate [66], and Yelp [67] datasets are about user-rate-item relations, where the item are products, Apps, beers, and food respectively. The Flickr [68] data is about the user belong to group relations. The Youtube [69] forms a homogeneous graph representing the following relation between users. The Tagged [5] dataset was collected from Tagged.com, a social network website; among the seven anonymized types of links between users, we chose the type-6 with the most number of edge and constructed a homogeneous graph. The micro-blog Sina Weibo dataset was crawled in Nov. 2013 from weibo.com, consisting of a user-retweet-message (bipartite) graph.

5.1.2. Implementations

We choose various clustering algorithms as baselines to compare to EagleMine, including X-means [9], G-means [10], DBSCAN [12], STING [15], and DensityPeaks [16], etc., their settings are listed as follows. The HDBSCAN [19] is omitted for its $O(an^2)$ time complexity, which is prohibitive for our large graphs, where a is the number of attributes describing the objects and n is the number of nodes in a graph, that is, $\sum_{h\in\mathcal{H}} h$.[6]

- X-means: initialize with k-means and 5 clusters.
- G-means: set $max\_depth = 5$, limiting no more than 16 clusters to avoid too many clusters; set the p-value $= 0.01$ which is insensitive.
- DBSCAN: set Eps = 1, and use Manhattan distance function (between bins); we searched for the MinPts from the average number of nodes in bins of a histogram to the max number by a step size = 50, and manually select the one consistent well with human visual recognition.[7]
- STING: initialize $c \approx \frac{\text{Minpts}+1}{\pi\,\text{Eps}^2}$ with DBSCAN's tuned optimal MinPts and Eps for clusters, and refine the visual result by fine-tuning.
- DensityPeaks: use Gaussian kernel and select the best number of clusters based on the "decision graph" [16].
- EagleMine and EagleMine_DM: are our proposed EagleMine with DTM Gaussian and whole multivariate Gaussian respectively.

In terms of the feature spaces, we chose the degree vs. PageRank and degree vs. triangle for the Tagged dataset, while the in-degree vs. authority and out-degree vs. hubness for the other datasets.

To construct the histogram for different node features, the bandwidth (i.e., bin size) for each feature can be selected according to the plug-in methods or kernel density-estimators [72]. In our settings, we use a heuristic rule as [2] by dividing a discrete feature, such as degree, number of triangles, etc., into bins with fixed bandwidth in the logarithmic scale, and for other continuous features, like spectral features (hubness and authority), are evenly divided into similar sizes in the logarithmic scale as well. We use a fixed step size for raising water levels for EagleMine in our experiments, i.e., set $\rho = 0.2$, and it is universally suitable for all datasets from different applications.

5.2. Q1. Qualitative evaluation (vision-based)

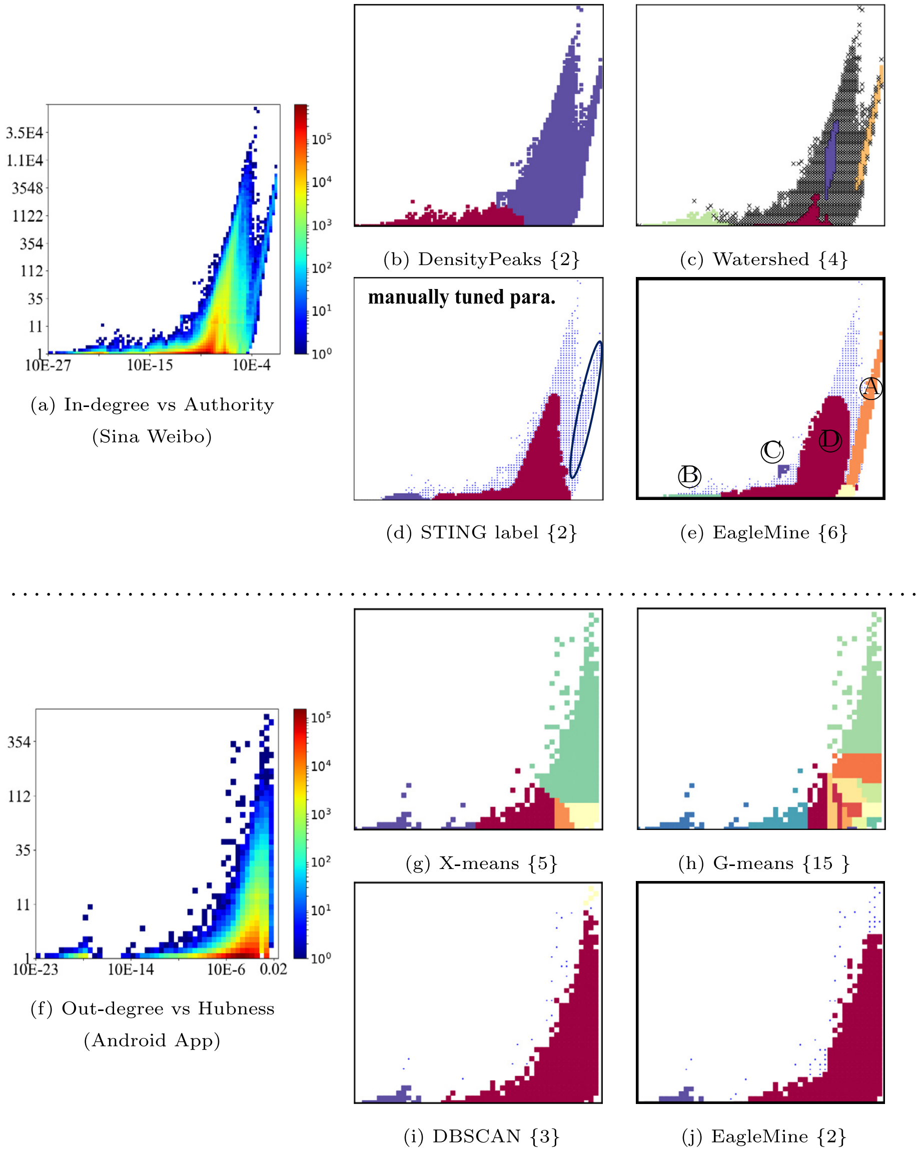
In this part, we illustrate the results of the two-dimensional histogram for vision-based qualitative comparison. We exhibit the results of the comparison of different baselines and different feature spaces in an alternative way due to the space limit. We chose X-means, G-means, DensityPeaks, DBSCAN, STING, and Watershed as the baselines. The feature spaces include out-degree vs. hubness, in-degree vs. authority, and #triangle vs. degree.

