EagleMine: Vision-Guided Mining in Large Graphs

Wenjie Feng[1,2], Shenghua Liu[1], Christos Faloutsos[3], Bryan Hooi[3], Huawei Shen[1], Xueqi Cheng[1]
[1]CAS Key Laboratory of Network Data Science & Technology,
Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
[2]University of Chinese Academy of Sciences (UCAS), Beijing 100049, China
[3]School of Computer Science, Carnegie Mellon University, PA, USA
fengwenjie@software.ict.ac.cn,liu.shengh@gmail.com,christos@cs.cmu.edu
bhooi@andrew.cmu.edu,shenhuawei@ict.ac.cn,cxq@ict.ac.cn

ABSTRACT

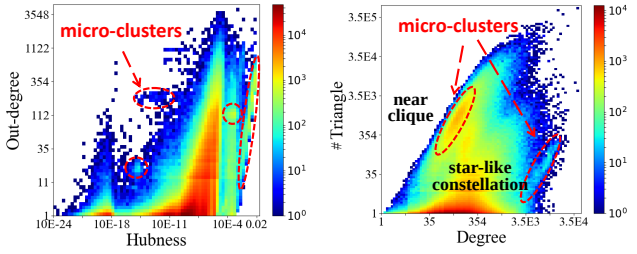
Given a graph with millions of nodes, what patterns exist in the distributions of node characteristics, and how can we detect them and separate anomalous nodes in a way similar to human vision?

In this paper, we propose a vision-guided algorithm, EagleMine, to recognize and summarize graph node groups in feature spaces. EagleMine hierarchically discovers node groups, and each group is an internally connected dense area in some feature space. EagleMine utilizes a water-level tree to capture group structures according to vision-based intuition at multiple resolutions. EagleMine traverses the water-level tree, applying statistical hypothesis test to determine the optimal node groups that should be fitted along the path. Moreover, EagleMine can identify anomalous micro-clusters (i.e., micro-size groups), who exhibit very similar behavior in some feature space, and deviate away from the majority. Experiments on real-world data show that our method can recognize intuitive groups as human vision does, and achieve the best performance in summarization compared to baselines. In terms of anomaly detection, EagleMine also outperforms well-known state-of-the-art graph-based methods by significantly improving accuracy in a microblog dataset.

1 INTRODUCTION

Given real-world graphs with millions of nodes and connections, the most intuitive way to explore the graphs is to construct a scatter-plot of graph nodes in coordinates of correlated features, known as correlation plot [29, 47]. Usually a heat map of those scatter-plot points is used to show their density, which is a two-dimensional histogram [24]. In this histogram, people can visually recognize nodes gathering into disjointed dense areas separately as groups (Fig. 1), which help to explore patterns (like communities, lockstep behaviors) and detect anomalies (e.g., fraudsters, attackers, fake-reviews, outlier etc.) in a more interpretable way [13, 26].

In particular, a graph can represent friendships in Facebook, ratings from users to items in Amazon, or retweets from users to messages in Twitter. A snapshot of such graphs can have numerous correlated features, e.g., degree, triangles, spectral vectors, and



(a) Microblog Sina Weibo data **(b) Tagged [17] data**

Figure 1: Heat maps (histograms) of correlation plots.

PageRank etc., which generate correlation plots by the combinations of them. It becomes, however, labor-intensive to manually monitor and recognize patterns from the snapshots of temporal graphs. Moreover, if we include more features, say, 4 features, visualizing and recognizing patterns becomes extremely difficult.

This raises the following questions: ***Given*** *a heat map (i.e., histogram) of the correlation plot of graph nodes in some feature space, how can we design an algorithm to automatically*

- ***recognize*** *and* ***monitor*** *the node groups as human vision does?*
- ***summarize*** *the graph nodes in the feature space and* ***identify*** *suspicious micro-clusters?*

'Micro-cluster' refers to relatively small group of people, that exhibit very similar behavior in the feature space. In the paper we demonstrate some of the possible feature spaces, namely

i out-degree vs hubness - Fig. 1a - this can spot nodes with high out-degree, but low hubness score (i.e., they have many outgoing edges to non-important nodes, probably, customers, that paid them) [28].

ii #triangles vs degree - Fig. 1b - spotting a near-clique group (too many triangles, for their degree), as well as star-like constellations (too few triangles for such high degree) [27].

In this paper, we propose EagleMine, a novel tree-based mining approach to recognize and summarize the node groups in a correlation plot of graph nodes. EagleMine can identify anomalous micro-clusters. Experiments show that EagleMine outperforms baselines and achieves better performance both in quantitative (i.e., code length for compact model description) and qualitative (i.e., consistent with vision-based judgment) comparisons. EagleMine detects a micro-cluster of hundreds of bots in a real-world microblog data, Sina Weibo[1], which presents strong signs of sharing unusual login-name prefixes, e.g., 'best*', 'black*' and '18-year-old*', and exhibiting very similar behavior in the feature space (see Fig. 2b).

