A Probabilistic Approach to Uncovering Attributed Graph Anomalies

Nan Li* Huan Sun* Kyle Chipman† Jemin George‡ Xifeng Yan*

Abstract

Uncovering subgraphs with an abnormal distribution of attributes reveals much insight into network behaviors. For example in social or communication networks, diseases or intrusions usually do not propagate uniformly, which makes it critical to find anomalous regions with high concentrations of a specific disease or intrusion. In this paper, we introduce a probabilistic model to identify anomalous subgraphs containing a significantly different percentage of a certain vertex attribute, such as a specific disease or an intrusion, compared to the rest of the graph. Our framework, gAnomaly, models generative processes of vertex attributes and divides the graph into regions that are governed by background and anomaly processes. Two types of regularizers are employed to smoothen the regions and to facilitate vertex assignment. We utilize deterministic annealing EM to learn the model parameters, which is less initialization-dependent and better at avoiding local optima. In order to find fine-grained anomalies, an iterative procedure is further proposed. Experiments show gAnomaly outperforms a state-of-the-art algorithm at uncovering anomalous subgraphs in attributed graphs.

1 Introduction

The proliferation of rich information in real-world social and communication networks raises new challenges for graph mining. An important feature of these graphs is that vertices are often associated with attributes and events. For example, in a social network, a user could be annotated by the type of disease he/she has been infected with; in a communication network, a computer might record various attacks it receives. Information such as diseases or intrusions usually does not propagate evenly in these networks. Anomalous regions found with high concentrations of a specific disease or intrusion give rise to new interesting data mining problems. We might ask, why does a disease spread much faster in some portions of a network? Why do a subset of computers receive most of the attacks in the past day, and are they therefore targeted attacks?

*Computer Science Department, University of California, Santa Barbara. {nanli, huansun, xyan}@cs.ucsb.edu.

†Neural Science Research Institue, University of California, Santa Barbara. kchipman@gmail.com.

‡Army Research Lab. jemin.george.civ@mail.mil.

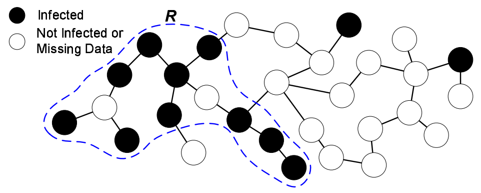


Figure 1: A Subgraph with Many Infected Vertices

Consider a human network in Figure 1, where an edge represents a friendship and a vertex color shows whether a person is infected with a certain disease. The region, $\mathcal{R}$, exhibits a very different distribution of vertex colors, as most people in $\mathcal{R}$ are infected, in contrast to outside of $\mathcal{R}$. It is interesting to identify such regions so that we can study what caused the aggressive spread of this disease within $\mathcal{R}$. Applications of such detection abound. For instance, in a customer network, we can uncover interesting customer chains, a great percentage of whom purchased a certain product.

Subgraphs like $\mathcal{R}$ in Figure 1 with a high concentration of an attribute are examples of anomalies. Various types of graph anomalies have been studied [1, 3, 6, 14]. [1] finds abnormal vertices by checking if their *ego-nets* comply with some power law-based rules. [14] uses a variant of the minimum description length (MDL) principle to uncover surprising substructures and subgraphs. Community outliers are studied in [6], which finds contextual abnormal vertices in information networks. These studies are not able to discover the kind of anomalies illustrated in Figure 1, where abnormal distribution of attributes in a graph is captured.

We propose a probabilistic framework, gAnomaly, that automatically captures and describes anomalies in a vertex-attributed graph by modeling various attribute distributions. An anomaly is a connected subgraph whose distribution of attributes significantly differs from the rest of the graph, as a result of non-random behaviors. Our motivation is to identify regions where information has spread abnormally so that further study of such anomalous propagation can be conducted.

Our work is related to some previous studies. Abnormal vertices with respect to specific communities are studied in [6], which proposes a unified framework for outlier and community discovery. However, the outliers are individual vertices scattered in the communities; no stress is laid on the connectivity among outliers. In contrast, gAnomaly finds connected substructures exhibiting aberrant distribution of attributes. [21] introduces a probabilistic approach to find graph clusters exhibiting high edge density and attributive homogeneity. gAnomaly not only finds such clusters, but also sparsely-connected regions as long as their attribute distribution deviates from the majority. We further propose an iterative procedure tailored towards fine-grained anomaly detection, which makes gAnomaly more robust and versatile. gAnomaly assumes that different graph regions are governed by different attribute distributions, and serves as a fundamental model to uncover such underlying distributions and the corresponding regions. gAnomaly helps understand vertex behaviors in the structural and attributive space, a key to subsequent social influence and information diffusion analysis.

Our contributions: (1) gAnomaly adopts an extended finite mixture model (FMM) [5] to interpret the underlying attribute distributions. (2) We propose multiple network regularizers to account for the graph structure, and an entropy regularizer to facilitate the vertex assignment into different mixture components. (3) An iterative procedure is proposed to find more fine-grained anomalies in challenging graphs. (4) Experiments on both synthetic and real network data demonstrate the effectiveness of our framework.

2 Problem Definition

gAnomaly identifies anomalies by modeling the underlying generative processes for vertex attributes. In a vertex-attributed graph, there might exist regions whose distribution of attributes significantly deviates from the majority. Using the previous example, if we consider the attribute "infected" which has two values, {"Yes", "No"}, there could exist regions with an abnormally higher percentage of infected people.

Let $G = (V, E, \mathcal{A})$ be an undirected vertex-attributed graph. V is the vertex set, E is the edge set, and $\mathcal{A}$ is a function that maps a vertex to an attribute value, $\mathcal{A} : V \to \mathbb{A}$, where $\mathbb{A}$ is the set of distinct attribute values in G. For the ease of presentation, we assume there is only one attribute, which has binary values. Without loss of generality, assume $\mathbb{A} = \{1(black), 0(white)\}$. gAnomaly is extensible to multiple attributes with categorical attribute values.