Figs. 2d–2i show the results on the user-retweet-message graph of Sina Weibo data. The features are user's out-degree and hubness indicating how many important messages are retweeted. As we can see, the DensityPeaks algorithm has poor cluster results since it fails to find all micro-clusters and outliers. Without removing some low-density parts as the background, the Watershed algorithm treats all the groups as one or two large clusters; hence we manually tuned the threshold of background to attain a better result, which is similar to the groups in some level of our water-level tree, and the final background is shown with the gray color in Fig. 2f; however, it only recognized a few very dense groups while failing to separate the two groups on the right and leaving other micro-clusters unidentified. Our EagleMine recognized node groups more intuitively and also identified those micro-clusters missed by DBSCAN and STING. Note that an unusually high portion of users in the missed micro-clusters ① and ③ had been deleted, they were suspended by the system operators for anti-spam. Besides, the micro-clusters ③ and ④ include users who had high out-degree but low hubness scores, i.e., they retweeted many un-important messages (e.g., advertisements). So, EagleMine identifies very useful micro-clusters automatically as human vision does.

Moreover, Figs. 5 and 6 illustrate more examples (4 groups) of different datasets and feature spaces. The original histogram plots are given at the beginning of each group and followed by the clustering result (labeled with the # cluster) of different methods. Different colors represent different clusters (node groups) for each method; the outliers (bins) are labeled with the *blue* point or 'x' marker. Hence, we can see that G-means and X-means produce several node groups and over-separate those recognized by the human vision; DensityPeaks under-separates clusters and cannot find micro-clusters and outliers. Although manually tuned DBSCAN and STING can capture the majority of dense regions in each histogram, they also overlook some suspicious micro-clusters, e.g., micro-clusters Ⓐ and Ⓒ in Fig. 5e. Thus, EagleMine illustrates its advantages for recognizing groups, especially identifying some micro-clusters, which are more consistent with human vision recognition.

[6] The available open source implementation/packages [70,71] do not support clustering for the weighted data as the histogram.

[7] Since DBSCAN is manually tuned, we do not use OPTICS to search parameters for it.



(a) In-degree vs Authority (Sina Weibo)

(b) DensityPeaks {2}

(c) Watershed {4}

(d) STING label {2}

(e) EagleMine {6}

(f) Out-degree vs Hubness (Android App)

(g) X-means {5}

(h) G-means {15 }

(i) DBSCAN {3}

(j) EagleMine {2}

Fig. 5. EagleMine visually recognizes better node groups than baselines in qualitative comparison for different feature spaces. The label plots reflect the node groups recognized by each algorithm, and '{·}' gives the number of clusters (node groups). (a)–(e): use Sina Weibo data which are message nodes corresponding to user nodes in Fig. 2. (f)–(j): use Android apps' rating data. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

5.3. Q2. Quantitative evaluation

We use the Minimum Description Length (MDL) as the metric to measure the summarization performance as [14] did, by envisioning the problem of clustering as a compression problem. In short, it follows the assumption that the more we can compress the data, the more accurate we can learn about its underlying patterns, and the best model achieves the shortest MDL encoding length. With the MDL principle, the encoding length of EagleMine is:

$$L = \log^*(C) + L_{\mathcal{S}} + L_{\Theta} + L_{\mathcal{O}} + L_{\epsilon}. \tag{4}$$

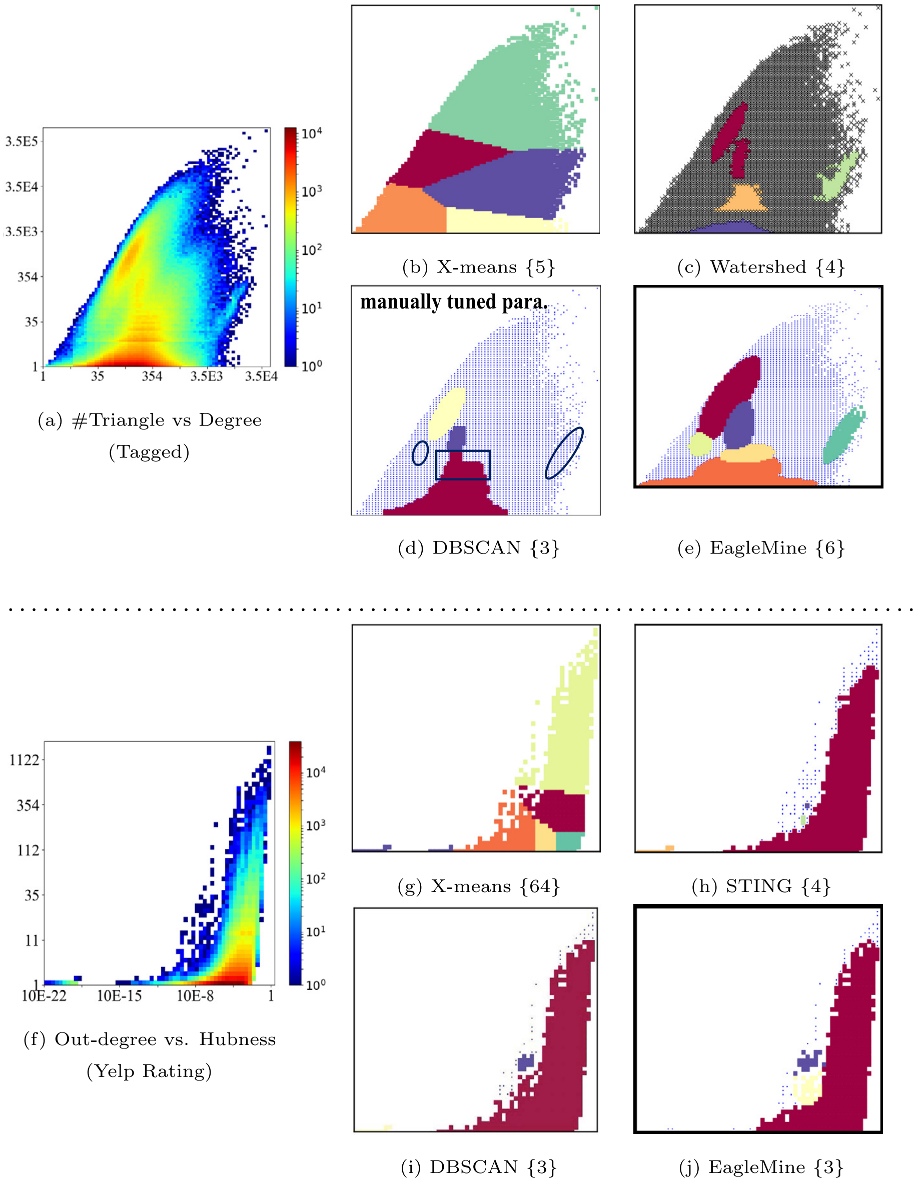
This model description consists of the following terms:

- The number of clusters requires $\log^*(C)$ bits.[8]
- The assignment $\mathcal{S}$ of distribution vocabulary for C node groups requires $L_{\mathcal{S}} = C \cdot \log(\mathcal{Y})$ bits.
- Each DTM Gaussian has $|\theta| = F + \frac{(1+F)F}{2} + 1$ free parameters, e.g. $|\theta| = 6$ for 2D distribution for two features. So, its encoding requires $|\theta| \cdot l_0$ bits, where l_0 is the floating point cost and we used 4×8 bits in our setting. The total encoding length for parameters is $L_{\Theta} = C|\theta| \cdot l_0$ bits.
- The outliers $\mathcal{O}$ require $L_{\boldsymbol{o}}$ bits to encode the bin indices.
- The model error requires L_{ϵ} bits. For a bin $\boldsymbol{b}$ in the island α_i, the expected number of nodes is $\tilde{h} = \left\lfloor 2^{\tilde{N}_i \cdot P(\boldsymbol{b}|\theta_i)} \right\rfloor$, the original count can be accurately recovered as $h = \tilde{h} + \epsilon$. Thus, we encode the total description error as $L_{\epsilon} = \sum_{\boldsymbol{b}} (\log^*(h - \tilde{h}) + 1)$ bits, where 1 is for encoding the sign.

As for the other methods in comparison, we calculated the MDL with the same principle as [14,74].

For these 2D histograms, the comparing results of MDL cost are reported in Fig. 7. We can see that EagleMine achieves the shortest encoding length, indicating a more concise and

[8] Here, $\log^*$ is the universal code length for integers, defined as $\log^*(x) \approx \log_2(x) + \log_2\log_2(x) + \cdots$. where only the positive terms are included in the sum [73].



(a) #Triangle vs Degree (Tagged)

(b) X-means {5}

(c) Watershed {4}

(d) DBSCAN {3}

(e) EagleMine {6}

(f) Out-degree vs. Hubness (Yelp Rating)

(g) X-means {64}

(h) STING {4}

(i) DBSCAN {3}

(j) EagleMine {3}

Fig. 6. EagleMine visually recognizes better node groups in qualitative comparison. (a)–(e): are for the homogeneous graph from Tagged website (#triangle vs. **degree**). (f)–(j): use Yelp rating data (out degree vs. hubness).

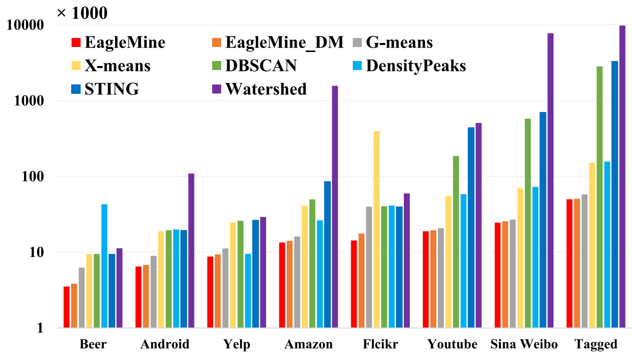


Fig. 7. Quantitative performance comparison for EagleMine and baselines. MDL is compared on different real-world datasets. EagleMine achieves the shortest description code length, which means concise summarization, and outperforms all other baselines.

Table 3
EagleMine's summary for multi-dimensional feature spaces.

Features	Dataset	Dimension of feature space		
		3	4	5
Out-degree + top-k hubness	Amazon	71,029	106,029	201,696
	Yelp	41.391	46,971	59,267

better summarization. On average, EagleMine reduces the MDL cost more than 81.6%, 79.0%, 65.5%, 20.2% compared with STING, DBSCAN, X-means, and G-means respectively. EagleMine also outperforms EagleMine_DM over 4.6% benefiting from a proper vocabulary selection. Therefore, EagleMine summarizes the histogram by those recognized groups in the shortest description length.

For the multi-dimensional histogram, we conducted EagleMine in 3-, 4- and 5-dimensional features spaces on Amazon and Yelp datasets, and used the out-degree vs. top-k hubness (top-k left singular vectors) as features. Here we only used digitized

Table 4
Suspiciousness ranking of found micro-clusters in Sina Weibo dataset.

Feature space	Suspiciousness score $\kappa(\cdot)$ rankings
Out-degree vs. Hubness	①, ②, ③, ④, ⑥, ⑦, ⑧, ⑤
In-degree vs. Authority	Ⓐ, Ⓒ, Ⓑ, Ⓓ

multivariate Gaussian as the vocabulary for simplicity, i.e., EagleMine_DM. The summarization results are listed in Table 3. We can roughly conclude that including more features leads to larger description cost for summarizing the corresponding histograms.