[1]One of the largest microblog websites in China

ODD v5.0, KDD'18, August 2018, London, United Kingdom

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

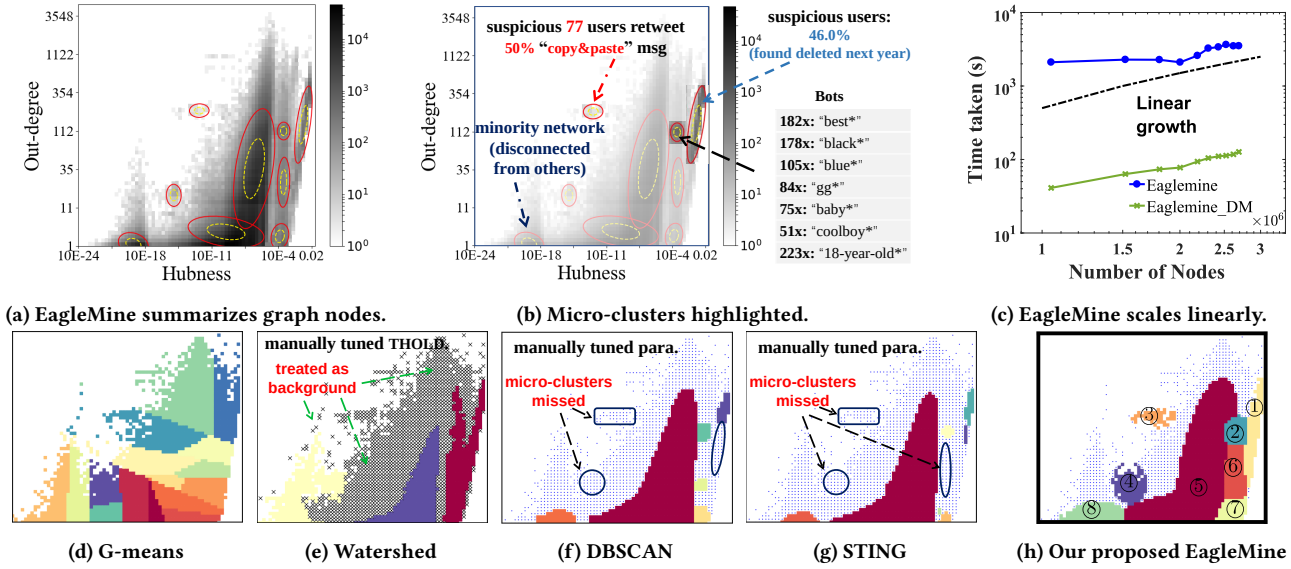


Figure 2: Our proposed EagleMine achieves effective results on microblog Sina Weibo data. (a) EagleMine summarizes the graph nodes in a feature space with truncated Gaussian distributions, where we show the 1.5·Σ and 3·Σ ellipses. Σ is the covariance matrix. The majority group (having the largest area and the most of nodes) in the middle is described by two overlapped and truncated Gaussians. (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 unusual prefix 'best'. (c). The *blue* curve shows the running time of EagleMine, compared to a linear function, and the *green* curve shows the generalization vocabulary (DM-Gaussian) of EagleMine. (d)-(h) are the results of recognizing node groups by the baselines and our EagleMine respectively. A relatively good results of Watershed is achieved by manually tuning the threshold for image background. The blue scattering points in (f)-(h) denote individual outlier nodes. DBSCAN and STING need extensive tuning of parameters. Still, some micro-clusters of low density are not recognized.

In summary, the proposed EagleMine has following advantages:

- **Automated summarization:** EagleMine automatically summarizes a histogram plot derived from correlated graph features (see Fig. 2a). EagleMine recognizes node groups forming disjointed dense areas as human vision does (see Fig. 2h).
- **Effectiveness:** EagleMine detects interpretable groups, and outperforms the baselines and even those with *manually tuned parameters* in qualitative (see Fig. 2d-2h), and quantitative experiments (see Fig. 6).
- **Anomaly detection:** EagleMine can spot, and even explain anomalies on real data by identifying suspicious micro-clusters (see Fig. 2b). Compared with the graph-based anomaly detection methods, EagleMine achieves higher accuracy for finding suspicious users in Sina Weibo data.
- **Scalability:** EagleMine is scalable, with nearly linear time complexity in the number of graph nodes, and can deal with more correlated features in multi-dimensional space.

Reproducibility: Our code is open-sourced at **https://github.com/wenchieh/eaglemine**, and most of the datasets we use are publicly available online.

In Section 2, we provide notions and our proposed model. In Section 3, we propose EagleMine algorithm for recognizing and summarizing graph nodes. In Section 4, we present experimental results. After discussing related work in Section 5, we offer conclusions in Section 6.

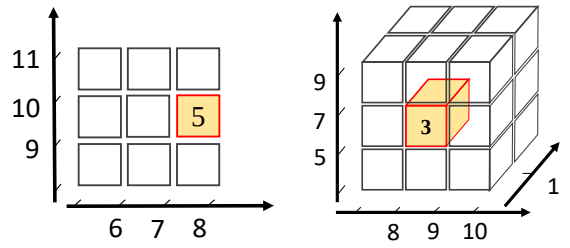
2 PROPOSED MODEL

Consider a graph $\mathcal{G} = (\boldsymbol{V}, \boldsymbol{E})$, where $\boldsymbol{V}$ is the node set and $\boldsymbol{E}$ is the edge set. $\mathcal{G}$ can be either homogeneous, such as friendship/following relations, or bipartite as users rating restaurants.

For the informal problem, as the introduction puts it, for graph $\mathcal{G}$ and correlated features of nodes $\mathbf{V}$, our goal is to optimize the goodness-of-fit (GoF) of node distribution, and the consistency with human visual recognition.

To estimate the density of scattering nodes in a feature space, we construct a histogram by mapping nodes into bins of bucketized features. The bandwidth (i.e., bin size) for each feature of the histogram can be selected according to the plug-in methods or kernel density-estimators [46]. such as degree and number of triangles, into bins with fixed bandwidth in logarithmic scale. Other continuous features, such as spectral features (hubness and authority), are evenly divided into similar sizes in logarithmic scale as well.

Notations for histogram: Generally, histograms can be viewed as multi-dimensional tensors that generalize vectors (1D histograms) and matrices (2D histograms). Consider a F-dimensional histogram $\mathcal{H}$ of size $I_1 \times \cdots \times I_F$, namely, having F correlated features. Each $(i_1, \cdots, i_F)$-th bin of $\mathcal{H}$ has non-negative value $h_{i_1,\cdots,i_F}$ as the number of nodes in that bin. i_n is the bin index of the n-th feature. Different from tensor, a histogram bin is an interval for one feature, a rectangle if $F = 2$, a cube if $F = 3$, and a hyper-cube if including more features. Hence the boundary for the $(i_1, \cdots, i_F)$-th bin of



(a) 2D histogram view (b) 3D histogram view

Figure 3: Pictorial depiction of the concepts of our histogram formalization. The colored are indicates one bin, in which the value is h.

histogram is an $F \times 2$ matrix $\boldsymbol{b}_{i_1,\cdots,i_F}$. Each row $\boldsymbol{b}_{i_n}$ is an interval of the i_n-th bin in n-th feature. Note that we may use h and $\boldsymbol{b}$ without any index for brevity, if no specific indices are needed in context. A pictorical view of the histogram is shown in Fig. 3.

To summarize the histogram of graph node feature more succinctly, we utilize some statistical distributions as vocabulary, being able to describe the density and randomness properties of nodes within its group, with some characteristic parameters. Therefore, our vocabulary-based model on graph nodes consists of

- **Configurable vocabulary:** distributions $\mathcal{Y}$ for describing node groups in a feature space $\mathcal{H}$.
- **Model parameters:** $\Theta = \{\theta_1, \cdots, \theta_C\}$ for vocabulary distribution term of C node groups (clusters).
- **Assignment variables:** $\mathcal{S} = \{s_1, \cdots, s_C\}$ for the assignment of distribution to each node group.
- **Outliers:** unassigned bins (indices) O for outlier nodes.

In terms of the configurable vocabulary $\mathcal{Y}$, it can include all suitable distributions, such as Uniform, Gaussian, Laplace, and exponential distributions, that depends on actual applications.