Problem Statement: Given a vertex-attributed graph G of black and white vertices, assuming the white vertices are the majority, our goal is to uncover anomalous subgraphs where a much higher percentage of black vertices occurs. The expected solution should balance the trade-off between two factors: (1) The size of the connected anomalous subgraphs; discovering small sets of black vertices that are scattered is not interesting. (2) The percentage of black vertices in the anomalies. The solution ought to accommodate different propagation models. For example, different diseases have different spreading patterns in a social network. In the epidemic of a contagious disease such as flu, the neighbors of an infected individual are likely infected too; while for a non-contagious genetic disease, two patients can be one or two hops away. In addition, the solution should be robust against noisy and missing data, which is normal in social and communication networks.

3 Data Model and Regularization

We first make assumptions about how vertex attributes are generated and create a model to describe them. Inspired by the anomaly detection model in [5], we employ a two-component *mixture model* to interpret the observed data. Anomaly detection is materialized essentially through assigning each vertex to one of the mixture components. Let $V^{(0)}$ be the set of *majority* (*background*) vertices, and $V^{(1)}$ the set of *anomaly* vertices. $V = V^{(0)} \bigcup V^{(1)}$, and $V^{(0)} \bigcap V^{(1)} = \emptyset$. Given a vertex v_i, with probability $\theta_i^{(k)}$, v_i belongs to class $V^{(k)}, k = \{0, 1\}$. Let P be a mixture model interpreting the overall distribution for a vertex, we have

$$P(v_i) = \sum_{k=0}^{1} \theta_i^{(k)} P^{(k)}(v_i), \tag{3.1}$$

where $\{\theta_i^{(0)}, \theta_i^{(1)}\}$ is the vertex-dependent mixture weights and $\theta_i^{(0)} + \theta_i^{(1)} = 1$. $P^{(0)}$ is the background model, and $P^{(1)}$ is the anomaly model. $P^{(k)}(v_i), k = \{0, 1\}$ is the conditional likelihood of observing v_i, given model $P^{(k)}$. Depending on the mixture weights $\{\theta_i^{(0)}, \theta_i^{(1)}\}$, each vertex is better explained by either the anomaly model, $P^{(1)}$, or the background model, $P^{(0)}$.

3.1 Bernoulli Mixture Model Since we assume there is only one attribute in G, each vertex either has attribute value 1 (containing this attribute) or 0 (otherwise). Let $X_i = \{1, 0\}$ be a Bernoulli random variable indicating if v_i has this attribute. We can model each component $P^{(k)}$ as a *Bernoulli distribution*. Let $\boldsymbol{p}^{(k)} = (\boldsymbol{p}^{(k)}(1), \boldsymbol{p}^{(k)}(2))^T$ be the outcome probabilities in this distribution. $\boldsymbol{p}^{(k)}(1) + \boldsymbol{p}^{(k)}(2) = 1.$[1] $\boldsymbol{p}^{(k)}(1)$ is the probability for a vertex to contain the attribute in this mixture component.

$$P^{(k)}(v_i) = \boldsymbol{p}^{(k)}(1)^{X_i}(1 - \boldsymbol{p}^{(k)}(1))^{1-X_i}. \tag{3.2}$$

gAnomaly is extensible to more complicated data models. If there are multiple independent attribute types,

[1]In this paper, we typeset vectors in boldface (e.g., $\boldsymbol{p}^{(k)}$) and use parentheses to denote an element in the vector (e.g., $\boldsymbol{p}^{(k)}(1)$).

we can model each of them separately and multiply the likelihoods. If an attribute has more than two distinct values, we can model $P^{(k)}$ using a categorical distribution. gAnomaly can also be extended to multiple levels of anomalies, by setting the number of mixture components to be greater than two.

In order to determine the *best* component model to describe each vertex, we fit the model with the observed data and compute the total data likelihood of V,

$$L(V) = \prod_{i=1}^{N} P(v_i) = \prod_{i=1}^{N} \sum_{k=0}^{1} \left(\theta_i^{(k)} P^{(k)}(v_i)\right). \tag{3.3}$$

For computational reasons, we compute the log-likelihood to turn multiplication to addition,

$$\ell(V) = \sum_{i=1}^{N} \log P(v_i) = \sum_{i=1}^{N} \log \sum_{k=0}^{1} \left(\theta_i^{(k)} P^{(k)}(v_i)\right). \tag{3.4}$$

However, simply maximizing the above likelihood overlooks the network structure. It will generate the same estimates even if we change the edge structure of G. In fact, it will just group all black vertices together as the anomaly and leave the white vertices as the background. We thus employ network regularization to smoothen the connectivity in each mixture component.

3.2 Network Regularizer If we group vertices based on their color, it is bound to produce the highest data likelihood. However, such assignment produces little practical value, because the vertices within the same mixture component (class) are spread out in the graph. In reality, it is desirable for vertices in the same class to exhibit satisfying connectivity. A distinctive feature of our model is to smoothen the mixture weights across the graph, so that neighboring vertices have similar model memberships. Inspired by the **NetPLSA** model in [11], we employ a graph-based discrete regularizer. Such harmonic regularization is succinct and intuitive: vertices which are connected should have similar model membership priors, thus the mixture weights. Let $\mathbf{\Theta}$ be the $N \times 2$ mixture weights matrix, where $\mathbf{\Theta}(i, k+1) = \theta_i^{(k)}$ is the mixture weight vertex v_i has for component $P^{(k)}, k = \{0, 1\}$. Let $\boldsymbol{M}_i$ denote the i-th row in a matrix $\boldsymbol{M}$. The network regularizer in [11], $R_N^{(0)}(\mathbf{\Theta})$, is formulated as,

$$\begin{aligned} R_N^{(0)}(\mathbf{\Theta}) &= \frac{1}{2} \sum_{(v_i,v_j)\in E} \sum_{k=0}^{1} (\theta_i^{(k)} - \theta_j^{(k)})^2 \\ &= \frac{1}{2} \sum_{(v_i,v_j)\in E} \|\mathbf{\Theta}_i - \mathbf{\Theta}_j\|^2, \end{aligned} \tag{3.5}$$

where $\|\cdot\|$ is the l_2 norm of a vector. The essence of $R_N^{(0)}(\mathbf{\Theta})$ is: by deducting this term from the data log-likelihood in Equation (3.4), we can minimize the sum of squared differences of the mixture weights of all connected vertex pairs in G. $R_N^{(0)}(\mathbf{\Theta})$ is used to smoothen the topic proportions of neighboring documents in [11].

