An Embedding Approach to Anomaly Detection

Renjun Hu[†], Charu C. Aggarwal[‡], Shuai Ma[†,*] and Jinpeng Huai[†]
[†] SKLSDE Lab, Beihang University, Beijing, China
[‡]IBM T. J. Watson Research Center, New York, USA
{hurenjun, mashuai, huaijp}@buaa.edu.cn, charu@us.ibm.com

***Abstract*—Network anomaly detection has become very popular in recent years because of the importance of discovering key regions of structural inconsistency in the network. In addition to application-specific information carried by anomalies, the presence of such structural inconsistency is often an impediment to the effective application of data mining algorithms such as community detection and classification. In this paper, we study the problem of detecting structurally inconsistent nodes that connect to a number of diverse influential communities in large social networks. We show that the use of a network embedding approach, together with a novel dimension reduction technique, is an effective tool to discover such structural inconsistencies. We also experimentally show that the detection of such anomalous nodes has significant applications: one is the specific use of detected anomalies, and the other is the improvement of the effectiveness of community detection.**

I. Introduction

The problem of anomaly detection has been studied widely in the literature in the context of different data domains such as multidimensional, time-series and network data. In recent years, the detection of anomalies in networks has started attracting significant attention [1], [3], [5], [7], [15], [17], [24], [27], [30], because of the increasing attention on social network analysis models. Network anomalies are useful to discover, both from the perspective of their application-specific significance, and the possibility to improve the performance of other network-centric data mining tasks such as community detection and classification.

Networks are inherently complex entities, and, hence, anomalies may be defined in a wide variety of ways. In this paper, our goal is to discover *structural inconsistencies, i.e., the anomalous nodes that connect to a number of diverse influential communities*, inspired by the concept of social brokers across groups, which provide social capital in networks [10]. The presence of such structural inconsistencies may have a substantial impact on the structure of original networks, and prevent the effective application of the methods such as community detection to a variety of data mining problems.

To better understand structural inconsistencies, we illustrate with an example in Fig. 1. The red nodes labeled A, B and C in the network tend to connect with many different communities, instead of a single community. As a joint result, one of the common observations in many community detection algorithms is that all nodes tend to form one large cluster through preferential attachment to anomalous nodes, and only very careful tuning of the algorithms is able to achieve meaningful clusters [2]. Note that clusters, groups and communities are used alternatively in this paper.

The challenge caused by such nodes is not restricted to

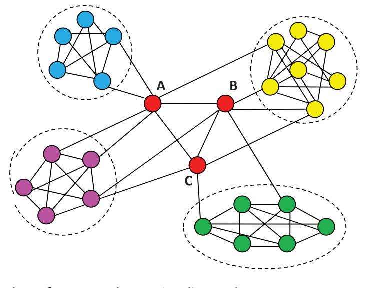


Figure 1. Example of anomalous (red) nodes

the problem of community detection. For example, the notion of *homophily*, which assumes that linked nodes have similar properties, is fundamental to the design of a wide variety of algorithms in network science. However, nodes such as A, B and C in Fig. 1 are connected to many diverse regions in the network, and, hence, the homophily assumption is violated by many of the incident links. In those cases, the presence of such nodes is also likely to lead to challenges for label propagation algorithms. The principle of homophily, which is violated by such nodes, is in fact fundamental to the effective design of algorithms such as community detection, collective classification, link prediction, and influence analysis. Therefore, the detection of such anomalous nodes, from its own right, has rather broad potential applicability from an application-specific perspective, such as structural hole broker detection [10] and spam node detection in large networks [5].

Contributions. In this paper, we will design an approach to detecting anomalous nodes in large networks, based on a network embedding methodology.

(1) While a variety of graph embeddings, such as multidimensional scaling [8], are available in the literature, they aim to preserve (global) pairwise similarities and are not optimized to networks and the problem of anomaly detection. Hence, they cannot be directly used for the detection of structural inconsistencies proposed in this paper. Therefore, a novel embedding method will be developed, which is specifically designed to ferret out the anomalous nodes in large networks.

The embedding approach is based on a model, in which each dimension of the embedding corresponds to a clustered region in the network. In other words, the similarity of different nodes along a particular dimension, indicates their similarity to a particular clustered region. Therefore, this embedding retains a very high level of interpretability in terms of the original graph data, which is very useful from an application-specific perspective. The nature of the embedding also makes it possible to detect anomalous nodes, by examining the interaction of each node with the different regions in terms of the embedding. In particular, we measure the *level of anomalousness* of a node in terms of the embedding imposed on the node and its neighbors.

(2) Such an approach is however rather hard to apply to the case of large networks, because the complexity of the approach is in proportion to the square of the number of nodes when optimizing the embedding and because the noises in the embedding seriously impair the accuracy of detected anomalous nodes. Hence, we incorporate the sampling and graph partitioning techniques, and, moreover, propose a novel dimension reduction technique to make the approach more scalable and effective for large networks.

(3) Using both real-life data (DBLP and AMAZON) and synthetic data (SYNTHETIC), we conduct an extensive experimental study. We find that our embedding approach to network anomaly detection is both effective and efficient. (a) The modularity [11] was increased about 2.9% and 2.4% for AMAZON, 4.9% and 4.2% for DBLP and 6.3% and 4.2% for SYNTHETIC, with our approach and OddBall [5], respectively; (b) The F_1 measure was about 88% and 70% for SYNTHETIC with our approach and OddBall, respectively; And (c) the running time of our embedding approach increases reasonably with the increase of graph sizes, even with large number of communities, while traditional multidimensional scaling approach [8] ran out of memory.

Organization. Section II reviews the related work. The basic intuition and model behind the approach are discussed in Section III. The algorithm and its optimizations for node anomaly detection are discussed in Sections IV and V, respectively. Experiments are discussed in Section VI, followed by conclusions in Section VII.

II. RELATED WORK

Anomaly detection. While, the problem of outlier detection is well studied in its own right [1], the problem in graph data has only recently started receiving attention in the literature [5], [24]. Many of these graph anomaly detection techniques use specific models for anomaly detection, such as the low random-walk-based similarity between neighbors of a given node [28], the violation of specific power laws in the locality of a node [5], the heavy entries in the non-negative residual matrix of matrix factorization [30], and the detection of "groups" of anomalous nodes [16]. Recent work has also investigated the impact of temporal aspects of the data on outlier detection [7], [17], [25], [28]. However, none of these methods model anomalous nodes in terms of their impact on network mining algorithms because of the inconsistency in the link structures. The impact of anomalous nodes on network mining techniques has been observed in numerous recent works [2], [19]. These methods show that the presence of such anomalies can have a very detrimental impact on locality-centric network mining algorithms. However, a principled approach to discovering such nodes is not discussed by these methods.

Graph embedding. Traditional graph embedding is an approach to finding the desired low-dimensional representations of a graph that best characterize the similarity relationships between node pairs [32]. Many methods have been proposed for graph embedding, including multidimensional scaling [8], fastmap [14], isomap [29], locally linear embedding [26], stochastic neighbor embedding [18] and spectral methods [31]. However, they are designed for compressing graph while preserving certain properties. For instance, the methods in [8], [14] preserve global Euclidean distances over the entire graph, and the one in [29] preserves the shortest path distances over a derived neighborhood graph. Both can not be directly used for detecting structural inconsistencies. While the methods in [18], [26], [31] do preserve local structure, they are, however, designed to preserve relationships between nodes and their k-nearest neighbors [26], [31] or the probability that a node chooses another as its neighbor [18], which are not for detecting structural inconsistencies as well.

Graph embedding based anomaly detection. There already have been works investigating graph embedding for anomaly detection [4], [15], [27]. While [4] shows the possibility of using embedding for outlier detection, an automatic detection method remains missing. [27] uses the commute time distance for detecting anomalies in dynamic graphs, where the eigenspace embedding only serves to approximately compute the commute time distance, and [15] proposes to use the spectral embedding to reveal anomalous community structure across multiple sources. Different from these works, our approach adopts a new measure to evaluate the level of anomalousness, incorporates the graph partitioning technique, and proposes a novel dimension reduction technique to make the approach more scalable and effective for large networks.

Other related concepts. Structural inconsistencies that we consider bear similarities with betweenness centrality [9], [34], *i.e.,* the number of shortest paths among all node pairs that pass through that node. However, structural inconsistencies focus on node anomalies caused by edges across distinct communities, while betweenness centrality does not distinguish edges in or across communities. Further, structural hole spanners studied in [22] are those nodes whose removal disconnects groups, significantly different from structural inconsistencies.

III. NODE ANOMALY DETECTION: BASIC MODEL

The major problem caused by the nodes of anomalous structure is that they "bring together" diverse nodes, which "ought not" be connected together by short paths. So the question here is to decide how to identify such nodes which bring together diverse portions of the network, i.e., *those nodes that connect to a number of diverse influential communities*. One problem, which is unique to network data (with respect to multidimensional data), is that the nodes are not associated with any inherent positional information, as a result of which it is much harder to assess the diversity or similarity of the underlying representation, and which portions of the network may be structurally inconsistent with others. A natural approach is to use embedding in order to associate each node with a multidimensional position, and to determine the nodes which cause high level of "*anomalousness*" with the embedding.

Graph embedding. While existing graph embedding approaches, such as multidimensional scaling, are commonly used in the literature for low dimensional representations of networks, the approach is not designed for identification of anomalous nodes. This is because the approach works with a (global) distance matrix, rather than the *local linkage structure* and *community structure*, which are the key to the identification of inconsistencies. Therefore, we will design a novel embedding approach, which is specifically designed for the problem of structural anomaly detection.