3 OUR PROPOSED METHOD

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

Trait 1. *Human vision usually detects connected components, which can be rapidly recognized by eyes despite substantial appearance variation[14, 32].*

This motivates us to identify each node group as an internally connected dense area in the heatmap, and different groups are disjointed from each other, which guides the refinement for smoothing.

Trait 2. *Top-to-bottom recognition and hierarchical segmentation[5]. 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.

Algorithm 1 describes the overall structure of EagleMine. It hierarchically detects micro-clusters in the histogram $\mathcal{H}$, and outputs the optimal summarization including the model parameters Θ, and assignment S for each node group, and outliers indices O. The following subsections will elaborate each step in detail.

Algorithm 1 EagleMine Algorithm

Input: Histogram $\mathcal{H}$ for node features of graph $\mathcal{G}$.
Output: summarization $\{\mathcal{S}, \Theta, O\}$.

1: Build a hierarchical tree structure $\mathcal{T}$ for $\mathcal{H}$. ▷ see section 3.1.
2: Describe node of $\mathcal{T}$ with the vocabulary. ▷ see section 3.2.
3: Explore the tree $\mathcal{T}$ and use hypothesis test as metric to determine the best node groups, which are summarized by the model parameters Θ and the assignment $\mathcal{S}$, as well as the outliers O. ▷ see section 3.3.
4: **return** summarization $\{\mathcal{S}, \Theta, O\}$.

3.1 Water-level tree

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

To organize all the islands in different water levels, we propose a water-level tree structure ($\mathcal{T}$), in which each node represents an island and each edge represents the relationship: where 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 moving from root to leaves.

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

The WaterLevelTree algorithm is shown in Alg. 2. We start from the root, and raise water level r in logarithmic scale from 0 to $\log h_{max}$ with step ρ, to account for a the power-law-like distribution of h. Let $h_{max} = \max \mathcal{H}$. We use the binary opening[2] operator ($\circ$) [18] for smoothing each internally jointed island, which is able to remove small isolated bins (treated as noise), and also separate weakly-connected islands with a specific structure element. Afterwards, we link each island at current level r_{curr} to its parent at lower water level r_{prev} of the tree (see Fig. 4a). Finally, when r reaches $\log h_{max}$, the flooding process stops, and we build a water-level tree.

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

Contract: The current tree $\mathcal{T}$ may contain many nodes with only one child, meaning no new islands separated, which are redundant. Hence we *search the tree using depth-first search (DFS); once a single-child node is found, we remove it and link its children to its parent.* This process is shown in Fig. 4a, where the dashed lines with arrow depict that the single-child's children are linked to its parent, and the gray links are removed.

Prune: The purpose of pruning is to smooth an island which has noise peaks on top of the island due to fluctuations of h in neighboring bins. An example is shown at bottom right of Fig. 4b. The island α at water-level r, contains fluctuation noises on the top.

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

Algorithm 2 WaterLevelTree Algorithm

Input: Histogram $\mathcal{H}$.
Output: Water-level tree $\mathcal{T}$.

```
T = {positive bins in H as root}.
for r = 0 to log h_max by step ρ do
    H^r = assign h ∈ H to zero if log h < r.
    H^r = H^r ∘ E.                    ▷ binary opening to smooth.
    islands A^r = {jointed bin areas in H^r}.
    link each island in A^r to its parent in T.
end for
Contract T: iteratively remove each single-child island and
    link its children to its parent.
Prune T: heuristically remove noise nodes.
Expand islands in T.
return T
```

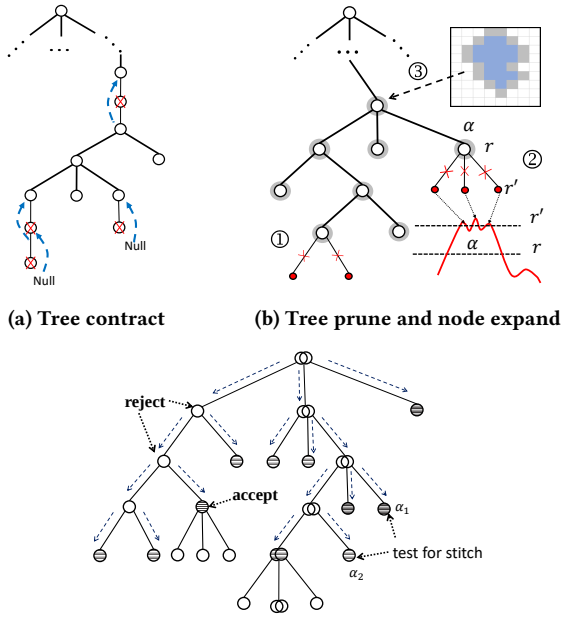
So when water level raises to r', the noises become three separated 'tiny' islands, linking to their parent α. Hence we *prune such child branches (including children's descendants) based on their total area size: the ratio of sum of h in child bins over sum of h in parent bin, is no more than* 95%. The pruning branches are illustrated in Fig.4b (① and ②).

Expand: We include additional surrounding bins to avoid overfitting for learning distribution parameters and to eliminate the possible effect of uniform step ρ for logarithmic scale. Hence we *iteratively augment towards positive bins around each island by a step of one-bin size, until islands touch each other, or doubling the number of bins in original island.* The expansion of node is illustrated with shadowed rings in Fig.4b. For the node ③, the light-blue labeled irregular part represents the original island area and the outer-peripheral gray bins are 1-st step expanded part as the pictorial depiction shows, and the further expansion follows similar process until it gets to the above constraints.

Note that in the Watershed formalization [43], the foreground of $\mathcal{H}$ are defined as catchment basins for clustering purpose, and can capture the boundaries between clusters as segmentation. We will see in experiments (section 4.2), the segmentation in Watershed approximates the islands in one level of tree $\mathcal{T}$, with a threshold parameter for background. STING also selects clusters in the same level, and needs a density threshold. However, EagleMine has no tuning parameters, and searches water-level tree to find a better combination of islands with hypothesis test, which may from different levels (see section 3.3).

3.2 Describing islands

The vocabulary $\mathcal{Y}$ can contain any proper user-defined distributions. In the feature space that we may concern, we use multivariate Gaussian as one of the vocabularies. Note that many node features, such as degrees and number of triangles, typically follow a power-law distribution. Hence some bins along histogram boundary may have larger number of nodes (see the bottom bins in Fig. 1). Thus truncated Gaussian distribution is a proper choice to describe such a truncated ellipse as far as we known. Moreover, since bins in $\mathcal{H}$ are discrete units, distributions must be discretized, namely, defining a probability function in each bin instead of probability density



(a) Tree contract **(b) Tree prune and node expand**

(c) Statistical hypothesis test and optimal islands search

Figure 4: Key steps in proposed algorithms.

function. Now the *discretized, truncated, multivariate* Gaussian distribution (DTM Gaussian for short) is defined as follows:

Definition 1 (DTM Gaussian). *The probability function in a bin with boundary matrix* $\boldsymbol{b}$ *is*

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

where $\boldsymbol{x}$ *is F-dimensional variable,* $\boldsymbol{b},\boldsymbol{\beta} \in \mathbb{R}^{F\times 2}$, $\boldsymbol{\beta}$ *is the truncation bound, and* $\psi(\cdot)$ *is density function of truncated normal distribution with mean* $\boldsymbol{\mu}$ *and co-variance* $\boldsymbol{\Sigma}$.