Such regularization finds anomaly vertices with satisfying connectivity among each other. Nonetheless, $R_N^{(0)}(\mathbf{\Theta})$ suffers from one major drawback: it is biased towards high-degree vertices. In other words, high-degree vertices are affected more by the network regularizer than small-degree vertices. As a result, small-degree vertices are usually separated to one component while high-degree vertices stay together in the other component. We call this the *neighborhood size effect.* To alleviate this problem, we propose two variations of the network regularizer.

[Type 1: Minimizing Mean] $R_N^{(1)}(\mathbf{\Theta})$ minimizes the sum of the average difference between the mixture weights of a vertex and those of its neighbors, for all vertices in G. $R_N^{(1)}(\mathbf{\Theta})$ is essentially a vertex degree-normalized version of $R_N^{(0)}(\mathbf{\Theta})$.

$$\begin{aligned} R_N^{(1)}(\mathbf{\Theta}) &= \frac{1}{2} \sum_{v_i\in V} \frac{1}{|N(i)|} \sum_{v_j\in N(i)} \sum_{k=0}^{1} (\theta_i^{(k)} - \theta_j^{(k)})^2 \\ &= \frac{1}{2} \sum_{v_i\in V} \frac{1}{|N(i)|} \sum_{v_j\in N(i)} \|\mathbf{\Theta}_i - \mathbf{\Theta}_j\|^2, \end{aligned} \tag{3.6}$$

where $N(i)$ is the set of neighbors of vertex v_i, and $|\cdot|$ denotes the cardinality of a set.

[Type 2: Minimizing Minimum] $R_N^{(2)}(\mathbf{\Theta})$ minimizes the smallest difference between the mixture weights of a vertex and those of its neighbors, for all vertices in G.

$$R_N^{(2)}(\mathbf{\Theta}) = \frac{1}{2} \sum_{v_i\in V} \underset{v_j\in N(i)}{min} \|\mathbf{\Theta}_i - \mathbf{\Theta}_j\|^2. \tag{3.7}$$

Unlike the original network regularizer $R_N^{(0)}(\mathbf{\Theta})$, the effect of vertex degrees is discounted by either the *average* or the *minimum* function in our proposed regularizers. The rationales behind $R_N^{(1)}(\mathbf{\Theta})$ and $R_N^{(2)}(\mathbf{\Theta})$ are very different. $R_N^{(1)}(\mathbf{\Theta})$ makes a vertex *close* to the majority of its neighbors with respect to mixture weights. Using $R_N^{(1)}(\mathbf{\Theta})$ helps a vertex to be assigned to the same class as most of its neighbors. Intuitively, this contributes to larger connected regions in each mixture component. In contrast, $R_N^{(2)}(\mathbf{\Theta})$ makes a vertex close to the neighbor that it has the most similar mixture weight with, which most likely shares the same attribute value as the vertex itself. Therefore $R_N^{(2)}(\mathbf{\Theta})$ further contributes to high attributive homogeneity in each mixture component. Section 7 empirically compares these two regularizers.

Regularizer choices depend on applications and their information propagation mechanism. Therefore there are alternative formulations. For example, instead of minimizing the minimum difference, we can minimize the maximum difference. We can also modify Equation (3.6). Instead of assigning the same normalization weight $(1/N(i))$ to all the neighbors, different emphasis can be assigned to different neighbors. For instance, we can assign higher weights to neighbors sharing the same attribute as the root vertex. We experimented with such alternatives, and observed no consistent advantages over the aforementioned two regularizers. Thus in this paper, we focus on $R_N^{(1)}(\mathbf{\Theta})$ and $R_N^{(2)}(\mathbf{\Theta})$.

3.3 Entropy Regularizer

We further introduce the entropy regularizer. It is possible that the learned two mixture components are somewhat similar, because G might not contain regions whose distribution is drastically different. In this case, the learned vertex mixture weights are fairly balanced across the two components. In order to assign vertices with confidence, we aim for more sparsified or biased mixture weights. Since sparsified mixture weights correspond to lower Shannon entropy, we incorporate an entropy regularizer to materialize this. It uses the sum of the negative entropy functions of all vertices, on the mixture weights. Intuitively by favoring a larger value of such regularizer, it results in more biased mixture weights. That is, the mixture weights tend to be more focused on one component, instead of being balanced across the two. Let $R_E(\mathbf{\Theta})$ denote the entropy regularizer.

$$R_E(\mathbf{\Theta}) = \sum_{i=1}^{N}(\mathbf{\Theta}_i \log \mathbf{\Theta}_i^T) = \sum_{i=1}^{N}\sum_{k=0}^{1}(\theta_i^{(k)} \log \theta_i^{(k)}). \tag{3.8}$$

We now enrich the original mixture model with the network and entropy regularizers. The regularized data likelihood over the entire vertex set has the form:

$$\begin{aligned}\hat{\ell}(V) &= \ell(V) - \lambda R_N^{(\tau)}(\mathbf{\Theta}) + \gamma R_E(\mathbf{\Theta}) \\ &= \sum_{i=1}^{N} \log \sum_{k=0}^{1}(\theta_i^{(k)} P^{(k)}(v_i)) \\ &\quad - \lambda R_N^{(\tau)}(\mathbf{\Theta}) + \gamma \sum_{i=1}^{N}(\mathbf{\Theta}_i \log \mathbf{\Theta}_i^T), \end{aligned} \tag{3.9}$$

where $\ell(V)$ is the log-likelihood of the mixture model, $R_N^{(\tau)}(\mathbf{\Theta})$ is one of the network regularizers, and $R_E(\mathbf{\Theta})$ is the entropy regularizer. λ and γ are the coefficients associated with each regularizer, respectively. Note that such a regularization framework can be generalized to other forms of likelihood and regularization functions.