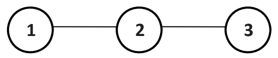


Figure 2. A simple network containing three nodes

Before introducing the model in detail, we will introduce some notations and definitions. We assume that we have an undirected graph $G = (V, E)$, with node set V, and edge set E. The cardinality of the node set V is n, and the cardinality of the edge set E is m. The nodes in V are consecutively labeled integers in the range of $\{1 \ldots n\}$ for notational convenience.

Each node $i \in V$ is associated with a d-dimensional data point $\overline{X_i}$, which corresponds to its embedded representation. The d dimensions of the embedding are denoted by $\overline{X_i} = (x_i^1, \ldots, x_i^d)$, where x_i^k represents the correlation between node i and community k. In this model, communities are not restricted to real-life ones. Hence, number d is essentially not required to be equal to the number of communities in networks detected by some algorithms. The goal in this embedding is to ensure that connected data points (nodes) have similar values of $\overline{X_i}$, and data points, which are not connected, have diverse values of $\overline{X_i}$. Thus, the *goal* of the embedding would be to find a multidimensional representation $\overline{X_1}, \ldots, \overline{X_n}$ for the nodes in V, so that the following holds true:

$$||\overline{X_i} - \overline{X_j}|| = \begin{cases} 0 & (i,j) \in E \\ 1 & (i,j) \notin E \end{cases} \tag{1}$$

Note that here (a) $||\overline{X}||$ is the (Euclidean) norm of vector $\overline{X}$; (b) The embedding ensures that the norm of any two vectors' difference is always equal to or less than 1, by imposing non-negative constraints on $x_i^1, \ldots, x_i^d$ and an upper bound of value $\sqrt{2}/2$ for $||\overline{X_i}||$; (c) The embedding aims to directly preserve local linkage structure rather than the pairwise distance of multidimensional scaling [8]; And (d) the embedding initialization is based on graph partitions revealing the community structures, to be seen in Section IV.

While it is the goal to obtain such an embedding, it would almost always be infeasible, to derive such an embedding in practice, even for very small networks. In order to illustrate this point, consider the simple network illustrated in Fig. 2 containing three nodes labeled 1, 2, and 3. Since the three nodes are connected, they ought to have the same embedded representation, in order to ensure that $\overline{X_1} = \overline{X_2} = \overline{X_3}$. However, since nodes 1 and 3 are not connected, which means $||\overline{X_1} - \overline{X_3}|| = 1$, there is no way to find an embedding which will *exactly* satisfy both constraints. However, a good 1-dimensional *real* embedding, which *approximately* satisfies the goals of the ideal embedding is $x_1 = 0$, $x_2 = 1/3$, and $x_3 = 2/3$. The *stress* $S(\overline{X_i}, \overline{X_j})$ along a node pair in such an approximate embedding, is the amount by which the goals of the *ideal* embedding are violated:

$$S(\overline{X_i}, \overline{X_j}) = \begin{cases} ||\overline{X_i} - \overline{X_j}||^2 & (i,j) \in E \\ (||\overline{X_i} - \overline{X_j}|| - 1)^2 & (i,j) \notin E \end{cases} \tag{2}$$

The stress indicates how poorly the node pair fails to achieve the goals of the embedding. Therefore, the problem of determining the optimum embedding can be formulated as a minimization problem over the objective function O:

$$O = \sum_{(i,j)\in E} S(\overline{X_i}, \overline{X_j}) + \alpha \cdot \sum_{(i,j)\notin E} S(\overline{X_i}, \overline{X_j}) \tag{3}$$

$$= \sum_{(i,j)\in E} ||\overline{X_i} - \overline{X_j}||^2 + \alpha \cdot \sum_{(i,j)\notin E} (||\overline{X_i} - \overline{X_j}|| - 1)^2 \tag{4}$$

Here α is a balancing factor, which regulates the importance of the constraints associated with the edges and the non-edges as well. Note that it is typically desirable to pick values of $\alpha < 1$, in order to ensure that the non-edge constraints do not dominate the objective function O. Typically, most real networks are sparse, and the non-edges contribute to the vast majority of the constraints. Therefore, this problem is formulated as an optimization problem, which can be solved in order to determine the optimal values of the embedding.

Anomaly detection with embedding. We now formally introduce how to detect structural inconsistencies using the embedding of a node and its neighbors. For each node i in graph G, let its neighbors be $NB(i) = \{j \mid (i,j) \in E\}$, and its embedding be $\overline{X_i} = (x_i^1, \ldots, x_i^d)$.

Firstly, we define the embedding $\overline{NB(i)}$ of $NB(i)$ as the weighted sum of the embedding of all neighbors of node i:

$$\overline{NB(i)} = (y_i^1, \ldots, y_i^d) = \sum_{j\in NB(i)} (1 - ||\overline{X_i} - \overline{X_j}||) \cdot \overline{X_j} \tag{5}$$

Note that the neighbors closer to node i in the embedding space have a higher weight, and $||\overline{X_i} - \overline{X_j}||$ is the square root of stress $S(\overline{X_i}, \overline{X_j})$ that measures the distance between nodes i and j. Intuitively, we use $\overline{NB(i)}$ to represent the correlation of node i with the d communities (instead of using $\overline{X_i}$ for node i alone), and the higher the value y_i^k is, the more node i tends to belong to community k ($k \in [1, d]$). Hence, $\overline{NB(i)}$ is able to be used to detect the node anomalies that connect to a number of diverse influential communities.

Then, given the embedding $\overline{NB(i)} = (y_i^1, \ldots, y_i^d)$ of node i, we introduce a measure, referred to as $AScore$, to indicate the (normalized) anomalousness level of node i:

$$AScore(i) = \sum_{k=1}^{d} \frac{y_i^k}{y_i^*},\ y_i^* = \max\{y_i^1, \ldots, y_i^d\} \tag{6}$$

We further refine the above equation to deal with *noises* and to distinguish *influential communities*. (a) Obviously nodes connect to communities to which they belong with a certain portion of edges. However, they often connect to communities to which they do not belong with a small number of (noisy) edges as well, and these communities are *non-influential*. To alleviate this noisy situation, for each neighbor $j \in NB(i)$, the entries in $\overline{X_j}$ whose values fall below the average one of $\overline{X_j}$ are regarded as non-influential and are replaced with 0. And (b) to distinguish influential communities from non-influential ones, we also replace those entries y_i^k in $\overline{NB(i)}$ ($k \in [1, d]$) with 0 if $y_i^k < \theta \cdot y_i^*$, where θ is a parameter in $[0, 1]$. That is, only those non-noisy communities that play a significant impact on $\overline{NB(i)}$ are *influential communities* of node i.

Finally, when $AScore(i) > thre$, node i is treated as an anomaly, which indicates that node i connects to *a number of diverse influential communities*. Intuitively, threshold $thre$ reflects the diversity of influential communities, and its choice depends on the strength of community structures in networks, *i.e.,* the proportion of inner-community and inter-community edges. Networks with strong community structures have a smaller $thre$, since the maximum entry y_i^* is very large because of a large number of inner-community edges, while the other entries are very small as a result of a small number of inter-community edges. On the other hand, networks with

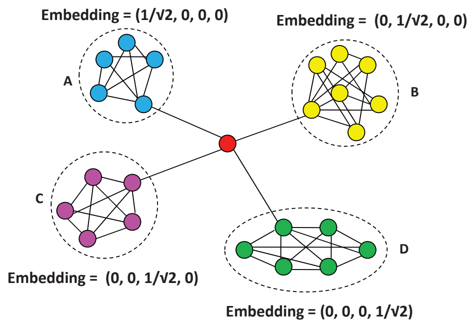

Embedding = (1/√2, 0, 0, 0)
Embedding = (0, 1/√2, 0, 0)
A
B
C
D
Embedding = (0, 0, 1/√2, 0)
Embedding = (0, 0, 0, 1/√2)

Figure 3. Anomalous (red) nodes in embedding

weak community structures have a larger $thre$. Typically, threshold $thre$ falls in $[1.5, 10]$.

We illustrate our node anomaly detection model with an example as follows.

Example 1: Consider the toy example in Fig. 3, a simplified version of the earlier example in Fig. 1. This graph contains four tightly knitting communities (almost cliques) A, B, C, D, all of which are connected to the central inconsistent node in red color. Also consider the case, where the dimensionality of embedding is equal to four, which is also equal to the number of real-life communities. In such a case, the ideal embedding would require that all nodes in the same communities A, B, C, D have the same embedded value. At the same time, different communities should be separated from each other by exactly one unit. Consider the embedding in which all nodes in community A have the value $(1/\sqrt{2}, 0, 0, 0)$, those in B, C and D have the values $(0, 1/\sqrt{2}, 0, 0)$, $(0, 0, 1/\sqrt{2}, 0)$ and $(0, 0, 0, 1/\sqrt{2})$, respectively. While the embedding for the central red node should have equal values in all dimensions, *i.e.,* (x, x, x, x) where $x = \sqrt{2}/4$.

It can be shown in this case, by optimizing the objective function O, that (a) there will be small stresses within a community because of missing edges, (b) no stresses across communities, because the difference in embedded values is exactly one unit, and (c) it is impossible to find an embedding for the red node, without causing stresses, as it connects to four nodes coming from distinct communities. Indeed, it is this abnormal phenomenon in the embedding that helps us to detect anomalous nodes, using $AScore$.