5.4. Q3. Anomaly detection

To demonstrate EagleMine can effectively detect the anomalous, we conducted experiments on both synthetic and real-world data, and compare the performance with state-of-the-art fraud detection algorithms, including SpokEn [34], GetScoop [35], and Fraudar [40].

In the synthetic case, we injected some different size blocks (sub-graph) as fraud (ground truth) into the real datasets. Considering the smart attacks, we also use the random camouflage strategy to inject blocks with 50% camouflage-ratio, i.e. randomly selecting different camouflage objects as many as the targets of fraud. For BeerAdovate, the density of the injected block is 0.05 and the block sizes are $1k \times 500$ and $2k \times 1k$. For Flickr, the densities of blocks are $\{0.05, 0.1, 0.2\}$ and the sizes are $2k \times 2k$ and $4k \times 2k$. We use the F1 score to measure the accuracy of detecting the nodes of injected blocks. Fig. 8a shows the average result over the above trials for each dataset, where GetScoop and SpokEn are omitted for catching nothing. It is obvious that EagleMine consistently outperforms Fraudar and achieves less variance for all the injection types with or without camouflages.

To verify that EagleMine accurately detects anomalies in the Sina Weibo dataset, we labeled these nodes, both user and message, from the results of baselines and sample nodes[9] of our suspicious clusters from EagleMine. Our labels were based on the following rules:

- user-accounts/messages which were deleted from the online system (system operators found as the spams).[10]
- a lot of users that share unusual login-names prefixes, and other suspicious signals: approximately the same sign-up time, the number of friends and followers.
- messages about the advertisement or retweeting promotion, or having lots of *copy-and-paste* text content.

In total, we labeled 5474 suspicious users and 4890 suspicious messages.

EagleMine returns the micro-clusters with suspiciousness score. Table 4 lists the ranking orders of micro-clusters identified by EagleMine for Figs. 2i and 5e according to their suspiciousness scores.

The anomaly detection results are reported in Fig. 2c. Using the AUC (area under the Receiver Operating Characteristic (ROC) curve) to measure the quality of the ordered detection result of each algorithm, the sampled nodes from these micro-clusters are ranked in descent order of hubness or authority. The results show that EagleMine achieves more than 10% and about 50% improvement for detecting anomalous users and messages respectively, and consistently outperforms other baselines. Furthermore, the anomalous users identified by Fraudar and SpokEn only fall into the micro-cluster ① in Fig. 2i due to that they simply focus on the densest core in a graph, but our EagleMine detects suspicious objects by recognizing all those noteworthy micro-clusters in the whole feature space. Therefore, EagleMine can spot more anomalies than the baselines, e.g., some extra micro-clusters ②, ③, and ④ in the feature space.

5.5. Q3. Case study and patterns we found

As discussed above, the micro-clusters ③ and ④ in the out-degree vs. hubness feature space of Fig. 2i contain those users frequently retweet some un-important messages. Also, we study the behavior patterns of the users in micro-clusters ① and ② on the right side of the major node group and find that almost half of them have been deleted by the system operator, and many existing users share unusual names prefixes as Fig. 2b shown, like the 'best*', 'baby*', and '18-year-old*', etc.

What patterns have we found? Fig. 8b shows the 'Jellyfish' structure of the subgraph consisting of users from the micro-clusters ① and ② in Fig. 2i. The head of 'Jellyfish' is the densest core (①), where the users created unusual large dense connections to a group of messages, showing high hubness. These users (spammers or bots) aggressively retweeted a similar message collection (i.e. Ⓐ in Fig. 5e). In the meantime, users from ② connected to some of the messages in Ⓐ, and 'copy-and-paste' many advertising messages on a few topics, e.g., 'new game', 'apps in IOS7', and 'Xiaomi Phone', their structure looks like 'Jellyfish' tail. Thus, the bots in ② show lower hubness than those in ① since being of the different spamming strategies, which are overlooked by other density-based detection methods.

5.6. Q4. Generalization

We investigate the ability of EagleMine for detecting other interesting patterns in different scenarios rather than graphs.

Given a set of retweeting activities for the post of several users, how can we spot the collective anomalies and synchronized behavior [2]? ND-SYNC [75] released their data about *Retweet Fraud* detection based on multiple time-related features extracted from the *retweet thread*, reflecting the intra- and inter-synchronicity of fraudsters compared to the honest for their temporal characteristics. This dataset includes 298 users and their 134,022 retweet threads (83,587 for the honest and 50,435 for fraudulent users) from Twitter, each of the thread has a manually verified label.

We tested the performance of EagleMine on the histograms constructed from the following features as examples,

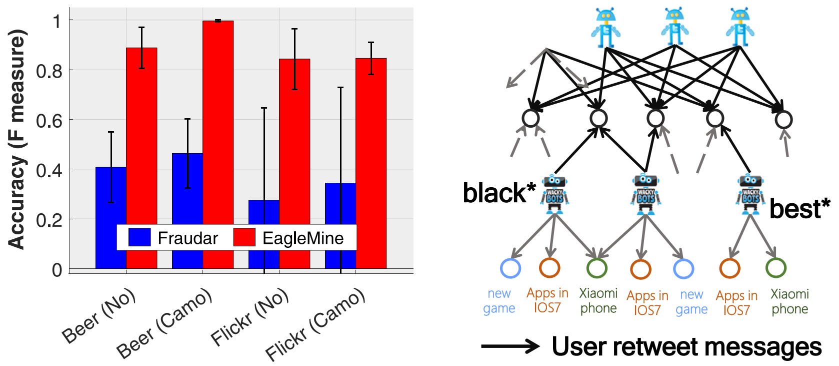
- Retweets: number of retweets
- Response time: time elapsed between the tweets' posting time and its first retweet
- Lifespan: time elapsed between the first and last (observed) retweet
- RT-Q2 response time: tine elapsed after the tweet's posting time to generate the first half of the retweets

Fig. 9 illustrates the detection results of EagleMine. Although there are some more complicated distributions than the graph features, EagleMine accurately identifies most of the vary-type micro-clusters and provides an intuitive summarization. These detected micro-clusters reveal the RT fraudsters' behavior in different feature spaces and their synchronic temporal activity.