4 Parameter Estimation

Given the above mixture model, we detect anomalies as follows: (1) learn the model parameters through maximizing the overall data likelihood; (2) assign each vertex to the best component, the anomaly or the background, based on the learned parameters.

Although the *expectation-maximization* (EM) [15] algorithm has been widely used to approximate the *maximum likelihood estimates* (MLE) in mixture model learning, a standard EM suffers from the drawbacks of local optima and initialization dependence. To address such drawbacks, *deterministic annealing* EM (DAEM) has been proposed and applied to various mixture model scenarios [7, 15]. DAEM reformulates the log-likelihood maximization as the problem of minimizing the free energy function by using a statistical mechanics analogy. The posterior probability of latent variables further includes a “temperature” parameter which controls the influence of unreliable model parameters. The annealing process of adjusting the temperature is able to reduce the dependency on initial model parameters. Therefore in this paper we adopt the DAEM approach, as outlined in Algorithm 1, to learn our model parameters.

In the DAEM algorithm, maximizing the log-likelihood of our mixture model, as shown in Equation (3.4), is reformulated as the problem of minimizing a *free energy* function

$$f_\beta(\Phi) = -\frac{1}{\beta}\sum_{i=1}^{N} \log \sum_{k=0}^{1}(\theta_i^{(k)} P^{(k)}(v_i))^\beta, \tag{4.10}$$

where $1/\beta$ is called the “temperature”, and Φ is the set of model parameters, $\{\mathbf{\Theta}, \boldsymbol{p}^{(k)}\}$. The temperature is initialized at a high value and decreases gradually as the iterations proceed. When $\beta = 1$, the negative free energy becomes the log-likelihood of our mixture model.

Maximizing the regularized data likelihood in Equation (3.9) is then reformulated into minimizing a regularized free energy function, $\hat{f}_\beta(\Phi)$, where the network and entropy regularizers are kept unchanged,

$$\hat{f}_\beta(\Phi) = f_\beta(\Phi) + \lambda R_N(\mathbf{\Theta}) - \gamma R_E(\mathbf{\Theta}). \tag{4.11}$$

We now present details on the expectation (E) step and maximization (M) step in the DAEM procedure.

4.1 E Step

Let z_i be the latent membership of vertex v_i. $z_i = k$ means that v_i is assigned to component k. Let Φ^t be the current estimate of the model parameters, and $w_i^{(k)}$ be the posterior probability of $z_i = k$. In the E step of a standard EM procedure, we calculate the posterior distribution of z_i as

$$\begin{aligned} w_i^{(k)} &= P(z_i = k | v_i; \Phi^t) \\ &= \frac{\theta_i^{(k)} P(v_i | z_i = k; \boldsymbol{p}^{(k)})}{\sum_{l=0}^{1} \theta_i^{(l)} P(v_i | z_i = l; \boldsymbol{p}^{(l)})}. \end{aligned} \tag{4.12}$$

The expectation of the complete log-likelihood with respect to the posterior distribution $P(z_i | v_i; \Phi^t)$ is:

$$Q(\Phi | \Phi^t) = \sum_{i=1}^{N}\sum_{k=0}^{1} w_i^{(k)} \log \left(\theta_i^{(k)} P^{(k)}(v_i)\right). \tag{4.13}$$

Algorithm 1: DAEM-Based Model Learning

```
Input: $G = (V, E, \mathbb{A})$
Output: Estimated parameters, $\boldsymbol{p}^{(k)}$'s and $\boldsymbol{\Theta}$
1 Initialize the mixture model parameters, and
  $\beta = \beta^{(0)} (0 < \beta^{(0)} < 1)$;
2 while $\hat{f}_\beta(\Phi)$ is not converged and $\beta \leq 1$ do
3     E step: estimate the latent memberships of each
      vertex, $w_i^{(k)}(\beta)$'s;
4     M step: update the model parameters via
      minimizing $\mathscr{F}_\beta(\Phi|\Phi^t)$;
5     Increase temperature parameter $\beta$;
6 Assign each vertex $v_i$ to a component;
7 return Estimated $\boldsymbol{p}^{(k)}$'s, $\boldsymbol{\Theta}$;
```

In DAEM, the posterior probability of belonging to either mixture component further contains the temperature parameter β. Let $w_i^{(k)}(\beta)$ denote this new posterior probability. As in [15], it is given by

$$\begin{aligned} w_i^{(k)}(\beta) &= P(z_i = k | v_i; \Phi^t, \beta) \\ &= \frac{\left(\theta_i^{(k)} P(v_i | z_i = k; \boldsymbol{p}^{(k)})\right)^\beta}{\sum_{l=0}^{1} \left(\theta_i^{(l)} P(v_i | z_i = l; \boldsymbol{p}^{(l)})\right)^\beta}. \end{aligned} \tag{4.14}$$

Using *Jensen's inequality*, we have

$$\begin{aligned} f_\beta(\Phi) &= -\frac{1}{\beta} \sum_{i=1}^{N} \log \sum_{k=0}^{1} w_i^{(k)}(\beta) \frac{(\theta_i^{(k)} P^{(k)}(v_i))^\beta}{w_i^{(k)}(\beta)} \\ &\leq -\sum_{i=1}^{N} \sum_{k=0}^{1} w_i^{(k)}(\beta) \log(\theta_i^{(k)} P^{(k)}(v_i)). \end{aligned} \tag{4.15}$$