For 2D histogram, $\boldsymbol{\beta} = [[0,+\infty];[0,+\infty]]$. With DTM Gaussian, we can describe islands with shapes of lines, circles, and ellipses, and their truncation. The red-yellow oval circles in Fig. 2a depict some node clusters of DTM Gaussian described.

Observing the multi-mode distribution of islands (skewed triangle-like island in Fig. 1a), we also add mixture DTM Gaussian as another vocabulary. 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, users with similar degree shares similar hubness, forming a nearly normal distribution in horizontal. Therefore, those majority users forms a triangle-like island in the feature space, and we use mixture DTM Gaussian for it.

To decide the assignment $\mathcal{S} = \{s_1, \cdots, s_C\}$ of vocabulary to each island, we can use distribution-free statistical hypothesis test, like Pearson's χ^2 test, or other distribution specified approaches.

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

Algorithm 3 TreeExplore Algorithm

Input: WaterLevelTree $\mathcal{T}$
Output: summarization $\{\mathcal{S}, \Theta, O\}$.
1: $\Theta = \emptyset$.
2: $\mathcal{S}$ = decide the distribution type s_α from vocabulary for each island in $\mathcal{T}$.
3: queue $\boldsymbol{Q}$ = root node of $\mathcal{T}$.
4: **while** $\boldsymbol{Q} \neq \emptyset$ **do** ▷ breath-first search (BFS).
5: $\alpha \leftarrow$ dequeue of $\boldsymbol{Q}$.
6: $\theta_\alpha = DistributionFit(\alpha, s_\alpha)$
7: Hypothesis test $\boldsymbol{H}_0$ = bins of island α come from the distribution s_α.
8: **if** $\boldsymbol{H}_0$ is rejected **then**
9: enqueue all the children of α 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 promising distributions in $\mathcal{S}$, then update Θ.
16: Decide outliers O deviating from the recognized groups.
17: **return** summarization $\{\mathcal{S}, \Theta, O\}$.

3.3 Tree explore Algorithm

With the hierarchical water-level tree and describing vocabulary, we can then determine the optimal node groups and their summarization. The procedure is described in Alg. 3, where we explore the tree with BFS, decide the distribution vocabulary s_α for each tree node (island) α, select the optimal islands by using hypothesis test as a metric, and refine the final results with stitching in final.

In general, one can decide the assignment of distribution by Pearson's χ^2. However, we heuristically assign mixture DTM Gaussian to the island with the largest number of nodes in each tree level, and DTM Gaussian to other islands for simplicity.

BFS explore and hypothesis test: Afterwards, we search along the tree $\mathcal{T}$ with BFS to select the optimal combination of clusters (see step 4 to 14). Starting from the root node, we use the following null hypothesis to decide whether to explore the children:

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

Due to the variability of the value of bins in each island, we have tried Pearson's χ^2 test, BIC and AIC criteria but the extreme heights made the test and other criteria unstable. Thus, we test an island based on its binary image, which focuses on whether the island's shape looks like a truncated Gaussian or mixture. With projection pursuit [23], we simplify the hypothesis test by projecting the bins data to one dimension where the test can be apply. The author of [19] also use this approach to perform standard Gaussian test for determining the optimal k for k-means. And we accept the null hypothesis only when $\boldsymbol{H}_0$ is true for all axes projections. We then use Quadratic class 'upper tail' Anderson-Darling Statistic test[3] [12, 41] (with 1% significant level) to test on all axes. If one of the tests is rejected, the null hypothesis will be rejected.

If $\boldsymbol{H}_0$ is not rejected, we stop exploring the island's children. Otherwise, we further explore the children of island α. The dashed lines with an arrow in Fig. 4c demonstrate this process. Finally, the final optimal combination of islands to summarize the histogram is achieved until the BFS search stops.

[3] It is useful to measure the GoF of the left-truncated Gaussian distribution.

Stitch: Furthermore, some micro-clusters (islands) from different parents, e.g., α_1 and α_2 in Fig. 4c, are physically close to each other. In such case, those islands, separated due to sparsity and non-smoothness of data distribution or be over-split by WaterLevelTree, can potentially be summarized by one distribution. Therefore, we use *stitch* process in step 15 to merge them by hypothesis test. We test and stitch every pair of such islands from different parents in $\mathcal{T}$, until no changes occur. When there are multiple pairs of islands that be merged at the same time, we heuristically choose the pair with the least average log-likelihood reduction after stitching:

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

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: We assign the bins that are close to a distribution (i.e., have probability at least 10^{-4}) of a island to the group. Other bins are outliers O.

Furthermore, since the only one majority island fitted by mixture distribution contains the majority and normal nodes, we give the suspiciousness of other islands by the weighted KL-divergence as,

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

$$\kappa(\theta_i) = \log \bar{d}_i \cdot \sum_{\boldsymbol{b}} N_i \cdot KL(P_{\theta_i}(\boldsymbol{b}) \,||\, P_{\theta_m}(\boldsymbol{b})),$$

where $P_\theta(\boldsymbol{b})$ is the probability in the bin $\boldsymbol{b}$ for the distribution with θ as parameter, N_i is the number of samples in the island i, and we use the logarithm of $\bar{d}_i$, average degree of all graph nodes in the island i, as the weight based on the domain knowledge that if other features are the same, higher-degree nodes are more suspicious.