As Figs. 9c and 9f show, most of these node groups away from the major island contain a high percentage of fraudsters (as the text marked). Among those micro-clusters, the least one contains 72% anomaly, and others contain 99%–100% items belonging to the anomaly, Compared with the histograms, we can find that the anomalies in micro-clusters have great diversity for different

[9] It is impossible to label all the nodes in graph considering the real limitation.

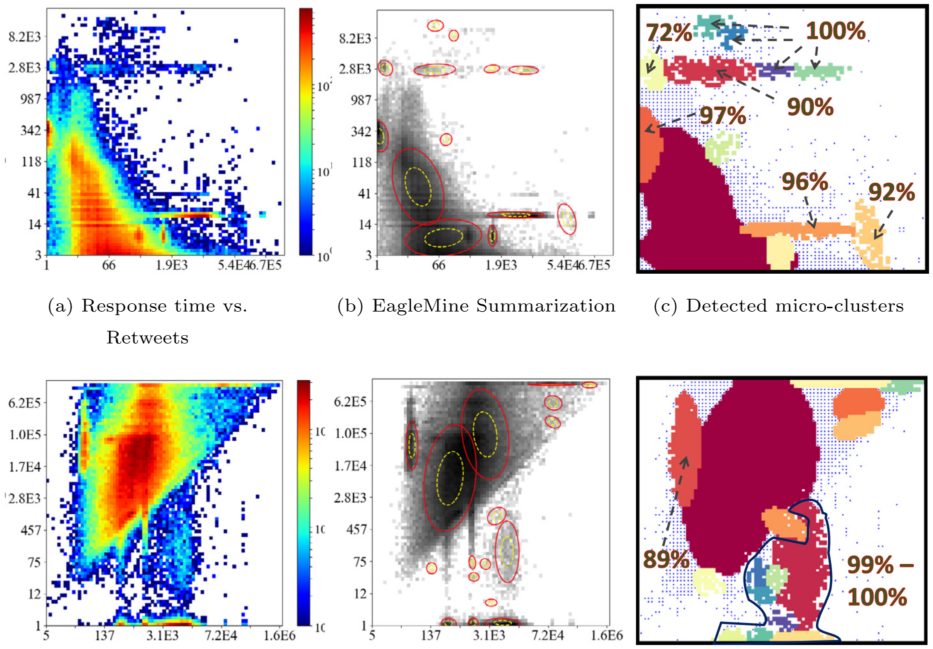
[10] The status is checked 3 year later (May 2017) with API provided by Sina Weibo.



(a) Accuracy for detecting injected fraud.*

(b) Graphical view of anomalies.

Fig. 8. EagleMine's performance for anomaly detection. (a) EagleMine achieves best accuracy for detecting injected fraud for BeerAdvocate ('Beer' as the abbr.) and Flickr data. *Note that GetScoop and SpokEn are omitted for failing to catch any injected object. (b) The anomalous 'Jellyfish' anomaly pattern identified in Sina Weibo.



(a) Response time vs. Retweets

(b) EagleMine Summarization

(c) Detected micro-clusters

(d) RT-Q2 response time vs. Lifespan

(e) EagleMine Summarization

(f) Detected micro-clusters

Fig. 9. EagleMine performance for temporal fraudsters retweet activity. The *retweet threads* are mapped to different feature space. (a): Response time vs. Retweets. (d): RT-Q2 response time vs. Lifespan. (b), (e) show the summarization of EagleMine with DTM Gaussian similar to Fig. 2a (c) and (f) show the detected cluster. Most of the micro-clusters contain a high percentage of fraudsters (marked with text).

features, e.g., the response time can be short as about 14 s or be as long as 300 s and even 2800 s

Therefore, our method can be used for general histogram analysis to detect some micro-clusters and interesting patterns for real-world applications. Besides, network traffic and livestream monitoring are also possible suitable scenarios, we can explore other patterns if related data are available.

5.7. Q5. Scalability

Fig. 10 shows the near-linear scaling of EagleMine's running time in the number graph nodes. Here we used the Sina Weibo dataset and selected the snapshots of the graph, i.e. the reduced subgraphs, according to the timestamp of edge creation in the top 3, 6, . . . , 30 days, to test our method. The slope of the *black dot* line in Fig. 10 indicates the linear growth; EagleMine_DM algorithm, the extension version, is also linearly scalable.

6. Conclusions

In this paper, we propose a tree-based approach EagleMine to mine and summarize all node groups in a histogram of a large graph. The EagleMine algorithm finds the optimal clusters based on a water-level tree and the statistical hypothesis test, and describes them with a configurable vocabulary-based model, and can also spot some suspicious node groups. EagleMine has the following desirable properties:

- **Automated summarization:** EagleMine automatically summarizes a histogram derived from some correlated features with the vocabulary of distributions, and recognizes node groups as human vision does.
- **Effectiveness:** EagleMine detects interpretable micro-clusters and achieves better summarization for a histogram. It outperforms the baselines in both qualitative

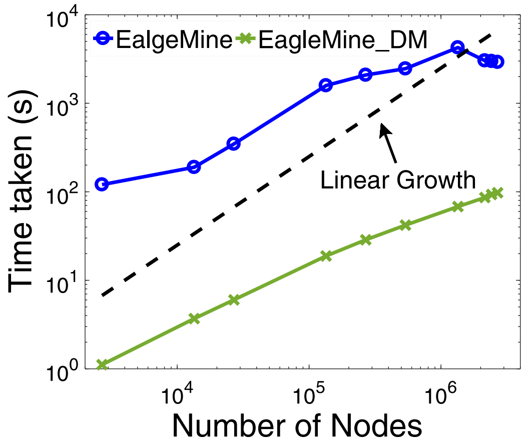


Fig. 10. EagleMine is scalable. EagleMine and EagleMine_DM scale (sub-) linearly with the number of nodes in graph (or the total value in histogram).

