Neighborhood Formation and Anomaly Detection in Bipartite Graphs

Jimeng Sun[1] Huiming Qu[2] Deepayan Chakrabarti[3*] Christos Faloutsos[1†]
[1]Carnegie Mellon Univ. {jimeng, christos}@cs.cmu.edu
[2]Univ. of Pittsburgh huiming@cs.pitt.edu
[3]Yahoo! Research deepay@yahoo-inc.com

Abstract

Many real applications can be modeled using bipartite graphs, such as users vs. files in a P2P system, traders vs. stocks in a financial trading system, conferences vs. authors in a scientific publication network, and so on. We introduce two operations on bipartite graphs: 1) identifying similar nodes (Neighborhood formation), and 2) finding abnormal nodes (Anomaly detection). And we propose algorithms to compute the neighborhood for each node using random walk with restarts and graph partitioning; we also propose algorithms to identify abnormal nodes, using neighborhood information. We evaluate the quality of neighborhoods based on semantics of the datasets, and we also measure the performance of the anomaly detection algorithm with manually injected anomalies. Both effectiveness and efficiency of the methods are confirmed by experiments on several real datasets.

1 Introduction

A bipartite graph is a graph where nodes can be divided into two groups V_1 and V_2 such that no edge connects the vertices in the same group. More formally, a bipartite graph G is defined as $G = \langle V_1 \cup V_2, E \rangle$, where $V_1 = \{a_i | 1 \leq i \leq k\}$ and $V_2 = \{t_i | 1 \leq i \leq n\}$, $E \subset V_1 \times V_2$ as shown in Figure 1.

Many applications can be modeled as bipartite graphs, for example:

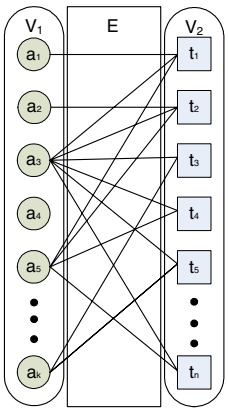


Figure 1. Bipartite Graph

1. *P2P systems:* V_1 is a set of files, and V_2 a set of peers. An edge e connects a file a and a peer t, if the peer t downloaded or uploaded the file a. In general, download/upload of a single file usually involves more than two peers. Ideally, files should be placed based on their "similarity", because peers are more likely to download files of the same style (or in the same neighborhood). Moreover, a peer that behaves much differently from others is also of interest: it usually consumes too much network resource or provides fictitious files for others.
2. *Stock markets:* The traders and stocks form V_1 and V_2 respectively, and the edges represent buying and selling actions between the two sets. It is useful to identify similar stocks and abnormal traders.
3. *Research publications:* Researchers publish in different conferences, and this conference-author relationship can be modeled as a bipartite graph. Similar conferences and interdisciplinary authors are again important.

In general, based on the application domain, the edges can be weighted. For instance, edge weights in the stock market graph can represent the trading volume, while in the research publication graph, they may represent the number of papers published by an author in a conference. For

* work performed while at CMU
† This material is based upon work supported by the National Science Foundation under Grants No. IIS-0083148, IIS-0209107, IIS-0205224, INT-0318547, SENSOR-0329549, EF-0331657, IIS-0326322, NASA Grant AIST-QRS-04-3031, CNS-0433540. This work is supported in part by the Pennsylvania Infrastructure Technology Alliance (PITA), a partnership of Carnegie Mellon, Lehigh University and the Commonwealth of Pennsylvania's Department of Community and Economic Development (DCED). Additional funding was provided by donations from Intel, and by a gift from Northrop-Grumman Corporation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, or other funding parties.

presentation purposes, we will only focus on unweighted graphs; our algorithms can be easily generalized to other graph types.

Under this setting, our work addresses two primary problems:

1. **Neighborhood formation(NF):** Given a query node a in V_1, NF computes the *relevance scores* of all the nodes in V_1 to a. The ones with higher relevance are the "neighbors" of a. For instance in the research publication example, given the conference ICDM, the neighborhood formation process computes the relevance scores for all the conferences. Presumably, the highest score is assigned to ICDM itself, with other data mining conferences like KDD, PAKDD, PKDD getting high scores as well.