We next explain how to detect the anomalous central red node with the above derived embedding, in which the $\overline{NB(i)}$ for a node i within a community has a dominant dimension and small values in other dimensions, *e.g.,* $(2.93, 0.10, 0.10, 0.10)$ for the node within community A that connects to the central red node, and, thus, its $AScore(i)$ is very close to 1. However, the $\overline{NB(red)}$ for the red node has almost equal values in the four dimensions, thus the $AScore(red)$ is close to 4, and is much larger than the rest of nodes, and, thus, the central red node is detected as an anomaly. □

In the above example, the number d is equal to the number of real-life communities, which is hard to obtain in practice. However, we argue that a reasonable choice of d suffices to detect anomalies for our embedding approach, as will be shown in the experimental study (Section VI).

Discussions. The objective function O is expressed as a summation of all the $O(n^2)$ possibilities for different node pairs. This can be very expensive for large networks. Further, the optimization of O typically requires a gradient descent method, which can be very expensive in practice, when the number of variables to be optimized is large. Another problem is the choice of the balancing factor α. As it turns out, these two aspects are related, since the size of the problem can be reduced, depending upon the choice of the parameter α. Finally, it is important to develop a fast solution of the optimization problem, in order to ensure an efficient solution. These issues will be addressed in the next two sections.

IV. Determining Optimal Embedding Efficiently

The major step in the modeling process discussed in the previous section lies in finding the optimal embedding value. In order to determine the optimal embedding, the objective function O discussed in the last section needs to be optimized.

The objective function O includes the use of a balancing parameter α. How should this parameter α be decided in practice? An immediate observation is that most real graphs are sparse, and therefore, if the balancing parameter α were not used, the objective function O would be dominated by the stresses on the non-edges. As a result, the impact of the edges themselves would be very limited. Therefore, the value of α should be picked less than 1. Furthermore, since it is desirable to provide equal weights to the edges and non-edges, the balancing parameter should be picked, so that these two components have approximately equal weights.

The number of edges in the network is $m << \binom{n}{2}$ for sparse networks, and the number of non-edge node pairs is $\binom{n}{2} - m$. These represent the numbers of each type of the components in the objective function O. Hence, to provide each component with approximately similar weights, the value of α should be picked as follows:

$$\alpha = m/(\binom{n}{2} - m) \tag{7}$$

This choice ensures that the contributions of both components are approximately similar. Note that the value of α will vary with the level of sparsity of the underlying graph.

While this choice of α ensures that the two components are equally weighted, it does not solve the problem that the objective function O has $O(n^2)$ terms in total. This can be extremely inefficient when $m << n^2$. It turns out that it is possible to approximately represent O using sampling.

Let U be the set of universal $\binom{n}{2}$ node pairs, which can be constructed by choosing any two distinct nodes in V. Then, for any graph $G = (V, E)$, the set of non-edges E_n can be represented by the following set:

$$E_n = U - E \tag{8}$$

An equivalent way to use a weight of α for each non-edge would be to sample a set $E_s \subseteq E_n$ of size $\alpha \cdot |E_n|$, and rewrite the *sampled* objective function O *approximately* as follows:

$$O \approx \sum_{(i,j)\in E} S(\overline{X_i}, \overline{X_j}) + \sum_{(i,j)\in E_s} S(\overline{X_i}, \overline{X_j}) \tag{9}$$

$$= \sum_{(i,j)\in E} ||\overline{X_i} - \overline{X_j}||^2 + \sum_{(i,j)\in E_s} (||\overline{X_i} - \overline{X_j}|| - 1)^2 \tag{10}$$

Note that the parameter α is missing in the above representation of the objective function O, since the set of non-edges has already been down-sampled in order to re-weight the objective function O appropriately.

The best way of optimizing O in terms of the $n \cdot d$ embedding variables is by treating it as a nonlinear programming optimization problem. Such a problem can be optimized with the use of the gradient descent method for non-linear functions. However, for exact objective functions, it has been shown that the gradient descent method is typically not scalable for large networks. Hence we adopt the approximate approach, *i.e.,* using the mini-batch gradient descent method [12] as indicated by the approximate objective function in Equation (9). As will be shown in the experimental study (Section VI), the mini-batch gradient descent method reaches a good tradeoff between the efficiency and accuracy.

Gradient descent optimization requires the determination of the *gradient vector* for O with respect to the underlying problem variables. The gradient of O is a vector, each component of which is a partial derivative of O with respect to each of the variables (the $n \cdot d$ coordinates in the embedding) in O. Thus, the gradient will be a vector of length $n \cdot d$:

$$\nabla O = \sum_{(i,j)\in E} \nabla S(\overline{X_i}, \overline{X_j}) + \sum_{(i,j)\in E_s} \nabla S(\overline{X_i}, \overline{X_j}) \tag{11}$$

Each individual term in the gradient above can be simplified. The simplification of the term is slightly different, depending on whether it corresponds to an edge or a non-edge node pair.

$$\nabla S(\overline{X_i}, \overline{X_j}) = \begin{cases} 2||\overline{X_i} - \overline{X_j}|| \cdot \nabla||\overline{X_i} - \overline{X_j}|| & \text{for}(i,j) \in E \\ 2(||\overline{X_i} - \overline{X_j}|| - 1) \cdot \nabla||\overline{X_i} - \overline{X_j}|| & \text{else} \end{cases} \tag{12}$$

Therefore, the overall gradient for the objective function O can be expressed as follows:

$$\begin{aligned} \nabla O = & \sum_{(i,j)\in E} 2||\overline{X_i} - \overline{X_j}|| \cdot \nabla||\overline{X_i} - \overline{X_j}|| + \\ & + \sum_{(i,j)\in E_s} 2(||\overline{X_i} - \overline{X_j}|| - 1) \cdot \nabla||\overline{X_i} - \overline{X_j}|| \end{aligned} \tag{13}$$

The value of $\nabla||\overline{X_i} - \overline{X_j}||$ can be evaluated to a gradient vector, by using the partial derivative with respect to each of the $n \cdot d$ embedding variables x_i^k (where $i \in \{1, \ldots, n\}$, $k \in \{1, \ldots, d\}$) of the expression:

$$||\overline{X_i} - \overline{X_j}|| = \sqrt{\sum_{k=1}^{d} (x_i^k - x_j^k)^2} \tag{14}$$

The overall approach for gradient descent starts off with an initial embedding $(\overline{X_1^0}, \ldots, \overline{X_n^0})$ and iteratively updates $(\overline{X_1^t}, \ldots, \overline{X_n^t})$ to $(\overline{X_1^{t+1}}, \ldots, \overline{X_n^{t+1}})$ using the following formula:

$$(\overline{X_1^{t+1}}, \ldots, \overline{X_n^{t+1}}) \leftarrow (\overline{X_1^t}, \ldots, \overline{X_n^t}) - \gamma_t \cdot \nabla O \tag{15}$$

Note that here each iteration uses a new sampling set in order to decrease the loss of the approximate rewrite in Equation (9), and γ_t is the step-size in the tth iteration. A variety of off-the-shelf methods such as the line-search method, or the Newton's method can be used in order to determine the step sizes for the iterations.

Incorporating graph partitions into embedding. The detection of network anomalies using the gradient method is critically dependent on determining a good initialization for the method. According to the embedding model and Fig. 3, a natural solution is to use an ensemble of graph-partitioning algorithms in order to initialize the embedding. Intuitively, the embedding of a network of d communities is constructed in such a way, that each of the d dimensions of the embedding represents a closely clustered set of nodes. Specifically, a similar value on the ith component of the embedding for a set of nodes, reflects the propensity of that set of nodes to belong to the ith community. This is illustrated from the example of Fig. 3. Therefore, a natural solution is to use an ensemble of graph-partitioning algorithms in order to initialize the embedding. Of course, graph-partitioning is itself an NP-hard problem, and the use of such a slow method for the initialization defeats the purpose of a more sophisticated approach in the first place. A key observation here is that it is not necessary for each of these algorithms to return a precisely optimized partitioning (NP-hard). Rather, a modestly good partitioning, which can be determined quickly, is enough to be used for initialization.

Initializing embedding with partitions. Consider a partitioning with d clusters produced by a graph partitioning approach. For the embedding $\overline{X_i} = (x_i^1, \ldots, x_i^d)$ of each node $i \in V$, the jth component x_i^j may be defined as follows:

$$x_i^j = \begin{cases} 1/\sqrt{2} & \text{if node } i \text{ belongs to cluster } j \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

Different from traditional graph embeddings, such as multi-dimensional scaling, this initial embedding provides a natural understanding of the underlying community structure and a high quality initialization for the gradient descent method.

METIS. The algorithms implemented in METIS are based on the multilevel graph partitioning paradigm [20], and have been shown to quickly produce high-quality partitionings. METIS can partition large irregular graphs into a user-specified number d of groups, and it has been developed at the University of Minnesota, and is freely distributed. Hence, we adopt METIS to produce a modestly good partitioning.

Remarks. It should be pointed out that our embedding tends to preserve the local linkage structure of a graph, and can therefore be used as a multidimensional representation of the graph in conjunction with any application, which is dependent on the concept of homophily, as discussed in Section I.

V. Reducing Dimensions for Better

As analyzed in Sections III and IV, our algorithm, referred to as Embed, essentially computes an optimal embedding, using the gradient descent method in Section IV, to detect anomalous nodes in a graph. To do this, each node i ($i \in [1, n]$) is associated with three d-dimensional vectors: $Current_i$, $Next_i$ and $Direction_i$, which represent the current embedding $(x_i^1, \ldots, x_i^d)$, the next embedding $(y_i^1, \ldots, y_i^d)$ and the partial derivative $(p_i^1, \ldots, p_i^d)$ of the objective function O with respect to point $Current_i$, respectively. Note that here a d-dimensional vector is essentially a point in the d-dimensional embedding space.