(consistent with human vision recognition) and quantitative comparison.

- **Anomaly detection:** EagleMine can detect some explainable anomalies on the real large graphs and achieves better accuracy for finding suspicious users and messages in Sina Weibo data compared with other baselines. EagleMine also detects micro-clusters for spotting suspicious objects in other applications beyond the graph, like the temporal synchronized behavior in the event stream.
- **Scalability:** EagleMine algorithm runs near linear in the number of graph nodes (or # of records), it can also deal with the multi-dimensional correlated features.

To obtain better vision judgment and summarization for the multi-dimensional histogram is still a challenging problem, our EagleMine_DM (with the multivariate Gaussian distribution as vocabulary terms) provides a promising choice for its comparable good performance and linear running time, and many theoretical well-defined hypothesis test approaches that can be used to determine the optimal clusters in the histogram.

However, we believe that there are still many directions for the possible extension of this work. Among others, an interesting problem is how to design data-driven approaches for constructing histogram to consider the distribution properties of different features, like the short-burst intuition of the temporal information for the fraudulent clusters; thus, this would greatly improve the algorithm's robustness and ability to distinguish between the honest and truly fraudulent clusters more accurately. An interesting theoretical direction would be exploring the properties of graph-related features and the tools or models to capture them, like the distribution of spectral features of graph nodes, as well as the high-dimensional statistical techniques for feature-based pattern mining to incorporate the powerful graph representation embedding.

CRediT authorship contribution statement

Wenjie Feng: Conceptualization, Data curation, Formal analysis, Methodology, Validation, Writing - original draft. **Shenghua Liu:** Conceptualization, Formal analysis, Methodology, Validation, Writing - original draft. **Christos Faloutsos:** Conceptualization, Methodology, Writing- review & editing. **Bryan Hooi:** Writing-review & editing. **Huawei Shen:** Writing- review & editing. **Xueqi Cheng:** Data curation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the Strategic Priority Research Program of CAS, China (XDA19020400), NSF of China (61772498, 61425016, 91746301, 61872206), and the Beijing NSF, China (4172059).

References

[1] D. Koutra, D. Jin, Y. Ning, C. Faloutsos, Perseus: an interactive large-scale graph mining and visualization tool, VLDB (2015).

[2] M. Jiang, P. Cui, A. Beutel, C. Faloutsos, S. Yang, Catchsync: catching synchronized behavior in large directed graphs, in: SIGKDD, 2014.

[3] L. Wilkinson, A. Anand, R. Grossman, Graph-theoretic scagnostics, in: Proceedings - IEEE Symposium on Information Visualization, INFO VIS, 2005, pp. 157–164.

[4] U. Kang, J.Y. Lee, D. Koutra, C. Faloutsos, Net-ray: Visualizing and mining billion-scale graphs, in: PAKDD, 2014.

[5] S. Fakhraei, J. Foulds, M. Shashanka, L. Getoor, Collective spammer detection in evolving multi-relational social networks, in: SIGKDD, in: KDD '15, ACM, 2015.

[6] V. Chandola, A. Banerjee, V. Kumar, Anomaly detection: A survey, ACM Comput. Surv. 41 (2009) 15:1–15:58.

[7] J.M. Kleinberg, Authoritative sources in a hyperlinked environment, J. ACM 46 (5) (1999) 604–632.

[8] U. Kang, B. Meeder, C. Faloutsos, Spectral analysis for billion-scale graphs: Discoveries and implementation, in: PAKDD, 2011.

[9] D. Pelleg, A.W. Moore, et al., X-means: Extending k-means with efficient estimation of the number of clusters., in: ICML, 2000, pp. 727–734.

[10] G. Hamerly, C. Elkan, Learning the k in k-means, NIPS (2004).

[11] T. Zhang, R. Ramakrishnan, M. Livny, Birch: An efficient data clustering method for very large databases, in: SIGMOD, 1996, pp. 103–114.

[12] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise., in: KDD, 1996.

[13] M. Ankerst, M.M. Breunig, H.-P. Kriegel, J. Sander, Optics: ordering points to identify the clustering structure, in: SIGMOD, 1999, pp. 49–60.

[14] C. Böhm, C. Faloutsos, J.-Y. Pan, C. Plant, Robust information-theoretic clustering, in: KDD, 2006.

[15] W. Wang, J. Yang, R. Muntz, et al., Sting: A statistical information grid approach to spatial data mining, in: VLDB, 1997, pp. 186–195.

[16] A. Rodriguez, A. Laio, Clustering by fast search and find of density peaks, Science 344 (6191) (2014) 1492–1496.

[17] J.B. Roerdink, A. Meijster, The watershed transform: Definitions, algorithms and parallelization strategies, Fundam. Inform. (2000).

[18] L. Vincent, P. Soille, Watersheds in digital spaces: an efficient algorithm based on immersion simulations, PAMI (1991) 583–598.

[19] R.J.G.B. Campello, D. Moulavi, A. Zimek, J. Sander, Hierarchical density estimates for data clustering, visualization, and outlier detection, ACM Trans. Knowl. Discov. Data 10 (1) (2015) 5:1–5:51.

[20] Y. Zhao, A survey on theoretical advances of community detection in networks, 2018, ArXiv, abs/1809.07691.

[21] F. Liu, S. Xue, J. Wu, C. Zhou, W. Hu, C. Paris, S. Nepal, J. Yang, P.S. Yu, Deep learning for community detection: Progress, challenges and opportunities, 2020, arXiv preprint arXiv:2005.08225.

[22] M. Heynckes, The predictive vs. the simulating brain: A literature review on the mechanisms behind mimicry, Maastricht Stud. J. Psychol. Neurosci. 4 (2016).