2. **Anomaly detection(AD):** Given a query node a in V_1, AD computes the *normality score*s for nodes in V_2 that link to a. A node with a low normality score is an anomaly to a. In the research publication example, an author is an anomaly if he/she publishes at the conferences that have low relevance scores to each other. More intuitively, they are the persons who published in different fields.

Nodes that belong to the same group (V_1 or V_2) have the same type; it is the connections *between* the two types of objects that hold the key to mining the bipartite graph. Given the natural inter-group connections (between V_1 and V_2), our objective is to discover the intra-group relationships, such as the clusters and outlier within the group. For example, in the research publication bipartite graph, we have two natural groups of entities: conferences and authors. The relationship between these two groups is reflected by the edges. Based on these edges, we want to find the similar conferences and unusual authors that publish in different communities. An effective mining algorithm should thus be able to utilize these links across the two natural groups.

Our algorithm for NF is based on the idea of random walks with restarts [8]. The method is simple, fast and scalable. In addition, we approximate the NF computation by graph partitioning to further boost the performance.

The algorithm for AD uses the relevance scores from NF to calculate the normality scores. Intuitively, a node (in V_2) is an anomaly if it links to two nodes (in V_1) that do not belong to the same neighborhood/community. For example, an author becomes an anomaly if he/she publishes papers in conferences from two different fields. In the sequel, we will use neighborhood and community interchangeably.

Note also that a natural symmetry exists in the roles of neighborhoods and anomalies. In particular, we can swap V_1 and V_2 and apply the same algorithms in order to obtain the relevance score in V_2 and the normality score in V_1.
In summary, the contributions of the paper are that:

Symbol	*Description*
V_1	the set of k row nodes
V_2	the set of n column nodes
M	the k-by-n bipartite matrix
M^T	the transpose of M
M_A	the $(k+n)$-by-$(k+n)$ adjacent matrix
P_A	the $(k+n)$-by-$(k+n)$ Markov transition matrix
$rs(a)$	1-by-k relevance score vector for $a \in V_1$
RS	k-by-k similarity matrix where row i equals $rs(i)$
$ns(t)$	the normality score of the column node $t \in V_2$
S_t	the set of row nodes linking to t
RS_t	the similarity matrix for column node t
c	the restarting probability

Table 1. Symbol Table

1. we identify two important problems (Neighborhood Formation and Anomaly Detection) on bipartite graphs;
2. we develop the exact algorithms based on random walks with restarts;
3. we propose a faster approximate algorithm using graph partitioning;
4. the results can be easily interpreted by the user; and
5. we evaluate the methods on real datasets to confirm their applicability in practice.

Section 2 proposes the data model and the formal problem specification. Section 3 presents the algorithms. In section 4, we evaluate the algorithms with real data. We discuss the related work in section 5 and conclude in section 6.

2 Problem Definition

We will first define our data model and terminology, and then describe the exact formulations of the NF and AD problems.

Data model: The data is viewed as a bipartite graph $G = \langle V_1 \cup V_2, E \rangle$, where $V_1 = \{a_i | 1 \le i \le k\}$ and $V_2 = \{t_i | 1 \le i \le n\}$, $E \subset V_1 \times V_2$. The graph G is conceptually stored in a k-by-n matrix M [1], where $M(i, j)$ is the weight of the edge $< i, j >$. The value can be 0/1 for an unweighted graph, or any nonnegative value for a weighted graph. For example, the unweighted graph in Figure 1 becomes the following matrix:

$$M_{k\times n} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & 0 & 0 & \dots & 0 \\ 1 & 1 & 1 & 1 & 1 & \dots & 1 \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ 1 & 1 & 0 & 1 & 0 & \dots & 1 \\ \dots & & & & & & \\ 0 & 0 & 1 & 0 & 1 & \dots & 1 \end{pmatrix}$$

[1] In practice, we adopt the sparse matrix representation where the storage space is proportional to the number of non-zero elements in the matrix.

The nodes in $V_1(V_2)$ are called row(column) nodes. Note that a column node links to a row node if the corresponding matrix element is not zero. Moreover, row node a connects to another row node b if there is a column node c linking to both a and b. We call that path a *connection* between a and b through c. Nodes a and b can have multiple connections via different column nodes. For example in the matrix above, rows 3 and 5 links through column 1, 2, 4 and n.