We first explain the detailed process in a single iteration of algorithm Embed. For each node i ($i \in [1, n]$), its $Next_i$ is computed as follows:

$$Next_i = Current_i - \gamma \cdot Direction_i \tag{17}$$

where γ is the step size. To satisfy the embedding constraint of Equation (1), vector $Next_i = (y_i^1, \ldots, y_i^d)$ is further normalized such that: (a) for each $j \in [1, d]$, y_i^j is set to 0 if

$y_i^j < 0$, and (b) $Next_i$ is scaled properly to be of length $1/\sqrt{2}$ if $||Next_i|| > 1/\sqrt{2}$. It's possible to quickly derive that the norm of the difference of any two embedding vectors is equal to or less than 1.

In each iteration, the step size γ is determined by using *backtracking (inexact) line search* [23] with the *Armijo rule* constraint [6] such that

$$O(Next) \leq O(Current) - c \cdot \gamma \cdot ||Direction||^2, \quad (18)$$

where O is the objective function, c is a constant parameter, and $Next$, $Current$, and $Direction$ are three $n{\cdot}d$-dimensional vectors defined as follows:

$$Next = (Next_1, \ldots, Next_n) \quad (19)$$

$$Current = (Current_1, \ldots, Current_n) \quad (20)$$

$$Direction = (Direction_1, \ldots, Direction_n) \quad (21)$$

The above process repeats until the change of the objective function O is equal to or less than a pre-defined threshold δ, or the iteration number exceeds a pre-defined threshold t.

Main challenges. It is easy to see that algorithm Embed takes $O(3n \cdot d)$ space for a graph with n nodes and d communities, and the objective function O involves with $n \cdot d$ variables. Indeed d can be large in practice, *e.g.,* 8,385 for YouTube and 6,288,363 for Orkut [33]. Hence, it could be computationally expensive for Embed on large graphs.

$k + \beta$ **reduction**. Observe that anomalous nodes typically connect to a limited number of communities, but not all the d communities. Furthermore, our goal is to identify anomalous nodes, and, it often suffices to ascertain anomalous nodes when they are found connected to a certain number of influential communities. In other words, there is no need to use the complete d-dimensions.

This motivates us to propose the usage of $(k + \beta)$-dimensions, instead of d-dimensions, for vectors $Current_i$, $Next_i$ and $Direction_i$ ($i \in [1, n]$). Intuitively, parameter k represents the maximum number of communities to which anomalous nodes connect, and β is a tolerance parameter. In practice, k and β are typically small, *e.g.,* 10 and 2 for a social network with more than 1 million nodes.

The $k + \beta$ reduction obviously reduces the space cost and improves the efficiency in the same time. Moreover, it further *improves the effectiveness of anomaly detection*, due to the removal of non-influential communities, *i.e.,* noises. Below we give an informal analysis.

As we analyzed in Section III, only influential communities play a role in the detection of anomalous nodes in networks. By the definition of $\overline{NB(i)}$, every node j within $NB(i)$ has a contribution to $\overline{NB(i)}$, no matter whether node j belongs to influential communities or non-influential ones. Thus, the joint embedding of non-influential communities may accumulate and finally make the $AScore(i)$ of node i large enough so that node i is mistakenly marked as an anomaly. By using the $k + \beta$ reduction, only the top-$(k + \beta)$ entries are kept in an embedding, and, hence, the above situation is alleviated to a large extent. As a result, the $k + \beta$ reduction further improves the effectiveness of anomaly detection, as will be shown in the experimental study.

Details. We next present the detailed process of the $(k + \beta)$ reduction optimization technique.

(1) For all the vectors $Current_i = (x_i^1, \ldots, x_i^d)$ and $Next_i = (y_i^1, \ldots, y_i^d)$ ($i \in [1, n]$), we only maintain the top $k + \beta$ largest values, instead of all d values, for x_i^j and y_i^j ($j \in [1, d]$). For all the vectors $Direction_i = (p_i^1, \ldots, p_i^d)$ ($i \in [1, n]$), we maintain the values of the $k + \beta$ dimensions of $Current_i$, and the top $k + \beta$ smallest values in the remaining $d - (k + \beta)$ values of p_i^j. Here we keep the small values of $Direction_i$ to derive large values of $Next_i$, as shown by Equation (17).

(2) When computing the partial derivatives $Direction_i$ ($i \in [1, n]$), we only use the top k largest values in $Current_i$ and in all the involved vectors $Current_j$ such that $S(\overline{X_i}, \overline{X_j})$ appears in the objective function O in Equation (9). We then keep $2 \cdot (k + \beta)$ values for $Direction_i$: the first $k + \beta$ values are from the same dimensions as $Current_i$, and the rest are the top $k + \beta$ smallest values from the remaining dimensions.

(3) When computing the vectors $Next_i$ ($i \in [1, n]$) using Equation (17), we first extend $Next_i$ and $Current_i$ with another $k + \beta$ dimensions with values $zero$ since $Direction_i$ has $2 \cdot (k + \beta)$ dimensions. We finally reduce $Next_i$ to $k + \beta$ dimensions, by keeping its top $k + \beta$ largest values.

(4) When determining the step size γ using Equation (18), we use the top k largest values of $Current_i$ and $Next_i$, and all the $2 \cdot (k + \beta)$ values of $Direction_i$ ($i \in [1, n]$), respectively.

With the $k + \beta$ reduction, the space cost and the number of variables in the objective function of Embed are reduced to $O(4 \cdot n \cdot (k + \beta))$ and $n \cdot k$, respectively. We also denote the algorithm Embed without and with $k + \beta$ reduction as Embed(d) and Embed$(k + \beta)$, respectively. Here (a) the parameter β provides the opportunity to tolerate mistakes for determining the k communities to which a node belongs. (b) As analyzed above, $k + \beta$ reduction reduces the space and time costs in the same time. (c) We will also experimentally verify these, and, better still, show that this technique further improves the effectiveness of anomaly detection due to the removal of noises in the experimental study (Section VI).

Remarks. By now we have introduced the entire framework of our embedding approach to network anomaly detection. We conclude the framework from the following aspects.

(1) We give an analysis on the structurally inconsistent anomalies in networks, both from the perspective of application-specific significance and the performance of network analyses.

(2) We propose a new embedding technique that aims to preserve local linkage structure and community structure of networks, and cast such a problem into an optimization problem. In the embedding model, each dimension represents a clustered region of networks, and graph partitions are incorporated to obtain a high quality initial embedding.

(3) The $AScore$ metric is well established, which seamlessly associates structural inconsistencies with the embedding, and makes use of a single parameter $thre$ to distinguish the anomalies from normal nodes.

(4) We present two algorithms, *i.e.,* Embed(d) and Embed$(k + \beta)$, under the framework. The space and time complexities are $O(3n \cdot d)$ and $O(t \cdot m \cdot d)$, respectively, for Embed(d), while reducing to $O(4n \cdot (k + \beta))$ and $O(t \cdot m \cdot (k + \beta)))$ for Embed$(k + \beta)$. By the $k + \beta$ reduction, our algorithm can scale

up to large networks, and, better still, it further improves the effectiveness because of the explicit removal of noises.

VI. Experimental Study

In this section, we present an extensive experimental study of our embedding approach to detecting network anomalies. Using both real-life and synthetic data, we conducted four sets of experiments to evaluate: (1) the application of anomalies found by our embedding approach with a case study, (2) the improvement of community detection by evaluating modularity after removing anomalies found by our algorithms Embed(d) and Embed($k+\beta$), (3) the quality of anomalies by evaluating the F_1 measure of anomalies found by Embed(d) and Embed($k+\beta$), and (4) the efficiency of our algorithms Embed(d) and Embed($k+\beta$) for detecting anomalies.

A. Experimental Settings

We first introduce the settings of our experimental study.

Datasets. We chose three datasets to test our approach.

(1) Amazon records a product co-purchasing network with 334,863 product nodes and 925,872 product-product edges[1]. It is based on the *Customers Who Bought This Item Also Bought* feature. That is, an edge from product x to y indicates that if people buy x, then they will also buy y with a very high probability. The graph has a single connected component.

(2) Dblp contains scientific publication information in the computer science domain from years 1936 to 2014[2]. We further processed the dataset to compose a co-author graph from it, in which each node is an author, and an edge between two nodes i and j indicates that authors i and j are co-authors in at least one publication. The generated complete co-author graph consists of 115,305 connected components. The largest one has 1,150,852 nodes while the second largest one has only 32 nodes. Hence we choose the largest connected component with 1,150,852 nodes and 5,098,175 edges as the Dblp dataset.

(3) Synthetic graphs with community structures vary from 10^5 to 4×10^6 nodes, and are produced according to the LFR-benchmark graph [21], which includes heterogeneous distributions of node degree and community size, and provides a more severe test of community detection methods.