3.4 Time complexity

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

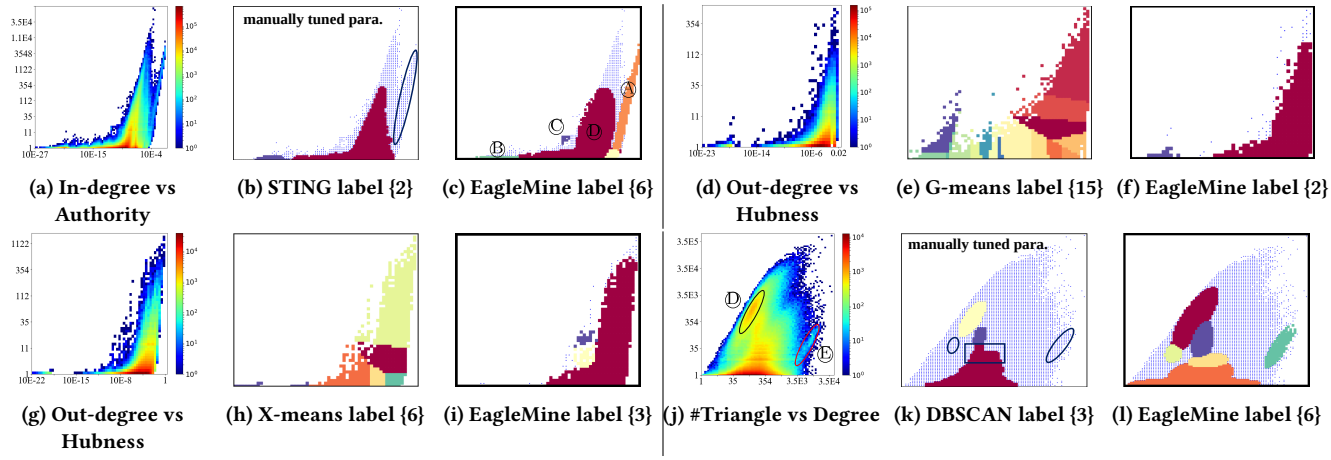
Let M be the number of non-empty bins in the histogram $\mathcal{H}$, and C be the number of clusters. We use gradient-descent to learn parameters in $DistributionFit(\cdot)$ of EagleMine algorithm, so we assume that the number of iterations is T, which is related to the differences between initial and optimal objective values, and $h_{max} = \max \mathcal{H}$ as defined before. Then we have:

Theorem 3.1. *The time complexity of EagleMine algorithm is* $O(\frac{\log h_{max}}{\rho} \cdot M + C \cdot T \cdot M)$.

4 EXPERIMENTS

We design the experiments to answer the following questions:

1. **Quantitative evaluation on real data:** Does EagleMine give significant improvement in concisely summarizing the graph?
2. **Qualitative evaluation (vision-based):** Does EagleMine accurately identify micro-clusters that agree with human vision?



(a) In-degree vs Authority (b) STING label {2} (c) EagleMine label {6} (d) Out-degree vs Hubness (e) G-means label {15} (f) EagleMine label {2}

(g) Out-degree vs Hubness (h) X-means label {6} (i) EagleMine label {3} (j) #Triangle vs Degree (k) DBSCAN label {3} (l) EagleMine label {6}

Figure 5: EagleMine visually recognizes better node groups than baselines in qualitative comparison. '{·}' gives the number of node groups recognized by each algorithm. (a)-(c): is in-degree and authority plot of Sina weibo data, which are message nodes corresponding to user nodes in Figure 2. (d)-(f): uses Android apps' rating data. (g)-(i): uses products online review in Yelp. (j)-(l): uses homogeneous graph from Tagged website.

3. **Anomaly detection:** How does EagleMine's performance on anomaly detection compare with the state-of-art methods? How much improvement does the visual-inspired info bring?
4. **Scalability:** Is EagleMine scalable with regard to the data size?

The data information used in our experiments is illustrated in Table 1. The Tagged [17] dataset was collected from Tagged.com social network website. It contains 7 anonymized types of links between users, and here we only choose the links of type-6, which is a homogeneous graph. The microblog Sina Weibo dataset was crawled in November 2013 from weibo.com, consisting of user-retweeting-message (bipartite) graph.

4.1 Q1. Quantitative Evaluation on Real Data

The comparison algorithms and their settings are listed as follows.

- 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 p-value= 0.01 which is insensitive.
- DBSCAN: set Eps=1, and use $\|h_i - h_j\|_\infty$ as distance function; we searched MinPts from the average number of nodes in a histogram bin until the max number by step 50, and manually select the one consistent well with human vision[4].
- STING: $c \approx \frac{\text{Minpts}+1}{\pi \text{Eps}^2}$ with DBSCAN's tuned optimal MinPts and Eps for clusters as initial, and refine the visual result by fine-tuning.
- EagleMine and EagleMine (DM): are our proposed EagleMine with DTM Gaussian and whole multivariate Gaussian respectively.

Envisioning the problem of clustering as a compression problem, we use Minimum Description Length (MDL) as the metric to measure the summarization as [7, 21] do. In short, it follows the assumption that the more we can compress the data, the more we can learn about its underlying patterns. The best model has the smallest MDL length:

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

This description of model consists of the following terms:

- The number of clusters requires $\log^*(C)$ bits. [5].
- The assignment $\mathcal{S}$ of all non-outlier bins in $\mathcal{H}$ requires $L_{\mathcal{S}} = |C| \cdot \log(\mathcal{Y})$ bits.
- The fitting parameters of each DTM Gaussian need $(F + \frac{(1+F)F}{2})$ free parameters and one fitted values $\tilde{N}$.So its encoding requires $l_0 \cdot (0.5F^2 + 1.5F + 1)$ bits, where l_0 is the floating point cost, We used 4×8 bits in our setting. So, the total parameter code-length is $L_{\Theta} = |C| \cdot l_0 \cdot (0.5F^2 + 1.5F + 1)$
- The outliers $\mathcal{O}$ require $L_{\mathcal{O}}$ bits to encode their indices.
- The model error requires L_ϵ bits. For a bin $\boldsymbol{b}$ in group (island) α_i, the expected number of nodes is $\tilde{h} = \left\lfloor 2^{\tilde{N}_i \cdot P_{s_i}(\boldsymbol{b})} \right\rfloor$, then the original count can be accurately recovered as $h = \tilde{h} + \epsilon$, Thus we encode the total description error as $L_\epsilon = \sum_{\boldsymbol{b}} (\log^*(h - \tilde{h}) + 1)$, where 1 denotes the sign bit.

As for the other methods in comparison, we calculated the MDL length with the same principle [7, 10, 15]. And for the different feature space, we chose the degree vs pagerank and degree vs triangle for Tagged dataset, and choose in-degree vs authority and out-degree vs hubness for the rest.

The comparison of MDL length is reported in Fig. 6a. We can see that EagleMine achieves the shortest description length, indicating a concise and good summarization. On average, EagleMine reduces the MDL code length more than 81.6%, 79.0%, 65.5%, 20.2% compared with STING, DBSCAN, X-means and G-means respectively. We also computed the MDL for Watershed clustering method on all dataset, it is even much larger than DBSCAN, so we do not show them in Fig. 6 for a better chart view. EagleMine also outperforms EagleMine (DM) over 4.6%, benefiting from a proper vocabulary selection. Therefore, EagleMine summarizes the histogram with recognized groups in the best description length.