We can construct the adjacency matrix M_A of G using M easily:

$$M_A = \begin{pmatrix} 0 & M \\ M^T & 0 \end{pmatrix} \quad (1)$$

In particular, $M_A(a,t)$ denotes the element at a-th row and t-th column in M_A.

Suppose we want to traverse the graph starting from the row node a. The probability of taking a particular edge $< a,\ t >$ is proportional to the edge weight over all the outgoing edges from a. More formally, $P_A(a,t) = M_A(a,t)/\sum_{i=1}^{k+n} M_A(a,i)$. Therefore, the Markov transition matrix P_A of G is constructed as: $P_A = col\_norm(M_A)$, where $col\_norm(M_A)$ normalizes M_A such that every column sum up to 1.

The main reasons to have M instead of working directly on M_A and P_A are the computational and storage savings. Next, we define the two problems addressed in the paper:

Neighborhood Formation (NF): Given a node $a \in V_1$, which nodes in V_1 are most related to a? There are two ways to represent the neighborhoods: 1) select a set of nodes as the neighbors and the other nodes are not the neighbors (Hard Neighborhood); 2) assign a relevance score to every node where "closer" nodes have high scores, and no hard boundary exists (Soft Neighborhood). In this paper, we adopt the soft neighborhood, because the score can help identify neighborhood but also differentiate the neighbors. In particular, we want to compute a relevance score to a for every node $b \in V_1$. The higher the score is, the more related that node is to a. More specifically, the node with the highest score to a is a itself; the nodes that are closer to a probably have higher scores than the other nodes that are further away from a.

Anomaly Detection (AD): What are the anomalies in V_2 to a query node a in V_1? Again we adopt the notion of soft anomalies by computing the normality scores for nodes in V_2 that link to a. Hence, the nodes with lowest normality score are the anomalies to a.

3 Proposed Method

In this section we discuss the algorithms that solve the two problems presented above. We first define relevance score and describe how to compute the relevance scores for the row nodes (neighborhood formation) in section 3.1. Then, based on the relevance scores, we define normality score and illustrate how to obtain the normality scores for the column nodes (anomaly detection) in section 3.2.

3.1 Algorithms for Neighborhood Formation

Given a row node $a \in V_1$, we want to compute a relevance score for each row node $b \in V_1$. The final result is a 1-by-k vector consisting of all the relevance scores to a.

Intuition: Intuitively, we do multiple random walks starting from a, and count the number of times that we visit each $b \in V_1$. These counts reflect the relevance of those nodes to a. The probability of visiting $b \in V_1$ from a is the relevance score we want to obtain. In the following, we list some scenarios on which the row nodes have high relevance scores.

b usually has a high relevance score to a if (1) b has many connections to a as shown in Figure 2; or (2) the connections only involve a and b as shown in Figure 3. Scenario (1) is obvious because the row nodes b and a have many connections through the columns nodes, which indicates the strong relevance between b and a. Scenario (2) is less obvious. The intuition is that the connection that only links a and b brings more relevance between a and b than the connections linking a, b and other nodes. The relevance score is not only related to the number of connections but also to the number of nodes involved in the connections. One observation is that the node b with the highest relevance score is not necessarily the one with most connections to a. The reason is that those connections link to nodes other than a and b as well. Thus, the relevance is spread out among many different nodes. Nevertheless, all the scenarios above are well captured by our algorithm in spite of its simplicity.

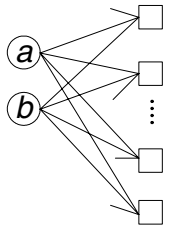


Figure 2. Many connections between a and b

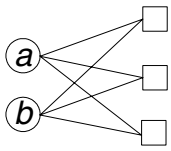


Figure 3. A few exclusive connection between a and b

Algorithms: We propose three methods for computing relevance scores: (1) **Exact NF** implements the basic idea but can have slow convergence rates, (2) **Parallel NF** implements the same algorithm in parallel, and (3) **Approximate NF** performs graph partitioning first which calculate results approximately but much more efficiently.

Exact NF: First, we transform the input row node a into a $(k+n)\times 1$ query vector $\vec{q}_a$ with 1 in the a-th row and 0 otherwise. Second, we need to compute the $(k+n)\times 1$ steady-state probability vector $\vec{u}_a$ over all the nodes in G. Last we extract the probabilities of the row nodes as the score vectors. Note that $\vec{u}_a$ can be computed by an iterated method from the following lemma.