The implementation[3] is controlled by five key parameters: (a) the number n of nodes; (b) the average degree $avgD$ of nodes to generate n^λ edges; (c) the mixing parameter μ such that each node shares $1-\mu$ of its links with the other nodes in its community; (d) the exponents t_1 and t_2 that correspond to the power laws of the degree and community size distributions, respectively; (e) the maximum degree $maxD$, given by $n^{1/(t_1-1)}$. To produce a network of size $n\cdot(1+p)$, where $p\cdot n$ *nodes are anomalies*, we first produced a network with $n+r$ nodes, in which r nodes were further processed to generate $(n\cdot p)/2$ anomalies using an agglomerative operation as follows: A group of q nodes was replaced by a single new node such that all edges connecting to these q nodes in the network were re-connected to the new node. The degrees of q nodes are less than or equal to $2\cdot avgD$, such that these new nodes are tending to be anomalies as they connect to diverse influential communities. Here the parameter q varies from 2 to 21 with an increment of 1, thus the parameter r is $(23\cdot n\cdot p)/4$. Besides these agglomerated anomalies, we further injected another $(n\cdot p)/2$ nodes, each of which connects to nodes of original network randomly, and their degrees fall in $[avgD, maxD]$ and satisfy the power law of exponent t_1.

Table I. Summary of main parameter settings

Parameter	Description	Default
d	number of communities or clusters	$n/500$
k	community number in the $k+\beta$ reduction	$avgD$
β	tolerance parameter in the $k+\beta$ reduction	$\lfloor k/4 \rfloor$
θ	threshold for influential communities	0.1
$thre$	threshold of detecting anomalies	$--$
c	parameter in Equation (18)	0.04
δ	change threshold in gradient descent	0.001
t	iteration threshold in gradient descent	50
p	proportion of anomalies in Synthetic	1.0%
λ	degree parameter in Synthetic	1.15
μ	mixing parameter in Synthetic	0.4
t_1	negative exponent of power law in Synthetic	3
t_2	negative exponent of power law in Synthetic	2
q	number of nodes to be merged in Synthetic	$[2, 21]$
r	number of nodes to generate anomalies in Synthetic	$--$

Algorithms and implementation. We compared our embedding approach with BET [9], MDS(d) [8] and OddBall [5].

(1) Algorithm BET is the approximate solution in [34] to detect the nodes with the highest betweenness centrality as anomalies, whose number of hyperedges was fixed to $16,000$.

(2) Algorithm MDS(d) is similar to our Embed(d) except using *multi-dimensional scaling* [8] for obtaining the embedding, which aims to preserve global pairwise similarities of graphs. We exploited Landmark MDS [13] to efficiently compute embedding for large graphs. For a target space of dimensionality d, we randomly selected $2\cdot d$ landmark points. The computation of embedding $\overline{NB(i)}$ in Equation (5) is modified as follows:

$$\overline{NB(i)} = (y_i^1, \ldots, y_i^d) = \sum_{j\in NB(i)} \exp\{1 - ||\overline{X_i} - \overline{X_j}||\}\cdot \overline{X_j} \qquad (22)$$

Equation (22) remains the intuition that neighbors closer to node i have a higher weight when $||\overline{X_i} - \overline{X_j}||$ is not guaranteed to be equal to or less than 1 anymore.

(3) Algorithm OddBall discovers several power law patterns governing the ego-nets of all nodes, *i.e.,* subgraphs of nodes and their neighbors, and uses them for anomaly detection [5]. More specifically, OddBall first computes a Least Squares fitting line for a power law, and measures the anomalousness of each node according to its distance to the fitting line. We adopted the *Egonet Density Power Law*, which defines the relation between the number of nodes $|N_i|$ and the number of edges $|E_i|$ of the ego-net G_i:

$$|E_i| \propto |N_i|^\phi, 1 \le \phi \le 2 \qquad (23)$$

And the fitting line is in the form of $|E_i| = C\cdot|N_i|^\phi$, where values of $|E_i|$ of structural inconsistencies are typically less than the expected values of $C\cdot|N_i|^\phi$ as the neighbors of structural inconsistencies are from a number of diverse communities. As social networks are sparse, we also extended $|E_i|$ to the number of edges in 2 hops, and the intermediate nodes of 2-hop paths were not limited in the node set of G_i.

(4) All algorithms were implemented with Microsoft Visual C++. We adopted the Newman algorithm in the igraph library (http://igraph.sourceforge.net/) for computing communities and modularity [11]. We also implemented an algorithm for

[1]http://snap.stanford.edu/data/index.html
[2]http://www.informatik.uni-trier.de/~ley/db/
[3]http://homes.soic.indiana.edu/filiradi/Resources.html/

Table II. $\mathsf{Embed}(d)$ vs. $\mathsf{MDS}(d)$ using modularity and F_1 measure

	Modularity						**F_1 measure**	
Dataset	AMAZON (87.6%)		DBLP (56.8%)		SYNTHETIC (47.1%)		SYNTHETIC	
Algorithm	$\mathsf{MDS}(d)$	$\mathsf{Embed}(d)$	$\mathsf{MDS}(d)$	$\mathsf{Embed}(d)$	$\mathsf{MDS}(d)$	$\mathsf{Embed}(d)$	$\mathsf{MDS}(d)$	$\mathsf{Embed}(d)$
$\boldsymbol{d = 200}$	94.6%	**95.1%**	62.4%	**62.6%**	47.9%	**53.4%**	11.3%	**89.4%**
$\boldsymbol{d = 400}$	94.9%	**95.6%**	**61.5%**	61.0%	47.5%	**53.8%**	13.6%	**90.6%**
$\boldsymbol{d = 600}$	94.9%	**95.8%**	62.4%	**62.5%**	47.6%	**54.3%**	12.7%	**89.8%**
$\boldsymbol{d = 800}$	94.9%	**96.0%**	60.9%	**61.8%**	47.4%	**53.9%**	11.2%	**88.9%**
$\boldsymbol{d = 1000}$	95.2%	**96.2%**	59.6%	**62.1%**	47.3%	**54.0%**	7.9%	**85.5%**
Avgerage	94.9%	**95.7%**	61.4%	**62.0%**	47.5%	**53.9%**	11.3%	**88.8%**

generating subgraphs of AMAZON, DBLP and SYNTHETIC: we started with a node with high degree, and then produced a graph with n nodes using the breadth-first search strategy.

All experiments were run on a PC with an Intel Core i5-2400 CPU @3.10GHz and 16GB of memory. The usage of virtual memory was forbidden in all our tests. In cases when quantitative measurements are reported, the test was repeated over 3 times and the average is reported here.

B. Experimental Results

We tested the effectiveness and efficiency of our embedding approach using AMAZON, DBLP and SYNTHETIC datasets. Since the number d of communities typically increases along with the increase of graph sizes, we simply fix d to $n/500$ by default in our tests, where 500 is the average community size in our tested networks. As for the number k in $k+\beta$ reduction, we find that a good selection for k is to use the average node degree of networks, a bound of the average number of communities to which a node connects. All parameters and their default settings are summarized in Table I.

We next present our findings.

(I) Case Study. In the first set of experiments, *as an application of node anomaly detection* we show three interesting examples found in DBLP since it is the only available one with such details in our tested datasets.

Exp-1.1. Different people with the same name. For instance, *Wei Wang* and *Wei Li* were detected anomalies. There are 51 people with name "Wei Wang" and 43 people with name "Wei Li" in DBLP, which are unanimously treated as individual persons, and are connected to many communities of DBLP.

Exp-1.2. People with many collaborators in diverse institutes. For instance, *Ajith Abraham* and *Vincent Poor* were detected as anomalies. These people only belong to a very small number of community groups, due to the simple strategy used to identify communities by DBLP. (a) Dr. Ajith Abraham is the current director of machine intelligence research labs, which has members from more than 100 countries. He works in a multi-disciplinary environment involving machine (network) intelligence, cyber security, sensor networks, data mining and applied to various real world problems, and he has been associated with the teaching and research with 23 universities all around the world. He has 502 co-authors in DBLP. (b) Dr. H. Vincent Poor is the George Van Ness Lothrop Professor at Princeton University, and he is widely recognized as one of the world's leading educators and researchers in wireless communications, signal processing and related fields. He has 548 co-authors in DBLP.

These two researchers have a lot of collaborators who work in different institutes and form many different community groups, and hence they were detected as anomalies.

Exp-1.3. Undetected different people with the same name. For instance, *Jian Li* was detected as an anomaly. We did a careful check and found that there were quite a few people with name "Jian Li", while DBLP mistakenly treats them as the same person. For instance, there are distinct people from IBM Austin Research Laboratory, University of Florida, Tsinghua University, Dalian University of Technology, Chinese Academy of Sciences, Hubei University of Education, Southwest Petroleum University, Concordia University, Beihang University, Harbin Institute of Technology etc.. As shown in DBLP, these people are treated as the same person, and belong to 24 different community groups.

(II) Effectiveness Study. For the effectiveness study, we (1) compared our embedding approach with betweenness centrality, (2) used the modularity to evaluate the improvement of the effectiveness of community detection, and (3) adopted the F_1 measure to evaluate the quality of anomalous nodes found.

(1) Modularity. It is the fraction of the edges that fall within the given groups minus the expected such fraction if edges were distributed at random. It is designed to measure the strength of division of a network into communities, and is often used in community detection optimizations. The rational behind is the removal of anomalies results in better communities.

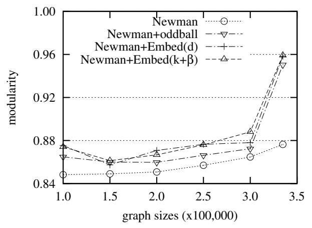
(2) F_1 measure. It is a measure of a test's accuracy, and we used it to further measure the quality of anomalous nodes found. As this needs the ground truth of anomalies, which is hard to obtain in real life networks, we used the anomaly injection technique on the SYNTHETIC dataset [28], [30].