In the E step of DAEM, we estimate the expectation of the complete log-likelihood with respect to the new posterior distribution $P(z_i | v_i; \Phi^t, \beta)$ as follows, and $f_\beta(\Phi)$ is upper bounded by $-\hat{Q}_\beta(\Phi|\Phi^t)$.

$$\hat{Q}_\beta(\Phi|\Phi^t) = \sum_{i=1}^{N} \sum_{k=0}^{1} w_i^{(k)}(\beta) \log \left(\theta_i^{(k)} P^{(k)}(v_i)\right). \tag{4.16}$$

4.2 M Step M step boils down to minimizing a regularized upper bound function, $\mathscr{F}_\beta(\Phi|\Phi^t)$:

$$\mathscr{F}_\beta(\Phi|\Phi^t) = -\hat{Q}_\beta(\Phi|\Phi^t) + \lambda R_N(\boldsymbol{\Theta}) - \gamma R_E(\boldsymbol{\Theta}). \tag{4.17}$$

In gAnomaly, we adopt the widely used L-BFGS [10] algorithm for the optimization. Instead of storing the dense Hessian approximation matrix during optimization, L-BFGS saves only a few vectors to represent the approximation, which significantly reduces the memory requirement. Since L-BFGS is a classic and well-established algorithm [10], we omit the detailed analysis on its complexity and convergence rate.

4.3 Vertex Assignment Upon the convergence of the iterative DAEM process, the next step is to assign each vertex to either the anomaly or the background mixture component. We can subsequently use such assignment to uncover anomaly regions in G. Let z_i^* be the component/class label that v_i is assigned to after convergence. z_i^* can be estimated using the updated mixture weight matrix at the time of convergence.

$$z_i^* = \operatorname*{argmax}_k \theta_i^{(k)}, \tag{4.18}$$

where z_i^* is estimated as the label of the mixture component for which v_i has the highest mixture weight.

5 Iterative Anomaly Detection

gAnomaly so far makes two assumptions: (1) There are only two generative processes, anomaly and background. (2) The size of the anomaly is comparable to that of the background. However, real-world networks might violate such assumptions. We therefore propose an iterative procedure to account for challenging scenarios: (1) There are more than two attribute distributions. (2) The size of the anomaly is small. This procedure iteratively removes the background and refocuses the search within the anomaly. In each iteration, the two-component mixture model is applied on the current graph, and classify the vertices into anomaly and background. Only the anomaly-induced subgraph is fed into the next iteration. This continues until an anomaly with desirable size and attribute distribution is found.

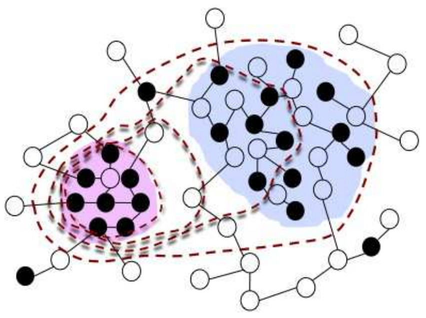


Figure 2: Iterative Fine-Grained Anomaly Detection

Figure 2 shows how such iterative process works on a small example. Suppose we have a graph whose vertex attributes are generated by three distributions, an anomaly distribution with the highest probability of black vertices, a background distribution with the lowest probability of black vertices, and a noise distribution that lies in between. The red region is the anomaly region, the blue region is the noise region, and the rest is the background. If the noise region is significantly larger than the anomaly region, a non-iterative model will most likely fail to identify the red anomaly region.

We address this through iteration. As shown, the iteration starts off with labeling both the anomaly and noise as anomaly, and gradually shrinks the boundary of the anomaly (red dotted line), until the most anomalous region of the graph is located. There are two possible stop conditions: (1) The iteration stops when the uncovered anomaly size is $\pm\rho$ of the desirable size (ρ is user-specified); or (2) it stops when the percentage of black vertices in the anomaly exceeds a threshold.

6 Performance Measurement

We evaluate the algorithms using the largest anomaly, S_0, discovered in the original graph. Two variables are extracted: (1) the percentage of black vertices in S_0; (2) the number of vertices in S_0.

6.1 Mahalanobis Distance

M-distance is a multivariate version of z-score. It gauges the distance of an observation from the centroid of a multivariate distribution, given the covariance of the distribution. We use it to evaluate if the pattern found in the original graph is a multivariate outlier against random cases. The following steps are taken: (1) Retrieve the pattern S_0 in G. Let $\mathcal{B}(S_0)$ and $|S_0|$ be the percentage of black vertices in S_0, and the size of S_0, respectively. (2) Randomly shuffle the vertex attributes in G, while keeping the total number of black vertices. Let $\{G_1, \ldots, G_r\}$ be the r randomly-shuffled graphs. (3) Create a reference random sample set by retrieving the pattern S_i from G_i in the same manner. Let $\mathcal{B}(S_i)$ and $|S_i|$ be the respective two variables. (4) Let $\vec{S_i} = (\mathcal{B}(S_i), |S_i|)^T, i = \{0, 1, \ldots, r\}$, and $\vec{\mu}$ be the mean of the random samples $\{\vec{S_1}, \ldots, \vec{S_r}\}$. The M-distance of $\vec{S_0}$ is computed as:

$$D_M(\vec{S_0}) = \sqrt{(\vec{S_0} - \vec{\mu})^T \Sigma^{-1} (\vec{S_0} - \vec{\mu})}, \tag{6.19}$$

where Σ is the covariance matrix of the r random samples. A larger M-distance means a higher deviation from random cases.