Lemma 3.1. *Let c be the probability of restarting random-walk from the row node a. Then the steady-state probability vector $\vec{u}_a$ satisfies*

$$\vec{u}_a = (1-c)P_A\vec{u}_a + c\vec{q}_a \quad (2)$$

where P_A is already the column normalized.

Proof. See [16] □

Algorithm NFE(Exact NF)

Input: node a, bipartite matrix M, restarting probability c, tolerant threshold ε
0. initialize $\vec{q}_a = 0$ except the a-th element is 1 ($q_a(a) = 1$)
1. construct M_A (see Equation 1) and $P_A = col\_norm(M_A)$
2. while ($|\Delta\vec{u}_a|$[3] $> \varepsilon$)
   $\vec{u}_a = (1-c)P_A\vec{u}_a + c\vec{q}_a$
3. return $\vec{u}_a(1:k)$

The algorithm simply applies Equation 2 repeatedly until it converges. The actual computation of the algorithm can utilize the bipartite structure to have more saving. More specifically, we do not materialize M_A and P_A and modify Equation 2 as follows:

$$\vec{u}_a = (1-c)\begin{pmatrix} \texttt{col\_norm}(M)\vec{u}_a(k+1:k+n) \\ \texttt{col\_norm}(M^T)\vec{u}_a(1:k) \end{pmatrix} + c\vec{q}_a \quad (3)$$

where $\vec{u}_a(1:k)$ and $\vec{u}_a(k+1:k+n)$ are the vectors of first k and last n elements of $\vec{u}_a$, respectively. The relevance score $rs(a)$ is $\vec{u}_a(1:k)$. If we compute the relevance scores for all the nodes, we have a similarity matrix S.

The saving is significant when the number of rows k and the number of columns n differ a lot. Therefore, Equation 3 is always recommended in practice, while Equation 2 is only for demonstrating the concept.

[2] c is set to 0.15 for all experiments.
[3] We uses L_1 norm in the experiment.

Parallel NF: Very often we have the input of more than one row node. The task is to compute the relevance score for every input row node. Instead of applying algorithm NF_E for every input, we implement it in a more efficient way by running the algorithm in parallel for several inputs after vectorizing the code.

Approximate NF: One problem with the previous approaches is the large memory requirement. In particular, the algorithm is efficient when the entire matrix can fit in memory. However, one observation from our experiments suggest that that the relevance scores for the nodes are very skewed, with most nodes have almost zero relevance scores, and only a few nodes having high scores. This suggests that we can possibly filter out many "irrelevant" nodes before applying the NF computation. Based on this intuition, we apply graph partition first and perform NF only on the partition containing the query node. In particular, we use METIS [11] to partition the graph into κ non-overlapping subgraphs of about the same size. Note that the graph partition is a one-time cost to preprocess the data. The pseudo code is the following:

Algorithm NFA(Approximate NF)

Input: the bipartite graph G, the number of partitions κ, input node a
0. divide G into κ partitions $G_1 \ldots G_\kappa$ (one-time cost)
1. find the partition G_i containing a
2. construct the approximate bipartite matrix M' of G_i (ignore the edges cross two partitions)
3. apply NF_E on a and M'
4. set 0 relevance scores for the nodes that are not in G_i

3.2 Algorithm for Anomaly Detection

Based on the relevance scores for V_1 computed as shown above, we can compute the normality scores for the nodes in V_2. A node with a low normality score is an anomaly.

Given a column node $t \in V_2$, we first find the set S_t of row nodes to which t links: $S_t = \{a \mid < a,t > \in E\}$. Let k_t be the size of S_t. If t is "normal", then the relevance scores between any pair of elements in S_t should be high. More formally, we compute the k_t-by-k_t similarity matrix RS_t over S_t[4]. For example in Figure 1, for $t = 1, S_t = \{1,3,5\}$ and RS_t is a 3-by-3 matrix where each element is a relevance score from $a \in S_t$ to $b \in S_t$. Note that (1) RS_t is asymmetric, i.e., the relevance score from a to b may differ from the one from b to a, and (2) RS_t has a strong diagonal, i.e., every node has a high relevance score to itself. We ignore the diagonal for the normality score computation. In general, the normality score of t can be any function over RS_t. We define $ns(t)$ as the mean over all the non-diagonal elements in RS_t. The lower the normality score $ns(t)$ is, the more abnormal t is.