Here we chose small datasets to test modularity and F_1, as $\mathsf{Embed}(d)$ did not scale to large datasets, and parameter $thre$ for AMAZON, DBLP and SYNTHETIC was empirically fixed to 2.0, 3.0 and 3.3, respectively, by default.

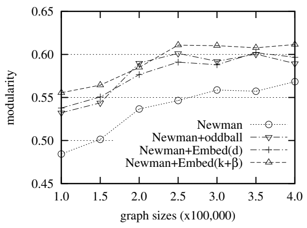
Exp-2: Structural inconsistency vs. betweenness centrality. To clarify these two different notions, we first detected structural inconsistencies using $\mathsf{Embed}(d)$, and then computed the same number of nodes with the highest betweenness centrality using BET on AMAZON, DBLP and SYNTHETIC. The total number of nodes was fixed to $400K$ for DBLP and SYNTHETIC, and the entire AMAZON, respectively.

The fraction of common nodes that were both detected by $\mathsf{Embed}(d)$ and BET was 18.2% on AMAZON, 16.4% on DBLP, and 37.1% on SYNTHETIC, respectively. Moreover, the average degrees of nodes detected by ($\mathsf{Embed}(d)$, BET) were (5.3, 13.6) on AMAZON, (22.3, 62.6) on DBLP, and (94.7, 176.1) on SYNTHETIC, respectively. Finally, the F_1 scores of nodes found by ($\mathsf{Embed}(d)$, BET) were (88.9%, 39.8%) on SYNTHETIC. These together show that structural inconsistency is significantly different from betweenness centrality.

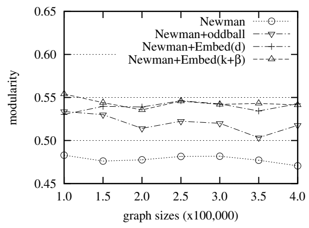
Exp-3: $\mathsf{Embed}(d)$ **vs.** $\mathsf{MDS}(d)$. Our embedding method is specifically designed to ferret out the anomalous nodes in large



(a) AMAZON dataset

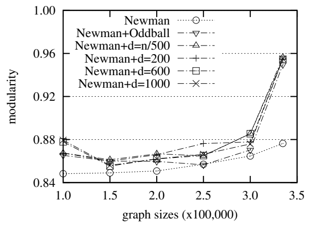


(b) DBLP dataset

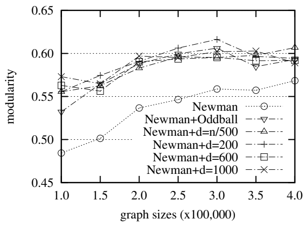


(c) SYNTHETIC dataset

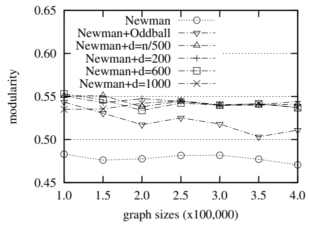
Figure 4. Impacts on the effectiveness of community detection (modularity): $k + \beta$ reduction



(a) AMAZON dataset

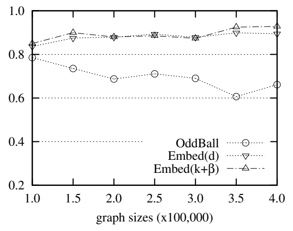


(b) DBLP dataset

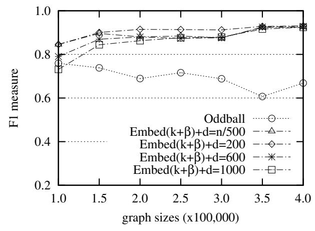


(c) SYNTHETIC dataset

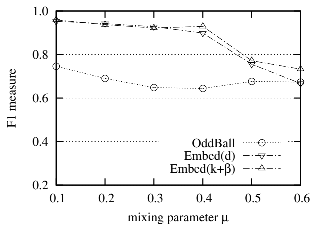
Figure 5. Impacts on the effectiveness of community detection (modularity): the number d of dimensions



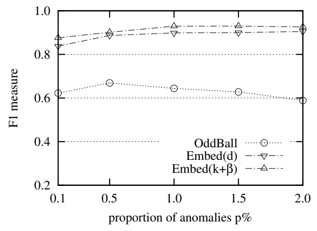
(a) $k + \beta$ reduction



(b) The number d of dimensions



(c) The mixing parameter μ



(d) The proportion p of anomalies

Figure 6. Impacts on the quality (F_1 measure): SYNTHETIC dataset

networks. To show this, we compared Embed(d) with MDS(d) using both modularity and F_1 measure.

Exp-3.1. We first tested the modularity of the community structure using algorithm Newman directly, and then tested the modularity after removing node anomalies found by Embed(d) and MDS(d), respectively. For the sake of fairness, we simply fixed the number of anomalies removed by MDS(d) the same as Embed(d), and varied d from 200 to 1000, while fixed the number of graph nodes, *i.e.,* graph size, to $400K$ for DBLP and SYNTHETIC, and the largest size $334.8K$ for AMAZON.

The results are reported in Table II. In all cases, our embedding method Embed(d) and MDS(d) consistently improve the modularity, compared with using Newman directly (results are in the second row of Table II). The improvements of Embed(d) with respect to d are also better than MDS(d) for almost all cases. In deed, the modularity was increased about (7.3%, 8.1%) for AMAZON, (4.6%, 5.2%) for DBLP and (0.4%, 6.8%) for SYNTHETIC on average, after the anomalies were removed by MDS(d) and Embed(d), respectively, in our tests.

Exp-3.2. We then tested the F_1 score of node anomalies detected by Embed(d) and MDS(d), respectively. Similar to using modularity, we varied d from 200 to 1000 while fixed graph size and number of anomalies.

The results are reported in Table II. In all cases, the F_1 scores of Embed(d) with respect to d were better than MDS(d). In deed, the F_1 scores of MDS(d) and Embed(d) were (11.3%, 88.8%) on average, respectively, in our tests.

These experimental results show that the direct use of MDS for the detection of structural inconsistencies is not appropriate, and our embedding approach Embed(d) clearly outperforms MDS(d), especially for the F_1 scores. Thus, we did not report the rest effectiveness comparisons with MDS(d).

Exp-4: Modularity evaluation. In the fourth set of experiments, we used *modularity* to evaluate the improvement of the effectiveness of community detection.

Exp-4.1. To evaluate the impacts of the $k + \beta$ reduction technique, we first tested the modularity of the community structure using algorithm Newman directly, and then tested the modularity after removing node anomalies detected by OddBall, Embed(d) and Embed($k + \beta$), respectively. Since Embed(d) and Embed($k+\beta$) find different numbers of anomalies, for the sake of fairness, we fixed the number of anomalies to be the small one. We varied the graph sizes, from $100K$ to $400K$ or the largest size $334.8K$ for AMAZON, while fixed d = $n/500$. The results are reported in Fig. 4.

When varying the graph sizes, the modularity using Embed($k + \beta$) was consistently close to the one using Embed(d) for AMAZON, DBLP and SYNTHETIC data. Their modularity differences were only 0.34% for AMAZON, 1.5% for DBLP and 0.61% for SYNTHETIC, respectively, in our tests. In most cases, the modularity using Embed($k + \beta$) was even better than the one using Embed(d), which may be caused by the removal of noises of embedding. In all cases, both our embedding methods Embed(d) and Embed($k+\beta$) consistently improve the modularity, compared with using Newman directly. The improvements of Embed(d) and Embed($k + \beta$) were also better than OddBall in most cases. In deed, the modularity was increased about (2.1%, 2.8%, 3.0%) for AMAZON, (4.2%, 4.1%, 5.6%) for DBLP, and (4.2%, 6.1%, 6.5%) for

SYNTHETIC on average, after the anomalies were removed by OddBall, Embed(d) and Embed$(k+\beta)$, respectively.

Exp-4.2. To evaluate the impacts of the dimension numbers, we first tested the modularity of the community structure using algorithm Newman directly, and then tested the modularity after removing node anomalies found by OddBall, Embed$(k+\beta)$ with d = $n/500$, 200, 600 and 1000, respectively. When varying d, Embed$(k+\beta)$ would also detect different counts of anomalies, although embedding approach and $thre$ were fixed the same. Similar to the setting of *Exp-4.1*, we varied the graph sizes, while fixed Embed$(k+\beta)$ and used the minimum number of anomalies with different d.

The results are reported in Fig. 5. When varying the graph sizes, the modularities using Embed$(k+\beta)$ with different numbers d of dimensions were consistently close to each other for AMAZON, DBLP and SYNTHETIC data. Their modularity differences were only 0.76% for AMAZON, 1.6% for DBLP and 0.85% for SYNTHETIC on average, respectively, in our tests. In all cases, our embedding method Embed$(k+\beta)$ consistently improves the modularity, compared with using Newman directly. The improvements of Embed$(k+\beta)$ with different d are also better than OddBall in most cases. In deed, the modularity was increased about (1.9%, 2.6%, 2.6%, 2.6%, 2.5%) for AMAZON, (4.5%, 4.9%, 5.4%, 4.7%, 5.3%) for DBLP and (4.3%, 6.5%, 6.6%, 6.4%, 6.1%) for SYNTHETIC on average, after the anomalies were removed by OddBall and Embed$(k+\beta)$ with $d = n/500$, 200, 600, 1000, respectively.