6.2 Pattern Probability

We use pattern probability to measure how "rare" a pattern is in G, without considering the pattern structure. We model an anomaly region using a binomial distribution. Let P_b denote the percentage of black vertices in G. P_b is considered as the probability to observe black vertices. Let N_0^b be the number of black vertices in S_0. The probability of observing N_0^b or more black vertices from a set randomly chosen vertices with size $|S_0|$ is:

$$P(S_0) = \sum_{n=N_0^b}^{|S_0|} \binom{|S_0|}{n} P_b^n (1 - P_b)^{|S_0|-n}. \tag{6.20}$$

A smaller pattern probability means a higher abnormality of S_0.

7 Experimental Evaluation

We evaluate gAnomaly using both synthetic and real-world networks. For the ease of presentation, we only show results using one attribute of interest with binary values in each network.

Last.fm Network. This is a subgraph of the Last.fm network in [18], with 5,000 users and 6,789 friendships. Vertex attributes are the artists the users listened to. Black vertices contain the most popular attribute "Radiohead" (1,978 vertices) [18].

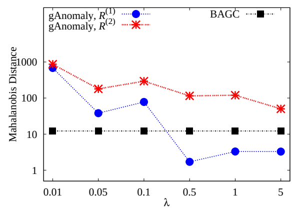
Synthetic Last.fm Networks. Each synthetic network is the previous Last.fm network annotated with synthetic vertex attributes. (1) Group I: The attributes are generated from two distributions: anomaly and background. The black probability for the anomaly, ω_A, varies from 60%, 70%, 80% to 90%; for the background, it varies from 40%, 30%, 20% to 10%. The anomaly region size is 30% of the network (randomly chosen). (2) Group II: The attributes are generated from three distributions: anomaly, noise and background. The black probability in each is 95%, 30%, and 5%, respectively. The anomaly region size varies from 1%, 5%, to 10%, and the noise region size is 20%. The purpose of Group II networks is to evaluate the iterative detection process for fine-grained anomaly detection.

Cora Network. The Cora network[2] contains 2,708 publications and their 5,429 citations relations. Publications are attributed as: {"Case Based", "Genetic Algorithms", "Neural Networks", "Probabilistic Methods", "Reinforcement Learning", "Rule Learning", "Theory"}. Black vertices are those containing the most prevalent attribute, "Neural Networks" (818 vertices).

DBLP Networks. A network of 6,307 authors and 8,709 collaborations is extracted from DBLP. There is an edge between two authors if they have coauthored at least five papers. Each author is attributed with a research field. We use two versions of this network: DBLP-IR and DBLP-DM, where black vertices contain the attribute "Information Retrieval" (795 vertices) and "Data Mining" (1,096 vertices), respectively.

We compare gAnomaly with the state-of-art probabilistic graph clustering algorithm BAGC (Bayesian Attributed Graph Clustering) [21], a Bayesian model where vertices grouped into one cluster share common attribute and edge distributions. The comparative study aims to see which method is better at uncovering non-random abnormal subgraphs. The entropy regularizer coefficient γ is set as 0.5, and 100 random samples are generated for M-distance computation.

[2]http://www.cs.umd.edu/~sen/lbc-proj/LBC.html


(a) Mahalanobis Distance

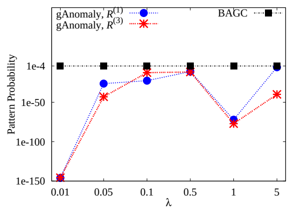
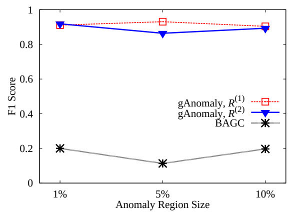

(b) Pattern Probability

Figure 3: M-Dist & Pattern-Prob on Group I, $\omega_A = 0.9$


(a) F1 vs. Anomaly Size

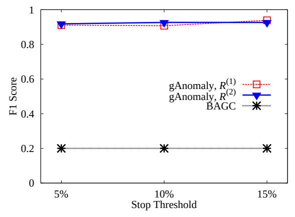
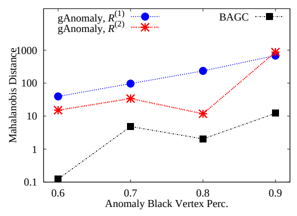

(b) F1 vs. Stop Threshold

Figure 5: F1 on Synthetic Group-II


(a) Mahalanobis Distance

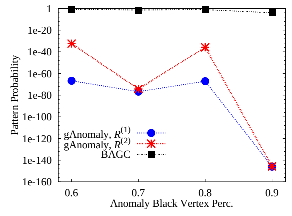

(b) Pattern Probability

Figure 4: M-Dist & Pattern-Prob on Group I, $\lambda = 0.01$

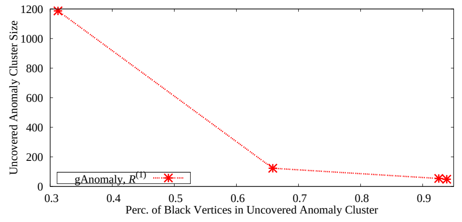
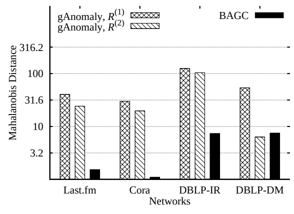


Figure 6: Convergence of gAnomaly on Group II with 1% Anomaly and 5% Stop Threshold


(a) Mahalanobis Distance

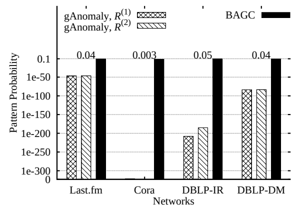

(b) Pattern Probability

Figure 7: M-Dist & Pattern-Prob on Real Networks

7.1 Results on Synthetic Data

For Group I, M-distance and pattern probability are used. We then use F1 score on Group II to evaluate gAnomaly on retrieving true anomalies in challenging scenarios.