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

[5]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. [15]

Table 1: Dataset statistics summary

	# of nodes	# of edges	Content
Amazon rating [1]	(2.14M, 1.23M)	5.84M	Rate
Android [33]	(1.32M, 61.27K)	2.64M	Rate
BeerAdvocate [34]	(33.37K, 65.91K)	1.57M	Rate
Yelp [2]	(686K, 85.54K)	2.68M	Rate
Tagged [17]	(2.73M, 4.65M)	150.8M	Anonymized Links
Youtube [35]	(3.22M, 3.22M)	9.37M	Who-follow-who
Sina weibo	(2.75M, 8.08M)	50.1M	User-retweet-msg

Due to the complexity of geometric properties of distributions in high-dimensional space [6, 48], we conducted EagleMine in 3-, 4- and 5-dimensional features spaces on Amazon and Yelp datasets, and use the out-degree vs. top-k hubnesss (top-k left singular vectors) as features. Here, EagleMine used only digitized multivariate Gaussian as the vocabulary for simplicity. We can roughly draw the conclusion that more feature dimensions will lead to larger cost to describe for the histogram.

4.2 Q2. Qualitative evaluation (vision-based)

In this part, we illustrate the results on 2D histogram for vision-based qualitative comparison. Due to the space limit, we try to exhibit the comparisons with different baselines and in different feature spaces in an alternative way. The baselines include X-means, G-means, DBSCAN, STING, and Watershed. The feature spaces include out-degree vs hubness, in-degree vs authority, and #triangle vs degree.

Fig. 2d-2h shows the results on user-retweet-message graph in Sina Weibo data. The plot features are user's out-degree and hubness indicating how many important messages retweeted. Without removing some low-density bins as background, Watershed algorithm easily identified all the groups into one or two huge ones. Hence we manually tuned the threshold of background to attain a better result, which is similar to the groups in a level of our water-level tree. The background for Watershed is shown with gray color in Fig. 2e. As we can see, Watershed 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 groups in a more intuitive way, and identify those micro-clusters missed by DBSCAN and STING. Note that the user deletion ratio in the missed micro-clusters ① and ③ is unusually high, they were suspended by the system operators for anti-spam. Besides, those micro-clusters ③ and ④ include the users that have high out-degree but low-hubness, i.e., users retweeting many non-important messages (e.g., advertisements). Therefore, EagleMine identify very useful micro-clusters automatically as human vision does.

Moreover, Fig. 5 illustrates more examples on different datasets and feature spaces in four groups. The original histogram plots are given at the beginning of each group following with label figures of different methods. Outlier nodes are labeled with the *blue* color. Different colors represent different groups by corresponding methods. From original plots, people will naturally expect that bins with the similar color (density) and jointed locations should be in one group. Hence we can see that G-means and X-means produce a number of groups, over-separating the groups recognized by human vision. Although manually tuned DBSCAN and STING can capture the majority dense region in each plot, while overlooking some suspicious micro-clusters, e.g., micro-clusters Ⓐ and Ⓒ in Fig. 5c. Thus, EagleMine illustrates its advantages of recognizing groups, especially identifying micro-clusters, which is more consistent with human vision.

Table 2: Suspicious ranking of micro-clusters in Sina Weibo.

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

4.3 Q3. Anomaly detection

To compare the performance for anomaly detection, we labeled these nodes, both user and message, from the results of baselines, and sample nodes of our suspicious clusters from EagleMine, considering that it is impossible to label all the nodes. Our labels were based on the following rules [22]:

- user-accounts/messages which are deleted from the online system (system operators found the spams) [6].
- a lot of users that share unusual login-names prefixes, and other suspicious signals: approximately the same sign-up time, friends and followers count.
- messages on advertisement or retweeting promotion, and having lots of *copy-and-paste* text content.

In total, we labeled 5,474 suspicious users and 4,890 suspicious messages. We compared with state-of-the-art fraud detection algorithms GetScoop[25], SpokEn [38], and Fraudar [22].

EagleMine returns the micro-clusters with suspiciousness scores. Table 2 lists the ranking orders of micro-clusters identified by EagleMine for Fig. 2h and 5c respectively, according to their scores.

The anomaly detection results are reported in Fig. 6b, 6c. Using the AUC (area under the ROC curve) to quantify the quality of the ordered result from the algorithm, the sampled nodes from micro-clusters are ranked in descendant order of hubness or authority. The results show that EagleMine achieves more than 10% improvement in anomalous user detection, and about 50% improvement in anomalous message detection, outperforms the baselines. The anomalous users detected by Fraudar and SpokEn only fall in the micro-cluster ① in Fig. 2h, since their algorithms can only focus on densest core in a graph. But EagleMine detects suspicious users by recognizing noteworthy micro-clusters in the whole feature space. Simply put, EagleMine detects more anomalies than the baselines, by identifying extra micro-clusters ②, ③, and ④ in the feature space.

4.4 Q3. Pattern case study

As discussed above, the micro-clusters ③ and ④ in out-degree vs hubness Fig. 2h contains those users frequently rewtweet non-important messages. Here we study the behavior patterns of micro-clusters ① and ② on the right side of the majority group. Note that almost half of the users are deleted by system operators, and many existing users share unusual name prefixes as Fig. 2b shown.

What pattern have we found? The Fig. 6d shows the 'Jellyfish' structure of the subgraph consisting of users from micro-clusters

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

① and ②. The head of 'Jellyfish' is the densest core (①), where the users created unusual dense connections to a group of messages, showing high hubness. The users (spammers or bots) aggressively retweeted the similar message collection (i.e., Ⓐ in Fig. 5c). At the meantime, users from ② connected to some of 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 shows lower hubness than those in ①, since of the different spamming strategies, which are overlooked by density-based detection methods.

4.5 Q4. Scalability

Fig. 2c shows the near-linear scaling of EagleMine's running time in the numbers graph nodes. Here we used Sina Weibo dataset, we selected the snapshot of the graph, i.e., the reduced subgraph, according to the timestamp of edge creation in top 3, 6, . . . , 30 days. Slope of *black dot* line indicates linear growth.

5 RELATED WORK

For the Gaussian clusters, K-means, X-means [37], G-means [19], and BIRCH [49] (which is suitable for spherical clusters) algorithms suffer from being sensitive to outliers. Those methods are distance based, which prefer to cluster the points with a short distance in feature space forming a spherical area. Density based methods, such as DBSCAN [16] and OPTICS [4] are noise-resistant and can detect clusters of arbitrary shape and data distribution, while the clustering performance relies on density threshold for DBSCAN, and also for OPTICS to derive clusters from reachability-plot. RIC [7] enhances other clustering algorithms as a framework, using minimum description language as goodness criterion to select fitting distributions and separate noise. STING [44] hierarchically merges grids in lower layers to find clusters with a given density threshold. Clustering algorithms [39] derived from the watershed transformation [43], treat pixel region between watersheds as one cluster, and only focus on the final results and ignores the hierarchical structure of clusters. Community detection algorithms [31], modularity-driven clustering, and cut-based methods [40] usually cannot handle large graphs with million nodes or fail to provide intuitive and interpretable result when applying to graph clustering.