[4] Note that RS_t can be obtained by taking a subset of columns and rows from the k-by-k similarity matrix RS.

Essentially, given an input $t \in V_2$, we first compute the relevance score vectors for every adjacent row node S_t to t (using any of the NF methods described in section 3.1). Then we obtain the similarity matrix RS_t and apply the score function on RS_t. A computational trade-off is whether or not to pre-compute the relevance score vectors of all the row nodes. It usually depends on the number of row nodes involved. For example, if the dataset has a large number of rows and the input queries are skewed, pre-computation is not recommended, because it incurs huge cost and most of them is wasted due to the skewed distribution of the queries.

Algorithm AD(Anomaly Detection)

Input: input node t, bipartite transition matrix P

0. find the set $S_t = \{a_1, a_2, ...\}$ such that $\forall a_i \in S_t, < a_i, t > \in E$.
1. compute all the relevance score vectors $\vec{R}$ of $a \in S_t$
2. construct the similarity matrix RS_t from $\vec{R}$ over S_t
3. apply the score function over RS_t to obtain the final normality score $ns(t)$
4. return $ns(t)$

Examples of anomalies: Figure 4 shows the typical example of an anomaly t, which links to two row nodes a and b that communicate to different sets of nodes. Without t, a and b belong to different neighborhoods. Note that one requirement of constructing the neighborhoods is that a and b need to have enough connections to establish their identities. For example, a and b in Figure 4 still have a number of connections without t, while in Figure 5, b has no other connections apart from t. This implies in Figure 5 the neighborhood of b is unknown (or we do not have enough confidence to say whether b belongs to the same neighborhood as a or not). Therefore, t in Figure 5 will not be identified as an anomaly, while t in Figure 4 will.

On the other hand, the example in Figure 5 is easy to be found by simply counting the degree of the row nodes and picking the ones with only one connection. Potentially, the number of such nodes can be huge. The point is that our method aims at a non-trivial case of the anomaly, which tries to identify the connections across multiple neighborhoods. For example, author A and B write many papers with different groups of authors. If there is a paper between A and B, it will be an anomaly, because we know A and B belong to different neighborhoods. However, if B only has one paper and A is the co-author, we cannot decide whether the paper is an anomaly, because we do not know the neighborhood of B other than the sole paper with A.

4 Experiments

In this section we evaluate the exact and approximate methods on neighborhood formation and anomaly detection. We focus on answering the following questions:

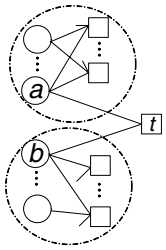


Figure 4. Without t, a and b belong to different neighborhoods

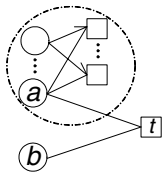


Figure 5. Without t, the identify of b is unknown

Dataset	Rows	Columns	Nonzeros	Weighted
CA	288590	2687	661535	yes
AP	315688	471514	1073168	no
IMDB	553388	204000	2269811	no

Table 2. Dataset summary

Q1: How accurate is the exact NF algorithm?

Q2: How accurate is the approximate NF algorithm?

Q3: Can the AD algorithm detect the injected anomalies?

Q4: What about the computational cost of different methods?

After describing the experimental settings in section **??**, we answer Q1 in section 4.2, using concrete examples from different datasets (i.e., compare exact NF algorithm vs. "ground truth"). Section 4.3 uses quantitative metric to compare approximate NF methods vs. exact NF method (Q2). Section 4.4 answers Q3 by injecting artificial anomalies and evaluating the performance on that. Finally, section 4.5 answers Q4 by providing the quantitative evidence of the dramatic computational saving on space and execution time.

4.1 Experiment Setting

Datasets: We construct the graphs using three real datasets, whose size are specified in Table 2.

Conference-Author(CA) dataset: Every row represents a conference; every column represents an author. The elements in the bipartite matrix M are nonnegative integers. On average, every conference has 510 authors, every author publishes in 5 conferences.