7.1.1 Group I: Graphs With Two Generative Processes

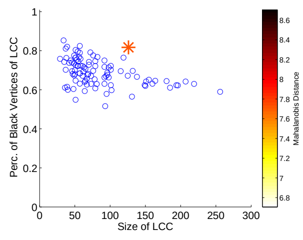
Figure 3 shows how the two metrics change with λ on the Group I synthetic network with $\omega_A = 0.9$. Both versions of gAnomaly outperform BAGC in M-distance when $\lambda <= 0.1$, and in pattern probability for all λ's. Performance of gAnomaly decreases as λ increases. This is because when λ is large, the patterns tend to be larger with less fraction of black vertices. In comparison with $R_N^{(1)}$, $R_N^{(2)}$ is less sensitive to λ, yielding good abnormality measures even for large λ. This conforms with the intuition of $R_N^{(2)}$. Figure 4 shows how gAnomaly and BAGC perform with varying ω_A on $\lambda = 0.01$. Both versions of gAnomaly significantly outperform BAGC on all networks. $R_N^{(1)}$ outperforms $R_N^{(2)}$ for $\lambda = 0.01$, since it generates larger patterns than $R_N^{(2)}$. The performance improves as the difference between anomaly and background increases.

7.1.2 Group-II: Graphs With Multiple Generative Processes

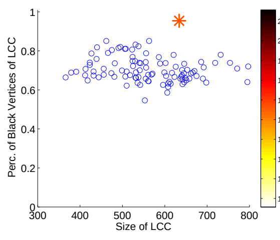
Now we evaluate the iterative detection process using more challenging networks: (1) there are more than two generative processes; (2) the fraction of anomaly is small. Group II synthetic networks are tested using F1 score to measure the quality of anomaly retrieval. Figure 5(a) shows how F1 changes with the size of the anomaly region ($\rho = 5\%$). How F1 changes with ρ (anomaly region size is fixed to 1%) is shown in Figure 5(b). With the iterative process, gAnomaly yields good F1 performance, and beats BAGC significantly. Figure 6 shows one case of iteration convergence using gAnomaly and $R_N^{(1)}$, where the anomaly region is 1%, and the stop threshold is 5%. We can see that the iteration starts off with a large anomaly region with a small fraction of black vertices; as the iteration continues, gAnomaly gradually shrinks the anomaly region and increases the percentage of black vertices within, until the iteration converges to a desirable anomaly size.

7.2 Results on Real Data

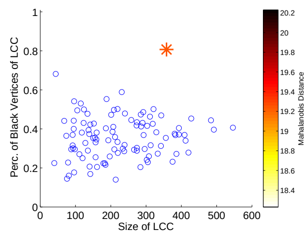
In this section, we report results on real networks. Figure 7 shows the results with $\lambda = 0.01$. gAnomaly significantly outperforms BAGC on all but DBLP-DM. The performance of gAnomaly depends on the structure and pattern distribution within the network. If the network does not contain a signif-



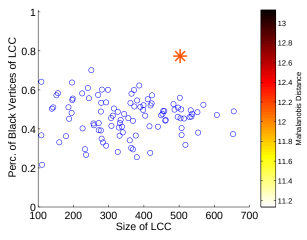
(a) gAnomaly, Last.fm



(b) gAnomaly, Cora

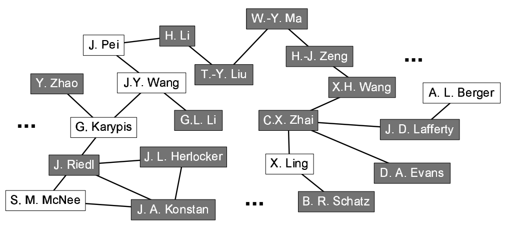


(c) gAnomaly, DBLP-IR

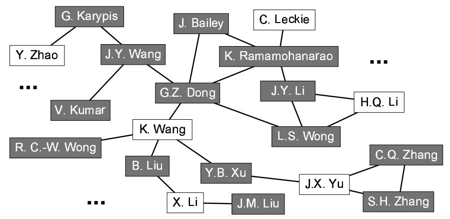


(d) gAnomaly, DBLP-DM

Figure 8: M-Distance Scatter Visualization on Real Networks



(a) DBLP-IR Pattern



(b) DBLP-DM Pattern

Figure 9: DBLP Case Studies

icant anomaly region, gAnomaly is not able to identify it; or if the anomaly consists of many cohesive patterns, BAGC will do better than gAnomaly. Figure 8 visualizes the M-distance of S_0 in each original network, to a set of patterns uncovered from randomized networks with the attributes shuffled. The blue circles are the anomalies found in the 100 random samples, $\{S_1, \ldots, S_{100}\}$, and the red star is S_0. The numeric value of the M-distance can be obtained by looking up the star color in the color bar. The distance from S_0 to the centroid of the random samples indicates the power of the algorithm for anomaly detection. $R_N^{(2)}$ is used in gAnomaly. As show, M-distance is much more significant in gAnomaly than BAGC, especially on Cora, DBLP-IR, and DBLP-DM.

7.3 Case Studies We further conduct case studies on the DBLP networks to examine the semantics of the anomalies. Figure 9 shows a portion of the anomalies gAnomaly uncovers from DBLP-IR and DBLP-DM. Black authors are those attributed as from the respective field. For example in Figure 9(a), Chengxiang Zhai has the attribute "Information Retrieval", whereas Geoge Karypis does not. We use the author attributes provided by [21]. Our observations are: (1) For a specific research field, gAnomaly uncovers a continuous region with a high concentration of authors from this field. (2) The uncovered region is not necessarily densely connected, since gAnomaly is not looking for cliques or near-cliques. (3) The region contains a small fraction of authors not from this field. gAnomaly includes such "bridge" authors so that a region can span across multiple research groups. Upon finding such anomalies, it is useful to further study user behaviors and conduct localized network influence analysis.