[23] M. Borkin, K. Gajos, A. Peters, D. Mitsouras, S. Melchionna, F. Rybicki, C. Feldman, H. Pfister, Evaluation of artery visualizations for heart disease diagnosis, IEEE Trans. Vis. Comput. Graph. (2011) 2479–2488.

[24] B. Perozzi, L. Akoglu, Discovering communities and anomalies in attributed graphs: Interactive visual exploration and summarization, ACM Trans. Knowl. Discov. Data 12 (2) (2018) http://dx.doi.org/10.1145/3139241.

[25] R.A. Rossi, N. Ahmed, R. Zhou, H. Eldardiry, Interactive visual graph mining and learning, ACM Trans. Intell. Syst. Technol. (TIST) 9 (2018) 1–25.

[26] A. Buja, P.A. Tukey, Computing and graphics in statistics, 1992.

[27] N. Gupta, D. Eswaran, N. Shah, L. Akoglu, C. Faloutsos, Beyond outlier detection: Lookout for pictorial explanation, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2018, pp. 122–138.

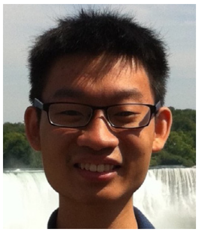
[28] N. Ahmed, R.A. Rossi, Interactive visual graph analytics on the web, in: ICWSM, 2015.

[29] L. Akoglu, H. Tong, D. Koutra, Graph based anomaly detection and description: a survey, Data mining and knowledge discovery (2015).
[30] Y. Weng, L. Liu, A collective anomaly detection approach for multidimensional streams in mobile service security, IEEE Access 7 (2019) 49157–49168.
[31] J. Yan, L. Shi, J. Tao, X. Yu, Z. Zhuang, C. Huang, R. Yu, P. Su, C. Wang, Y. Chen, Visual analysis of collective anomalies using faceted high-order correlation graphs, IEEE transactions on visualization and computer graphics (2018).
[32] V. Miz, B. Ricaud, K. Benzi, P. Vandergheynst, Anomaly detection in the dynamics of web and social networks using associative memory, in: The World Wide Web Conference, 2019, pp. 1290–1299.
[33] M. Ahmed, Collective anomaly detection techniques for network traffic analysis, Ann. Data Sci. 5 (4) (2018) 497–512.
[34] B.A. Prakash, A. Sridharan, M. Seshadri, S. Machiraju, C. Faloutsos, Eigenspokes: Surprising patterns and scalable community chipping in large graphs, in: PAKDD, 2010.
[35] M. Jiang, P. Cui, A. Beutel, C. Faloutsos, S. Yang, Inferring strange behavior from connectivity pattern in social networks, in: PAKDD, 2014.
[36] D. Koutra, U. Kang, J. Vreeken, C. Faloutsos, Vog: Summarizing and understanding large graphs, in: Proceedings of the 2014 SIAM International Conference on Data Mining, SIAM, 2014, pp. 91–99.
[37] N. Shah, D. Koutra, T. Zou, B. Gallagher, C. Faloutsos, Timecrunch: Interpretable dynamic graph summarization, in: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2015, pp. 1055–1064.
[38] M. Jiang, A. Beutel, P. Cui, B. Hooi, S. Yang, C. Faloutsos, A general suspiciousness metric for dense blocks in multimodal data, in: ICDM, 2015, pp. 781–786.
[39] K. Shin, B. Hooi, C. Faloutsos, M-zoom: Fast dense-block detection in tensors with quality guarantees, in: ECML-PKDD, 2016, pp. 264–280.
[40] B. Hooi, H.A. Song, A. Beutel, N. Shah, K. Shin, C. Faloutsos, Fraudar: Bounding graph fraud in the face of camouflage, in: SIGKDD, 2016, pp. 895–904.
[41] W. Feng, S. Liu, X. Cheng, Catchcore: Catching hierarchical dense subtensor, in: ECML/PKDD, 2019.
[42] D. Chakrabarti, C. Faloutsos, Graph mining: Laws, tools, and case studies, in: Graph Mining: Laws, Tools, and Case Studies, 2012.
[43] J. Wu, X. Zhu, C. Zhang, S.Y. Philip, Bag constrained structure pattern mining for multi-graph classification, IEEE Trans. Knowl. Data Eng. 26 (10) (2014) 2382–2396.
[44] J. Wu, S. Pan, X. Zhu, Z. Cai, Boosting for multi-graph classification, IEEE Trans. Cybern. 45 (3) (2014) 416–429.
[45] Q. Cao, M. Sirivianos, X. Yang, T. Pregueiro, Aiding the detection of fake accounts in large scale social online services, in: NSDI, 2012, pp. 197–210.
[46] S. Ghosh, B. Viswanath, F. Kooti, N.K. Sharma, G. Korlam, F. Benevenuto, N. Ganguly, K.P. Gummadi, Understanding and combating link farming in the twitter social network, in: WWW, 2012, pp. 61–70.
[47] Z. Zhang, P. Cui, W. Zhu, Deep learning on graphs: A survey, IEEE Trans. Knowl. Data Eng. (2020).
[48] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, S.Y. Philip, A comprehensive survey on graph neural networks, IEEE Trans. Neural Netw. Learn. Syst. (2020).
[49] Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, C. Zhang, Connecting the dots: Multivariate time series forecasting with graph neural networks, 2020, arXiv preprint arXiv:2005.11650.
[50] S. Wan, C. Gong, P. Zhong, S. Pan, G. Li, J. Yang, Hyperspectral image classification with context-aware dynamic graph convolutional network, IEEE Trans. Geosci. Remote Sens. (2020).
[51] S. Ji, S. Pan, E. Cambria, P. Marttinen, P.S. Yu, A survey on knowledge graphs: Representation, acquisition and applications, 2020, ArXiv abs/2002.00388.
[52] M.J. Zaki, J. Wagner Meira, Data Mining and Analysis: Fundamental Concepts and Algorithms, Cambridge University Press, 2014.
[53] A. Blum, J. Hopcroft, R. Kannan, Foundations of data science, Vorabversion eines Lehrbuchs (2016).
[54] J.J. DiCarlo, D. Zoccolan, N.C. Rust, How Does the Brain Solve Visual Object Recognition?.
[55] X.-M. Liu, R. Ji, C. Wang, W. Liu, B. Zhong, T.S. Huang, Understanding image structure via hierarchical shape parsing, in: CVPR, 2015.
[56] P. Arbelaez, M. Maire, C. Fowlkes, J. Malik, Contour detection and hierarchical image segmentation, PAMI (2011) 898–916.
[57] Z. Bylinskii, N.W. Kim, P. O'Donovan, S. Alsheikh, S. Madan, H. Pfister, F. Durand, B. Russell, A. Hertzmann, Learning visual importance for graphic designs and data visualizations, in: Proceedings of the 30th Annual ACM Symposium on User Interface Software & Technology, ACM, 2017, pp. 57–69.
[58] R.C. Gonzalez, R.E. Woods, Image processing, Digit. Image Process. (2007).
[59] H.R. Thompson, Truncated normal distributions, Nature 165 (1950) 444–445.
[60] C.M. Bishop, N.M. Nasrabadi, Pattern recognition and machine learning, J. Electron. Imaging 16 (2007).
[61] M. Cubedo, J.M. Oller, Hypothesis testing: a model selection approach, 2002.
[62] P.J. Huber, Projection pursuit, Ann. Statist. 13 (2) (1985) 435–475.
[63] A. Chernobai, S.T. Rachev, F.J. Fabozzi, Composite goodness-of-fit tests for left-truncated loss samples, in: C.-F. Lee, J.C. Lee (Eds.), Handbook of Financial Econometrics and Statistics, Springer New York, 2015, pp. 575–596.
[64] Amazon ratings network dataset – KONECT, 2017, URL http://konect.uni-koblenz.de/networks/amazon-ratings.
[65] J. McAuley, R. Pandey, J. Leskovec, Inferring networks of substitutable and complementary products, in: KDD, 2015.
[66] J.J. McAuley, J. Leskovec, From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews, in: WWW, 2013.
[67] Yelp dataset challenge, 2017, URL https://www.yelp.com/dataset_challenge.
[68] J. McAuley, J. Leskovec, Image labeling on a network: using social-network metadata for image classification, ECCV (2012) 828–841.
[69] A. Mislove, M. Marcon, K.P. Gummadi, P. Druschel, B. Bhattacharjee, Measurement and analysis of online social networks, in: SIGCOMM, 2007.
[70] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., Scikit-learn: Machine learning in python, J. Mach. Learn. Res. 12 (Oct) (2011) 2825–2830.
[71] L. McInnes, J. Healy, S. Astels, Hdbscan: Hierarchical density based clustering, J. Open Source Softw. 2 (11) (2017) 205.
[72] L. Wasserman, All of Nonparametric Statistics (Springer Texts in Statistics), Springer-Verlag New York, Inc., 2006.
[73] P. Elias, Universal codeword sets and representations of the integers, IEEE Trans. Inf. Theory (1975) 194–203.
[74] D. Chakrabarti, S. Papadimitriou, D.S. Modha, C. Faloutsos, Fully automatic cross-associations, in: SIGKDD, 2004, pp. 79–88.
[75] M. Giatsoglou, D. Chatzakou, N. Shah, A. Beutel, C. Faloutsos, A. Vakali, Nd-sync: Detecting synchronized fraud activities, in: Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer, 2015, pp. 201–214.