Author-Paper(AP) dataset: Every row represents an author; every column represents a paper. The elements in the bipartite matrix M are either 0 or 1. In particular, $M(i, j) = 1$ indicates that the i-th author is an author for the j-th paper. On average, every author has 3 papers, every paper has 2 authors. The distribution is very skewed as most of authors have only one paper.

IMDB dataset: Every row is an actor/actress; every column is a movie. The elements in the bipartite matrix M are either 0 or 1. In particular, $M(i, j) = 1$ indicates that the i-th actor/actress is in the j-th movie. On average, every actor/actress plays in 4 movies, and every movies has 11 actors/actresses.

4.2 (Q1) Evaluation of Exact NF

Exact NF: We want to check whether the nodes with high relevance scores are closely related to the query node. The goal is to ensure the result makes sense in the context of the applications. More specifically, we select some rows from the three datasets as the query nodes and verify the NF scores through user study. Due to the page limit, we just show one example from each dataset.

CA dataset: Figure 6(a) shows the top 10 neighbors of ICDM conference. As expected, the most related conferences are the other data mining conferences: KDD, PAKDD, PKDD. After that, the database conference (ICDE, SIGMOD, VLDB) and the machine learning conference (ICML) also have high relevance scores [5].

AP dataset: Figure 6(b) plots the top 10 neighbors of Prof. Jiawei Han. They are indeed the close collaborators to Prof. Han, who either have many joint papers with Prof. Han or have several exclusive joint papers.

IMDB dataset: For IMDB dataset, we perform the same set of experiment as above. Unlike the previous two datasets, the people in this dataset are not well-clustered, meaning that if a and b play in the same movie, it does not increase the likelihood that they will play together again in the future. Of course, they are exceptions in the sequels of successful movies.

We choose Robert De Niro as an example here. The persons with the highest relevance scores, as shown in Figure 6(c), are Billy Crystal and Lisa Kudrow because they all perform in the same 2 movies ("Analyze this" and the sequel "Analyze that"). Furthermore, they are the only main actors/actress in the movies. This is again due to the result of the combination of 2 scenarios in section 3.1.

4.3 (Q2) Evaluation of Approximate NF

We partition each dataset into k partitions with equal size using METIS [11]. The NF computation for a row node a only involves the nodes in the same partition as a (we assign 0 relevance scores to the row nodes in other partitions). The goal is to show that the neighborhood does not change much using the approximate method(partition method). The metric we use is *precision*, that is, the number of common neighbors over the neighborhood size. We set the neighborhood size to 10 and vary the number of partition κ as shown in Figure 7 as a function of *precision*. We observe the precision does not drop much, which suggests that the approximate method works well. We also vary the neighborhood size as a function of *precision* in Figure 8, while setting the number of partition $\kappa = 10$. We observe that the neighborhood can be fairly accurately captured over different ranges. Note that it does not make sense to have large neighborhood size for this evaluation, because the relevance scores will become very low (practically zero) for most of the nodes in the neighborhood. In particular, the effective neighborhood size for dataset AP is rather small, because the most of people only co-author with a small number of people. As a result, the precision drops faster as the neighborhood size increases.

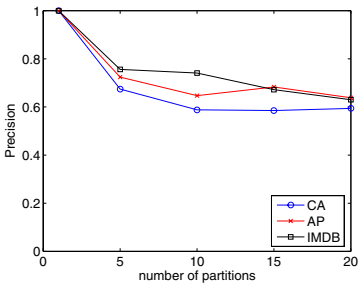


Figure 7. Precision(y-axis) vs. number of partition(x-axis)