Supported by human vision theory, including visual saliency, color sensitive, depth perception and attention of vision system [20], visualization techniques [8, 45] and HCI tools help to get insight into data [3, 42]. Scagnostic [9, 42] diagnoses the anomalies from the plots of scattered points. [47] improves the detection by statistical features derived from graph-theoretic measures. Net-Ray [26] visualizes and mines adjacency matrices and scatter plots of a large graph, and discovers some interesting patterns.

In terms of anomaly detection in graphs, [25, 38] find communities and suspicious clusters in graph with spectral-subspace plots. SpokEn [38] considers the "eigenspokes" pattern on EE-plot produced by pairs of eigenvectors of graphs, and is later generalized for fraud detection. As more recent works, dense block detection (DBD) has been proposed to identify anomalous patterns and suspicious behaviors [11, 22, 30]. A representative work, Fraudar [22], proposed a densest subgraph-detection method that incorporates the suspiciousness of nodes and edges during optimization.

Table 3: Comparison between algorithms.

	X-means [37]	G-means [19]	DBSCAN [16]	BIRCH [49]	STING [44]	Watershed [39, 43]	DBD [22, 36]	**EagleMine**
parameter free	✓	✓				?	✓	✓
non-spherical cluster			✓		✓	✓	?	✓
anomaly detection			✓		✓		✓	✓
summarization				✓	✓			✓
linear in #nodes				✓	✓	✓		✓

A comparison between EagleMine and the majority of the above methods is summarized in Table 3. Our proposed method EagleMine is the only one that matches all specifications.

6 CONCLUSIONS

We propose a tree-based approach EagleMine to mine and summarize all node groups in a histogram plot of a large graph. The EagleMine algorithm finds optimal clusters based on water-level tree and statistical hypothesis test, describes them with a configurable model vocabulary, and detects some suspicious. EagleMine has desirable properties:

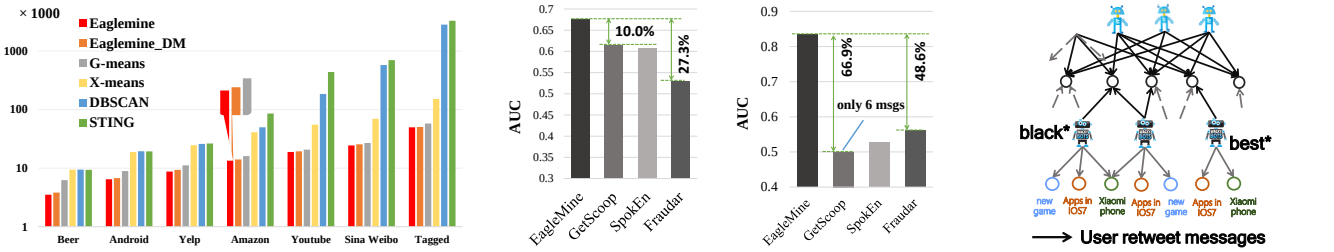
- **Automated summarization:** Our algorithm automatically summarizes a given histogram with vocabulary of distributions, inspired by human vision to find the graph node groups and outliers.
- **Effectiveness:** We compared EagleMine on real data with the baselines, the result shown that our detection is consistent with human vision, and also achieves better MDL in summarization.
- **Anomaly detection:** EagleMine can detect explainable anomalies on real data and achieve higher accuracy for finding suspicious users and bots micro-clusters.
- **Scalability:** EagleMine algorithm runs near linear in the number of graph nodes, and can handle the multi-dimensional correlated feature space.

To obtain better vision judgment and summarization for multi-dimensional features is still a challenge, our EagleMine-DM (with the multivariate Gaussian distribution as vocabularies) is a good choice in those cases for its comparable performance and better running times, and there are many theoretical well-defined hypothesis test approaches to be used to determine the optimal clusters.

ACKNOWLEDGMENTS

This work was partially funded by National Grand Fundamental Research 973 Program of China No. 2014CB340401, the Beijing NSF No. 4172059, the National NSF of China No. 61772498.

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 of China, or other funding parties.



(a) MDL Quantitative Evaluation (b) AUC for suspicious users (c) AUC for suspicious messages (d) Graphical view of anomalies.

Figure 6: EagleMine Alg. Performance. (a) MDL is compared on different real-world datasets. EagleMine achieves the shortest description code length, which means concise summarization, and outperforms all other baselines. (b), (c) EagleMine has the best AUC for detecting suspicious users and messages on Sina weibo. (d) The structure of anomalies identified in Sina weibo.