Exp-5: F_1 measure evaluation. In the fifth set of experiments, we used the F_1 measure to further measure the quality of anomalous nodes found. We compared the F_1 measure of our embedding approach with OddBall [5].

We chose small datasets to test the F_1 measure again, as Embed(d) did not scale to large ones. We further tested the mixing parameter μ and the proportion of anomalies p, in addition to the two algorithm factors of *Exp-4*.

Exp-5.1. To evaluate the impacts of the $k+\beta$ reduction technique, we varied the graph sizes from $100K$ to $400K$, while fixed $d = n/500$, $\mu = 0.4$, $p = 1.0\%$ and the number of anomalies removed as the smaller quantity of anomalies detected by Embed(d) and Embed$(k+\beta)$.

The results are reported in Fig. 6(a). The F_1 scores of OddBall, Embed(d) and Embed$(k+\beta)$ were (70%, 88%, 89%) on average for SYNTHETIC, respectively. When varying the graph sizes, the F_1 score using Embed$(k+\beta)$ is consistently close to the one using Embed(d). Their F_1 score difference was only 1.6% on average in our tests. In all cases, the F_1 scores for both our embedding methods Embed(d) and Embed$(k+\beta)$ were better than OddBall. In deed, the F_1 scores of Embed(d) and Embed$(k+\beta)$ were (18%, 19%) larger than the F_1 score of OddBall on average, respectively.

Exp-5.2. To evaluate the impacts of the dimension numbers, we varied d with $n/500$, 200, 600 and 1000, respectively. Similar to the setting of *Exp-5.1*, we varied the graph sizes, while fixed Embed$(k+\beta)$, $\mu = 0.4$, $p = 1.0\%$ and the number of anomalies removed as the minimum quantity of anomalies detected by Embed$(k+\beta)$ with different dimension numbers. The results are reported in Fig. 6(b).

The F_1 scores of OddBall and Embed$(k+\beta)$ with d = $n/500$, 200, 600 and 1000 were (70%, 89%, 91%, 88%, 86%) on average for SYNTHETIC, respectively. When varying the graph sizes, the F_1 scores using Embed$(k+\beta)$ with different numbers d of dimensions were consistently close to each other. Their F_1 score difference was only 4.5% on average in our tests. In almost all cases, our method Embed$(k+\beta)$ is consistently better than OddBall. Indeed, the F_1 scores of Embed$(k+\beta)$ with d = $n/500$, 200, 600 and 1000 were (19%, 21%, 18%, 16%) larger than OddBall on average, respectively.

Exp-5.3. To evaluate the impacts of the mixing parameter, we varied μ from 0.1 to 0.6, while fixed the graph size to $400K$, $d = n/500$, $p = 1.0\%$ and the number of anomalies removed as the smaller one detected by Embed(d) and Embed$(k+\beta)$. Since $thre$ is affected by μ, we fixed $thre$ to 1.8, 2.5, 2.5, 3.3, 4.5, 6.1 when varying μ from 0.1 to 0.6, respectively. Note that when μ is closer to 1, the generated graphs are closer to random graphs, and $\mu = 0.6$ is a higher value for real-life networks. The results are reported in Fig. 6(c).

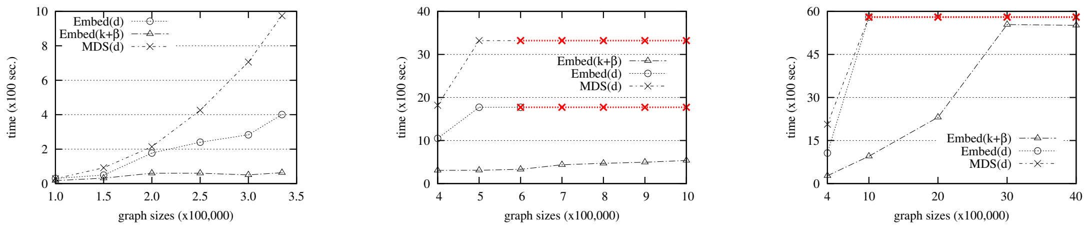
The F_1 scores of OddBall, Embed(d) and Embed$(k+\beta)$ were (68%, 86%, 88%) on average for SYNTHETIC, respectively. When varying the mix parameter, the F_1 score using Embed$(k+\beta)$ was consistently close to the one using Embed(d) for SYNTHETIC. Their F_1 score difference was only 2.2% on average in our tests. In almost all cases, the F_1 scores for both our embedding methods Embed(d) and Embed$(k+\beta)$ were better than OddBall. In deed, the F_1 scores of Embed(d) and Embed$(k+\beta)$ were (18%, 20%) larger than OddBall on average, respectively.

Exp-5.4. To evaluate the impacts of the proportion of anomalies, we varied the proportion from 0.1% to 2.0%, while fixed graph size to $400K$, $d = n/500$, $\mu = 0.4$ and the number of anomalies removed as the smaller quantity of anomalies detected by Embed(d) and Embed$(k+\beta)$.

The results are reported in Fig. 6(d). The F_1 scores of OddBall, Embed(d) and Embed$(k+\beta)$ were (63%, 89%, 91%) on average for SYNTHETIC, respectively. When varying the proportion, the F_1 score using Embed$(k+\beta)$ was consistently close to the one using Embed(d) for SYNTHETIC. Their F_1 score difference was only 2.7% on average in our tests. In all cases, the F_1 scores for both our embedding methods Embed(d) and Embed$(k+\beta)$ were better than OddBall. In deed, the F_1 scores of Embed(d) and Embed$(k+\beta)$ were (26%, 28%) larger than OddBall on average, respectively.

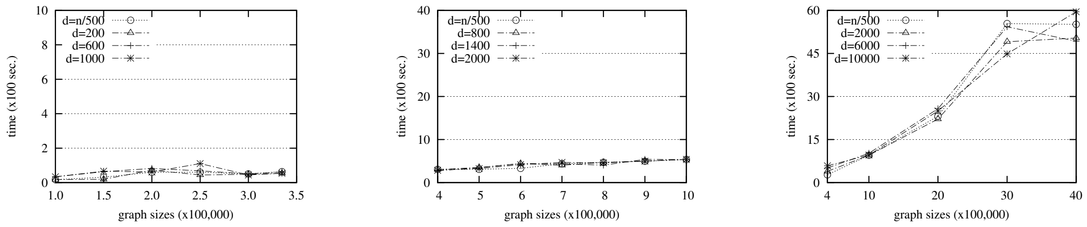
(III) Efficiency Study. In the last set of tests, we evaluated the efficiency of our algorithms. Here we chose larger datasets to test the efficiency, and evaluated the impacts of three factors that affect the efficiency of our embedding approach: the $k+\beta$ reduction, number of dimensions and number k in $k+\beta$ reduction. We produced another SYNTHETIC dataset with size $4,000K$, by setting the other parameters with default values, and generated subgraphs of different sizes with the same algorithm as the one on AMAZON and DBLP. Here we plotted with (red) markers $\times$ in the figures when algorithms Embed(d) and MDS(d) threw out a memory allocation exception.

Exp-6.1. To evaluate the impacts of the $k+\beta$ reduction technique, we tested the efficiency of MDS(d), Embed(d) and Embed$(k+\beta)$, respectively. We varied the number of graph nodes, *i.e.,* graph sizes, from $400K$ to $1,000K$ for DBLP, from $400K$ to $4,000K$ for SYNTHETIC and from $100K$ to



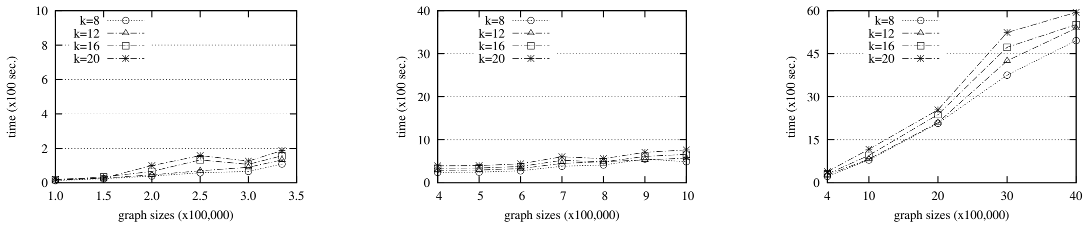
(a) Amazon dataset (b) Dblp dataset (c) Synthetic dataset

Figure 7. Impacts on the efficiency: $k + \beta$ reduction



(a) Amazon dataset (b) Dblp dataset (c) Synthetic dataset

Figure 8. Impacts on the efficiency of Embed$(k + \beta)$: the number d of dimensions



(a) Amazon dataset (b) Dblp dataset (c) Synthetic dataset

Figure 9. Impacts on the efficiency of Embed$(k + \beta)$: the number k

the largest size for Amazon, respectively, while fixed $d = n/500$ and $k = avgD$. The results are reported in Fig. 7.

When varying the graph sizes, the running time of MDS(d), Embed(d) and Embed$(k + \beta)$ increases with the increase of graph sizes. However, the running time of Embed(d) and Embed$(k + \beta)$ was consistently smaller than MDS(d). Moreover, the running time of Embed$(k + \beta)$ was also consistently smaller than Embed(d). Indeed, the running time of Embed$(k + \beta)$ was only $(35.3\%, 25.0\%)$, $(23.4\%, 13.1\%)$ and $(25.6\%, 13.2\%)$ of Embed(d) and MDS(d) on Amazon, Dblp and Synthetic on average, respectively, in our tests. The running time of $4,000K$ on Synthetic was much less than expected, due to the gradient descent finished in advance when the change of objective function O was less than the change threshold δ. Moreover, when the graph sizes were no less than $600K$ on Dblp and Synthetic, both MDS(d) and Embed(d) ran out of memory, and could not finish the tests.