Wenjie Feng is a Ph.D. student in the Institute of Computing Technology, Chinese Academy of Sciences. He received B.S. in Computer Science from Beijing Jiao Tong University. His research interests include large scale graph mining, anomaly detection, and social network analysis.

Shenghua Liu is an associate Professor at the Institute of Computing Technology, Chinese Academy of Sciences. He received his Ph.D. degree from the Computer Science and Technology Department, Tsinghua University. He was a visiting scholar at the University of California, Los Angeles and Carnegie Mellon University respectively. His current research interests are designing intelligent and automated algorithms for big data mining problems, related to big graphs and series.

Christos Faloutsos is a Professor at Carnegie Mellon University. He has received the Presidential Young Investigator Award by the National Science Foundation (1989), the Research Contributions Award in ICDM 2006, the SIGKDD Innovations Award (2010), 28 "best paper" awards (including 6 "test of time" awards), and four teaching awards. Eight of his advisees have attracted KDD or SCS dissertation awards, He is an ACM Fellow, he has served as a member of the executive committee of SIGKDD. His research interests include large-scale data mining with emphasis on graphs and time sequences; anomaly detection, tensors, and fractals.



Bryan Hooi is an assistant Professor in the National University of Singapore. He received his Ph.D. in Machine Learning from Carnegie Mellon University. His main research interests include high-dimensional inference, network analysis, biomedical, and social science applications.

HuaWei Shen is a Professor in the Institute of Computing Technology, Chinese Academy of Sciences (CAS). He received his Ph.D. degree from the Institute of Computing Technology in 2010. His major research interests include network science, social media analytics and recommendation. He has published more than 80 papers in prestigious journals and top international conferences, including in Science, PNAS, Physical Review E, WWW, AAAI, IJCAI, SIGIR, CIKM, and WSDM. He is an Outstanding Member of the Association of Innovation Promotion for Youth of CAS.

Xueqi Cheng is a Professor at the Institute of Computing Technology, Chinese Academy of Sciences. His main research interests include network science, web search and data mining, big data processing and distributed computing architecture, etc. He has published more than 100 publications in prestigious journals and conferences, including the IEEE Transactions on Information Theory, IEEE Transactions on Knowledge and Data Engineering, Journal of Statistics Mechanics: Theory and Experiment, Physical Review E., ACM SIGIR, WWW, ACM CIKM, WSDM, IJCAI, ICDM, etc.