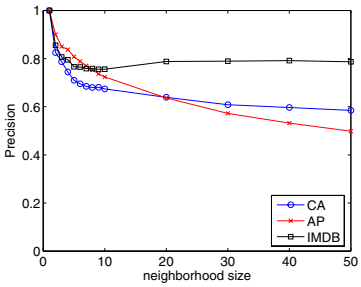
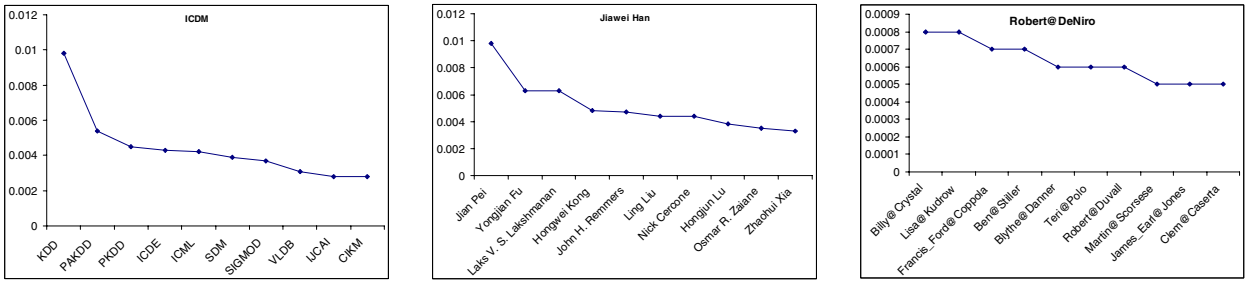


Figure 8. Precision(y-axis) vs. neighborhood size(x-axis)

4.4 (Q3) Evaluation of Anomaly Detection

Due to lack of information about the real anomalies, we manually inject random connections between nodes. In particular, we inject 100 column nodes in each dataset connect-

[5] Similar to our example, Klink et al. [12] developed DBLP browser which uses an author-based similarity metric to model the closeness of two conferences. i.e., two conferences that have many common authors are highly similar.



(a) ICDM (b) Jiawei Han (c) Robert DeNiro

Figure 6. Examples for neighborhood formation from three datasets

ing to k row nodes, where k equals the average degree of column nodes. The row nodes are randomly selected among the column nodes with large degree (greater than 10 times the average)[6]. Note that the difference between using exact and approximate NF is marginal. And we use approximate NF in the AD algorithm to reduce computational cost.

Figure 9 plots the average normality scores of genuine and injected nodes over three different datasets. We observe a big gap of the normality scores between genuine and injected ones. Hence, we can easily identify the anomalies by looking at the ones with the lower scores within the same dataset. Note that only the relative score matters in detecting anomaly not the absolute score. And it is hard to compare the scores across datasets because of the different graph structure.

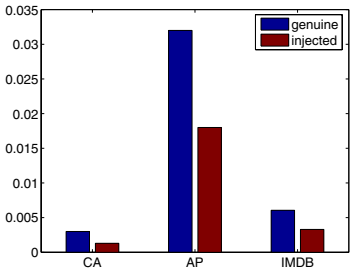


Figure 9. Normality scores between genuine and injected nodes across 3 datasets

4.5 (Q4) Evaluation of the computational cost

All the computation of different methods boils down to the NF computation. The only difference is how large the matrix is. Intuitively, the computational cost is large if we work with the entire dataset. It is usually beneficial to partition the dataset. The partition incurs a one time cost which can be amortized over the future queries (involving NF and AD computation). Figure 10 shows the computation cost on neighborhood formation vs. the number of partitions. Note that a dramatic cost reduction can be found when using the approximate NF computation method (the partition method).

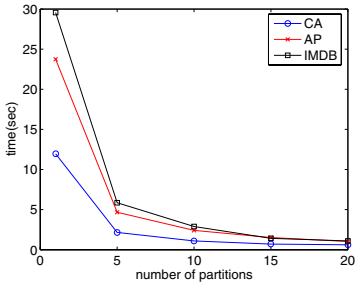


Figure 10. Computation time(sec): y-axis vs. number of partitions: x-axis

5 Related Work

There is a significant body on research related to our problem, which we categorize into four groups: graph partitioning,outlier detection on graphs, random walks on graphs, collaborative filtering.

Graph Partitioning: Popular methods for partitioning graphs include the METIS algorithm [11], spectral partitioning techniques [10], flow-based methods [6] information-theoretic methods [5], and methods based on the "betweenness" of edges [7], among others. These typically require some parameters as input; Chakrabarti [4] uses MDL criterion to automatically determine both the number of clusters and their memberships. Note that our work is orthogonal to this, and we can use any graph-clustering algorithm. In addition, as a by-product of our algorithms, the neighborhoods over nodes can represent *personalized clusters* depending on different perspectives.