Exp-6.2. To evaluate the impacts of the number of dimensions, we tested the efficiency of Embed$(k+\beta)$ with d = $n/500$ and (200, 600, 1000), (800, 1400, 2000), (2000, 6000, 10000) for Amazon, Dblp, Synthetic, respectively. Here we chose Embed$(k+\beta)$ instead of Embed(d), for larger graphs on Dblp and Synthetic. Similar to the setting of *Exp-6.1*, we varied the graph sizes, while fixed Embed$(k + \beta)$ and $k = avgD$. The results are reported in Fig. 8.

When varying the graph sizes, the running time of Embed$(k+\beta)$ with different dimensions increases with the increase of graph sizes, as expected. Moreover, the running time of Embed$(k + \beta)$ with different dimensions was consistently close to each other. The running time of Embed$(k + \beta)$ with different selections of d differed 54.5%, 14.2% and 24.5% on Amazon, Dblp and Synthetic on average, respectively, in our tests. Although the proportion result on Amazon is large, the running time difference was only 28 seconds on average. The efficiency is indeed mainly affected by the convergence speed of the gradient descent.

Exp-6.3. To evaluate the impacts of the number k in $k + \beta$ reduction, we tested the efficiency of Embed$(k + \beta)$ with k = 8, 12, 16, 20. Similar to the setting of *Exp-6.1*, we varied the graph sizes, while fixed $d = n/500$.

The results are reported in Fig. 9. When varying the graph sizes, the running time of Embed$(k + \beta)$ with different k increases with the increase of graph sizes, as expected. Moreover, its running time also decreases with the decrease of k. The running time of Embed$(k+\beta)$ with k = (16, 12, 8) was (87.7%, 66.8%, 55.7%), (86.3%, 76.3%, 66.5%) and (85.2%, 80.9%, 72.2%) of the one of Embed$(k + \beta)$ with $k = 20$ on Amazon, Dblp and Synthetic on average.

Summary. From these tests we find the followings.

(1) Our embedding approach to network anomaly detection is both effective and efficient.

The effectiveness evaluation using modularity for the improvement of community detection with algorithm Newman shows that the modularity was increased about 2.9% and 2.4% for Amazon, 4.9% and 4.2% for Dblp and 6.3% and 4.2% for Synthetic, by our approach and OddBall respectively.

The quality evaluation using the F_1 measure also shows that it was about 88% and 70% for Synthetic by our approach and OddBall, respectively.

The efficiency evaluation shows the running time of $\mathsf{Embed}(k+\beta)$ increases reasonably with the increase of graph sizes, even with large number of communities.

(2) The direct use of multi-dimensional scaling fails to effectively detect structural inconsistencies. The modularity was increased about 7.3% and 8.1% for AMAZON, 4.6% and 5.2% for DBLP, and 0.4% and 6.8% for SYNTHETIC, by $\mathsf{MDS}(d)$ and $\mathsf{Embed}(d)$ respectively. Moreover, the F_1 measure was only 11.3% for $\mathsf{MDS}(d)$, while it was 88.8% for $\mathsf{Embed}(d)$. Further, it ran out of memory for larger networks.

(3) Our $k + \beta$ reduction optimization technique reduces both space and time costs, and slightly improves the quality. The running time of $\mathsf{Embed}(k + \beta)$ was only 35.3%, 23.4% and 25.6% of its counterpart $\mathsf{Embed}(d)$ on AMAZON, DBLP and SYNTHETIC, respectively. Moreover, $\mathsf{Embed}(d)$ already ran out of memory on DBLP and SYNTHETIC with sizes equal to or larger than $600K$.

(4) While the numbers d and k of dimensions do have certain impacts on the quality and efficiency, their settings are relatively easy as long as they fall into a reasonable range, *e.g.,* $n/500$ for d and $avgD$ for k, compared with the real number of community groups that is often hard to determine.

VII. CONCLUSIONS

In this paper, we presented an embedding approach to detecting structurally inconsistent nodes (anomalous nodes) in massive social networks. The embedding approach is based on a model, in which each dimension of the embedding corresponds to a clustered region in the network, and the embedding retains a very high level of interpretability in terms of the original graph data. We utilize the gradient descent method to compute an embedding, and propose optimization techniques to make the approach more scalable for larger networks. Our extensive experimental results have demonstrated the effectiveness and efficiency of our approach to network anomaly detection, which also brings significant applications in social networks, such as the specific use of detected anomalies and the improvement of community detection. We are also exploring other possible applications of our approach.

ACKNOWLEDGMENT

This work is supported in part by 973 program (No. 2014CB340300), NSFC (No. 61322207&61421003) and MSRA Collaborative Research Program. For any correspondence, please refer to Shuai Ma.

REFERENCES

[1] C. C. Aggarwal. *Outlier Analysis*. Springer, 2013.

[2] C. C. Aggarwal, Y. Xie, and P. S. Yu. Towards community detection in locally heterogeneous networks. In *SDM*, 2011.

[3] C. C. Aggarwal, Y. Zhao, and P. S. Yu. Outlier detection in graph streams. In *ICDE*, 2011.

[4] A. Agovic, A. Banerjee, A. R. Ganguly, and V. Protopopescu. Anomaly detection using manifold embedding and its applications in transportation corridors. *Intell. Data Anal.*, 13(3):435–455, 2009.

[5] L. Akoglu, H. Tong, and D. Koutra. Graph based anomaly detection and description: a survey. *Data Min. Knowl. Discov.*, 29(3):626–688, 2015.

[6] L. Armijo. Minimization of functions having lipschitz continuous first partial derivatives. *Pacific J. Math*, 16(1):1–3, 1966.

[7] P. Bogdanov, C. Faloutsos, M. Mongiovì, E. E. Papalexakis, R. Ranca, and A. K. Singh. Netspot: Spotting significant anomalous regions on dynamic networks. In *SDM*, 2013.

[8] I. Borg and P. Groenen. *Modern Multidimensional Scaling: Theory and Applications (2nd ed.)*. Springer, 2005.

[9] U. Brandes. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 25(2):163–177, 2001.

[10] R. S. Burt. Structural holes and good ideas. *American Journal of Sociology*, 110(2):349–399, 2004.

[11] A. Clauset, M. E. J. Newman, and C. Moore. Finding community structure in very large networks. *Physical Review E*, 70:066111, 2004.

[12] A. Cotter, O. Shamir, N. Srebro, and K. Sridharan. Better mini-batch algorithms via accelerated gradient methods. In *NIPS*, 2011.

[13] V. de Silva and J. B. Tenenbaum. Global versus local methods in nonlinear dimensionality reduction. In *NIPS*, 2002.

[14] C. Faloutsos and K. Lin. Fastmap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In *SIGMOD*, 1995.

[15] J. Gao, W. Fan, D. S. Turaga, S. Parthasarathy, and J. Han. A spectral framework for detecting inconsistency across multi-source object relationships. In *ICDM*, 2011.

[16] 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*, 2010.

[17] M. Gupta, J. Gao, Y. Sun, and J. Han. Integrating community matching and outlier detection for mining evolutionary community outliers. In *KDD*, 2012.

[18] G. E. Hinton and S. T. Roweis. Stochastic neighbor embedding. In *NIPS*, 2002.

[19] J. Huang, H. Sun, J. Han, H. Deng, Y. Sun, and Y. Liu. SHRINK: a structural clustering algorithm for detecting hierarchical communities in networks. In *CIKM*, 2010.

[20] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput.*, 20(1):359–392, 1998.

[21] A. Lancichinetti, S. Fortunato, and F. Radicchi. Benchmark graphs for testing community detection algorithms. *Phys. Rev. E*, 78(4):046110, 2008.

[22] T. Lou and J. Tang. Mining structural hole spanners through information diffusion in social networks. In *WWW*, 2013.

[23] J. Nocedal and S. Wright. *Numerical Optimization*. Springer, 2000.

[24] S. Ranshous, S. Shen, D. Koutra, S. Harenberg, C. Faloutsos, and N. F. Samatova. Anomaly detection in dynamic networks: a survey. *Wiley Interdisciplinary Reviews: Computational Statistics*, 7(3):223–247, 2015.

[25] S. Rayana and L. Akoglu. Less is more: Building selective anomaly ensembles with application to event detection in temporal graphs. In *SDM*, 2015.

[26] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *SCIENCE*, 290(22):2323–2326, 2000.

[27] K. Sricharan and K. Das. Localizing anomalous changes in time-evolving graphs. In *SIGMOD*, 2014.

[28] J. Sun, H. Qu, D. Chakrabarti, and C. Faloutsos. Neighborhood formation and anomaly detection in bipartite graphs. In *ICDM*, 2005.

[29] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[30] H. Tong and C. Lin. Non-negative residual matrix factorization with application to graph anomaly detection. In *SDM*, 2011.

[31] U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[32] S. Yan, D. Xu, B. Zhang, H. Zhang, Q. Yang, and S. Lin. Graph embedding and extensions: A general framework for dimensionality reduction. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(1):40–51, 2007.

[33] J. Yang and J. Leskovec. Defining and evaluating network communities based on ground-truth. In *ICDM*, 2012.

[34] Y. Yoshida. Almost linear-time algorithms for adaptive betweenness centrality using hypergraph sketches. In *KDD*, 2014.