8 Related Work

Various studies exist on pattern mining in attributed graphs [8, 9, 13, 21, 22]. [13] introduces cohesive pattern, a connected dense subgraph with homogeneous feature values. [21] proposes a model to discover graph clusters within which vertices share common attribute and edge connection distributions. [22] addresses a similar problem through graph augmentation and a unified structural and attributive distance measure. All the above patterns tend to be dense in connectivity. In contrast, we aim to find anomalies with arbitrary edge densities.

Another related topic is graph anomaly detection [1, 2, 3, 4, 6, 12, 14, 16].[6] is closely related. However, it has to be modified significantly to suit our purposes, as its current model is not as flexible as our network regularizers. For example, it can not model the influence of the most similar neighbor. [1] finds abnormal vertices in an edge-weighted graph by examining their "ego-nets" against certain rules. [12] turns the adjacency matrix into transition matrix, and models the anomaly detection as a Markov chain process. [2] proposes a parameter-free graph clustering algorithm to find vertex groups, and further finds anomalies by computing distances between groups. Inspired by fraud detection, [4] defines a graph substructure as anomaly if it is isomorphic to the normative substructure within a certain amount of vertex or edge modifications. [14] uses the MDL principle to uncover infrequent anomalies. [3] extends [14]'s techniques to address numeric vertex attributes. [16] proposes a Bayesian spatial scan framework for event detection. All the above methods can not uncover the type of anomalies addressed in this work.

Probabilistic models are widely used in graph mining [5, 6, 11, 19, 20]. [11] extends PLSA in a network environment using a harmonic regularizer based on the network structure. [5] applies a mixture model to unsupervised intrusion detection, when the percentage of anomalous elements is small. [20] addresses feature selection via learning a Dirichlet process mixture model in the high dimensional feature space of the graph data. Many techniques have also been explored to regularize a mixture model to appeal to specific applications [11, 17].

9 Conclusions

We propose a probabilistic approach using a mixture model with regularization to detect graph anomalies, graph regions exhibiting significantly different attribute distributions. Future directions include: (1) extend the model to account for networks with rich content, such as edge direction and weight; (2) incorporate temporal information to detect anomalies in dynamic graphs; (3) conduct causal analysis among anomalies.

Acknowledgment

This research was sponsored in part by the Army Research Laboratory under cooperative agreements W911NF-09-2-0053 and NSF IIS-0954125. The views and conclusions contained herein are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notice herein.

References

[1] L. Akoglu, M. McGlohon, and C. Faloutsos. Oddball: Spotting anomalies in weighted graphs. In *PAKDD (2)*, pages 410–421, 2010.

[2] D. Chakrabarti. Autopart: Parameter-free graph partitioning and outlier detection. In *PKDD*, pages 112–124, 2004.

[3] M. Davis, W. Liu, P. Miller, and G. Redpath. Detecting anomalies in graphs with numeric labels. In *CIKM*, pages 1197–1202, 2011.

[4] W. Eberle and L. B. Holder. Discovering structural anomalies in graph-based data. In *ICDM Workshops*, pages 393–398, 2007.

[5] E. Eskin. Anomaly detection over noisy data using learned probability distributions. In *ICML*, pages 255–262, 2000.

[6] J. Gao, F. Liang, W. Fan, C. Wang, Y. Sun, and J. Han. On community outliers and their efficient detection in information networks. In *KDD*, pages 813–822, 2010.

[7] Y. Itaya, H. Zen, Y. Nankaku, C. Miyajima, K. Tokuda, and T. Kitamura. Deterministic annealing EM algorithm in acoustic modeling for speaker and speech recognition. *IEICE Transactions*, 88-D(3):425–431, 2005.

[8] N. Li, Z. Guan, L. Ren, J. Wu, J. Han, and X. Yan. gIceberg: towards iceberg analysis in large graphs. In *ICDE*, 2013.

[9] N. Li, X. Yan, Z. Wen, and A. Khan. Density index and proximity search in large graphs. In *CIKM*, pages 235–244, 2012.

[10] D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Math. Program.*, 45(3):503–528, Dec. 1989.

[11] Q. Mei, D. Cai, D. Zhang, and C. Zhai. Topic modeling with network regularization. In *WWW*, pages 101–110, 2008.

[12] H. D. K. Moonesinghe and P.-N. Tan. OutRank: a graph-based outlier detection framework using random walk. *International Journal on Artificial Intelligence Tools*, 17(1):19–36, 2008.

[13] F. Moser, R. Colak, A. Rafiey, and M. Ester. Mining cohesive patterns from graphs with feature vectors. In *SDM*, pages 593–604, 2009.

[14] C. C. Noble and D. J. Cook. Graph-based anomaly detection. In *KDD*, pages 631–636, 2003.

[15] K. Rose. Deterministic annealing for clustering, compression, classification, regression, and related optimization problems. In *Proceedings of the IEEE*, pages 2210–2239, 1998.

[16] K. Shao, Y. Liu, and D. B. Neill. A generalized fast subset sums framework for bayesian event detection. In *ICDM*, pages 617–625, 2011.

[17] L. Si and R. Jin. Adjusting mixture weights of gaussian mixture model via regularized probabilistic latent semantic analysis. In *PAKDD*, pages 622–631, 2005.

[18] A. Silva, W. M. Jr., and M. J. Zaki. Mining attribute-structure correlated patterns in large attributed graphs. *PVLDB*, 5(5):466–477, 2012.

[19] K. Tsuda and T. Kudo. Clustering graphs by weighted substructure mining. In *ICML*, pages 953–960, 2006.

[20] K. Tsuda and K. Kurihara. Graph mining with variational Dirichlet process mixture models. In *SDM*, pages 432–442, 2008.

[21] Z. Xu, Y. Ke, Y. Wang, H. Cheng, and J. Cheng. A model-based approach to attributed graph clustering. In *SIGMOD*, pages 505–516, 2012.

[22] Y. Zhou, H. Cheng, and J. X. Yu. Graph clustering based on structural/attribute similarities. *PVLDB*, 2(1):718–729, 2009.