Outlier Detection on Graphs: Autopart [3] finds outlier edges in a general graph; however, we need to detect outlier *nodes*. Noble and Cook [13] study anomaly detection on general graph with labeled nodes; yet, their goal is to identify abnormal substructure in the graph, not the abnormal *nodes*. Aggarwal and Yu [1] propose algorithms to find outliers in high-dimensional spaces, but its applicability to graphs is unclear: the nodes in a graph lie in a vector space

[6]The reason for not using all the row nodes is that most of row nodes have degree one and the injection to those nodes will not lead to an anomaly because statistically we do not have enough information to tell whether the injection is an anomaly.

formed by the graph nodes themselves, so the vector space and the points in it are related.

Random-walk on Graphs: Page-Rank [2] learns the ranks of web pages using the iterated power method on web graph M (adjacency matrix of the entire graph). The ranks of all webpages are cast as an N-dimensional vector, and then the fixed point is found for the following equation: $\vec{r} = (1-\alpha)M \times \vec{r} + \alpha\vec{p}$, where the α is the damping factor and $\vec{p} = [\frac{1}{N}]N \times 1$. Thus, there is an uniform prior on all the web pages. In order to deal with personalized query, Topic-Sensitive PageRank [8] increases the importance of certain web pages by putting non-uniform weights for $\vec{p}$. Similar random-walk approaches have been used into other domains; for example, Mixed Media Graph(MMG) [14] applies random walk with restart on image captioning application. We plan to further explore the random walk algorithm on bipartite graph and use it to identify anomaly nodes. Similar idea also appear in SimRank [9] which is a similarity measure between nodes in a graph with the intuition that two nodes are similar if they are related by similar nodes.

Collaborative Filtering Collaborative filtering is a well-studied method of making automatic filtering about the user interests based on the historical information from many users (collaborating) [15]. The goal is to develop a recommendation system not to find anomalies.

6 Conclusion

A variety of datasets can be modeled as bipartite graphs, such as P2P networks, stock trades, author-paper relationships, and so on. This paper addresses two problems on such bipartite graphs: 1) neighborhood formation; 2) anomaly detection. The main properties of the methods are:

- Fast convergence
- Scalability to large graphs
- Simplicity of implementation
- Results that are easily interpreted

The main idea is to use random-walk with restarts and graph partitioning. We evaluate the methods on several real datasets. Our experiments confirm the efficiency as well as the effectiveness of the proposed methods.

References

[1] C. Aggarwal and P. Yu. Outlier detection for high-dimensional data. In *SIGMOD*, pages 37–46, 2001.

[2] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

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

[4] Deepayan Chakrabarti, Spiros Papadimitriou, Dharmendra S. Modha, and Christos Faloutsos. Fully automatic cross-associations. In *KDD*, pages 79–88. ACM Press, 2004.

[5] I. S. Dhillon, S. Mallela, and D. S. Modha. Information-theoretic co-clustering. In *KDD*, 2003.

[6] Gary William Flake, Steve Lawrence, and C. Lee Giles. Efficient identification of Web communities. In *KDD*, 2000.

[7] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. In *Proc. Natl. Acad. Sci. USA*, volume 99, 2002.

[8] T. Haveliwala. Topic-sensitive pagerank. In *Proceedings of the Eleventh International World Wide Web Conference*, 2002.

[9] Glen Jeh and Jennifer Widom. Simrank: a measure of structural-context similarity. In *KDD*, 2002.

[10] R. Kannan, S. Vempala, and A. Vetta. On clusterings – good, bad and spectral. In *FOCS*, 2000.

[11] George Karypis and Vipin Kumar. Multilevel k-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed Computing*, 48(1):96–129, 1998.

[12] Stefan Klink, Michael Ley, Emma Rabbidge, Patrick Reuther, Bernd Walter, and Alexander Weber. Browsing and visualizing digital bibliographic data. In *VisSym*, pages 237–242, 2004.

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

[14] Jia-Yu Pan, Hyung-Jeong Yang, Pinar Duygulu, and Christos Faloutsos. Automatic multimedia cross-modal correlation discovery. In *KDD*, 2004.

[15] Upendra Shardanand and Pattie Maes. Social information filtering: Algorithms for automating "word of mouth". In *Human Factors in Computing Systems*, 1995.

[16] Gilbert Strang. *Introduction to Linear Algebra*. Wellesley-Cambridge Press, 3 edition, 1998.