REFERENCES

[1] 2017. Amazon ratings network dataset – KONECT. (April 2017). http://konect.uni-koblenz.de/networks/amazon-ratings
[2] 2017. Yelp dataset challenge. (2017). https://www.yelp.com/dataset_challenge
[3] Leman Akoglu, Duen Horng Chau, U Kang, Danai Koutra, and Christos Faloutsos. 2012. Opavion: Mining and visualization in large graphs. In *SIGMOD*. 717–720.
[4] Mihael Ankerst, Markus M Breunig, Hans-Peter Kriegel, and Jörg Sander. 1999. OPTICS: ordering points to identify the clustering structure. In *SIGMOD*. 49–60.
[5] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. 2011. Contour detection and hierarchical image segmentation. *PAMI* (2011), 898–916.
[6] Avrim Blum, John Hopcroft, and Ravindran Kannan. 2016. Foundations of data science. *Vorabversion eines Lehrbuchs* (2016).
[7] Christian Böhm, Christos Faloutsos, Jia-Yu Pan, and Claudia Plant. 2006. Robust information-theoretic clustering. In *KDD*.
[8] Michelle Borkin, Krzysztof Gajos, Amanda Peters, Dimitrios Mitsouras, Simone Melchionna, Frank Rybicki, Charles Feldman, and Hanspeter Pfister. 2011. Evaluation of Artery Visualizations for Heart Disease Diagnosis. *IEEE Trans. on Visualization and Computer Graphics* (2011), 2479–2488.
[9] Andreas Buja and Paul A Tukey. 1992. *Computing and graphics in statistics*. Springer-Verlag New York, Inc.
[10] Deepayan Chakrabarti, Spiros Papadimitriou, Dharmendra S. Modha, and Christos Faloutsos. 2004. Fully Automatic Cross-associations. In *SIGKDD*. 79–88.
[11] Jie Chen and Yousef Saad. 2010. Dense Subgraph Extraction with Application to Community Detection. *IEEE Trans. on knowledge and Data Engineering* (2010).
[12] Anna Chernobai, Svetlozar T Rachev, and Frank J Fabozzi. 2015. Composite goodness-of-fit tests for left-truncated loss samples. In *Handbook of Financial Econometrics and Statistics*. Springer.
[13] Alceu Ferraz Costa, Yuto Yamaguchi, Agma Juci Machado Traina, Caetano Traina Jr., and Christos Faloutsos. 2014. RSC: Mining and Modeling Temporal Activity in Social Media. In *SIGKDD*.
[14] James J. DiCarlo, Davide Zoccolan, and Nicole C. Rust. 2012. How Does the Brain Solve Visual Object Recognition? *Neuron* (2012), 415 – 434.
[15] Peter Elias. 1975. Universal codeword sets and representations of the integers. *IEEE trans. on information theory* (1975), 194–203.
[16] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise.. In *KDD*.
[17] Shobeir Fakhraei, James Foulds, Madhusudana Shashanka, and Lise Getoor. 2015. Collective Spammer Detection in Evolving Multi-Relational Social Networks. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '15)*. ACM, 1769–1778.
[18] Rafael C Gonzalez and Richard E Woods. 2007. Image processing. *Digital image processing* (2007).
[19] Greg Hamerly and Charles Elkan. 2004. Learning the K in K-means. *NIPS* (2004).
[20] Miriam Heynckes. 2016. The predictive vs. the simulating brain: A literature review on the mechanisms behind mimicry. *Maastricht Student Journal of Psychology and Neuroscience* (2016).
[21] Bryan Hooi, Shenghua Liu, Asim Smailagic, and Christos Faloutsos. 2017. BeatLex: Summarizing and Forecasting Time Series with Patterns. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer.
[22] Bryan Hooi, Hyun Ah Song, Alex Beutel, Neil Shah, Kijung Shin, and Christos Faloutsos. 2016. Fraudar: Bounding graph fraud in the face of camouflage. In *SIGKDD*. 895–904.
[23] Peter J. Huber. 1985. Projection Pursuit. *Annals of Statistics* 13, 2 (1985), 435–475.
[24] Meng Jiang, Peng Cui, Alex Beutel, Christos Faloutsos, and Shiqiang Yang. 2014. CatchSync: catching synchronized behavior in large directed graphs. In *SIGKDD*.
[25] Meng Jiang, Peng Cui, Alex Beutel, Christos Faloutsos, and Shiqiang Yang. 2014. Inferring strange behavior from connectivity pattern in social networks. In *PAKDD*.
[26] U. Kang, Jay Yoon Lee, Danai Koutra, and Christos Faloutsos. 2014. Net-Ray: Visualizing and Mining Billion-Scale Graphs. In *PAKDD*.
[27] U Kang, Brendan Meeder, and Christos Faloutsos. 2011. Spectral analysis for billion-scale graphs: Discoveries and implementation. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 13–25.
[28] Jon M Kleinberg. 1999. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)* 46, 5 (1999), 604–632.
[29] Danai Koutra, Di Jin, Yuanchi Ning, and Christos Faloutsos. 2015. Perseus: an interactive large-scale graph mining and visualization tool. *VLDB* (2015).
[30] Ravi Kumar, Jasmine Novak, and Andrew Tomkins. 2010. Structure and evolution of online social networks. In *Link mining: models, algorithms, and applications*. Springer.
[31] Andrea Lancichinetti and Santo Fortunato. 2009. Community detection algorithms: a comparative analysis. *Physical review E* (2009), 056117.
[32] Xian-Ming Liu, Rongrong Ji, Changhu Wang, Wei Liu, Bineng Zhong, and Thomas S Huang. 2015. Understanding image structure via hierarchical shape parsing. In *CVPR*.
[33] Julian McAuley, Rahul Pandey, and Jure Leskovec. 2015. Inferring Networks of Substitutable and Complementary Products. In *KDD*.
[34] Julian John McAuley and Jure Leskovec. 2013. From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. In *WWW*. 897–908.
[35] Alan Mislove, Massimiliano Marcon, Krishna P. Gummadi, Peter Druschel, and Bobby Bhattacharjee. 2007. Measurement and Analysis of Online Social Networks. In *SIGCOMM*.
[36] Jian Pei, Daxin Jiang, and Aidong Zhang. 2005. On mining cross-graph quasi-cliques. In *SIGKDDD*. 228–238.
[37] Dan Pelleg, Andrew W Moore, et al. 2000. X-means: Extending K-means with Efficient Estimation of the Number of Clusters.. In *ICML*. 727–734.
[38] B Aditya Prakash, Ashwin Sridharan, Mukund Seshadri, Sridhar Machiraju, and Christos Faloutsos. 2010. Eigenspokes: Surprising patterns and scalable community chipping in large graphs. In *PAKDD*. 435–448.
[39] Jos BTM Roerdink and Arnold Meijster. 2000. The watershed transform: Definitions, algorithms and parallelization strategies. *Fundamenta informaticae* (2000).
[40] Satu Elisa Schaeffer. 2007. Graph clustering. *Computer science review* (2007).
[41] Michael A Stephens. 1974. EDF statistics for goodness of fit and some comparisons. *Journal of the American statistical Association* (1974), 730–737.
[42] J. W. Tukey and P. A. Tukey. 1985. Computer graphics and exploratory data analysis: An introduction. *Nat Computer Graphics Association* (1985).
[43] Luc Vincent and Pierre Soille. 1991. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *PAMI* (1991), 583–598.
[44] Wei Wang, Jiong Yang, Richard Muntz, et al. 1997. STING: A statistical information grid approach to spatial data mining. In *VLDB*. 186–195.
[45] C. Ware. 1988. Color sequences for univariate maps: theory, experiments and principles. *IEEE Computer Graphics and Applications* (Sept 1988), 41–49.
[46] Larry Wasserman. 2006. *All of Nonparametric Statistics (Springer Texts in Statistics)*. Springer-Verlag New York, Inc.
[47] Leland Wilkinson, Anushka Anand, and Robert Grossman. 2005. Graph-theoretic scagnostics. *Proceedings - IEEE Symposium on Information Visualization, INFO VIS* (2005), 157–164.
[48] Mohammed J. Zaki and Jr. Wagner Meira. 2014. *Data Mining and Analysis: Fundamental Concepts and Algorithms*. Cambridge University Press.
[49] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. 1996. BIRCH: An Efficient Data Clustering Method for Very Large Databases. In *SIGMOD*. 103–114.