Summarization-based Mining Bipartite Graphs

Jing Feng[1], Xiao He[1], Bettina Konte[1], Christian Böhm[1], Claudia Plant[2]
[1]: University of Munich, [2]: Florida State University
{feng, he, konte, boehm}@dbs.ifi.lmu.de, cplant@fsu.edu

ABSTRACT

How to extract the truly relevant information from a large relational data set? The answer of this paper is a technique integrating graph summarization, graph clustering, link prediction and the discovery of the hidden structure on the basis of data compression. Our novel algorithm *SCMiner* (for Summarization-Compression Miner) reduces a large bipartite input graph to a highly compact representation which is very useful for different data mining tasks: 1) Clustering: The compact summary graph contains the truly relevant clusters of both types of nodes of a bipartite graph. 2) Link prediction: The compression scheme of *SCMiner* reveals suspicious edges which are probably erroneous as well as missing edges, i.e. pairs of nodes which should be connected by an edge. 3) Discovery of the hidden structure: Unlike traditional co-clustering methods, the result of *SCMiner* is not limited to row- and column-clusters. Besides the clusters, the summary graph also contains the essential relationships between both types of clusters and thus reveals the hidden structure of the data. Extensive experiments on synthetic and real data demonstrate that *SCMiner* outperforms state-of-the-art techniques for clustering and link prediction. Moreover, *SCMiner* discovers the hidden structure and reports it in an interpretable way to the user. Based on data compression, our technique does not rely on any input parameters which are difficult to estimate.

Categories and Subject Descriptors

H.2.8 [**Database applications**]: Data Mining

Keywords

bipartite graph, summarization, clustering, link prediction

1. MOTIVATION

Relational or graph-structured data are prevalent in nature, science and economics. Many aspects of real life can be well represented as a bipartite graph which has two types of vertices and edges representing the connections between them. Consider for example newsgroups where one type of vertices represents persons and the other type of vertices represent groups. An edge between a person and a group means that he or she is member of this group. Or consider the interaction between drugs and proteins: An edge between a particular substance and a protein means that the corresponding protein is responding to the drug. Such bipartite graphs are stored as large adjacency matrices often having millions of vertices and edges. Effective and efficient data mining methods are essential for answering high-level domain specific questions like: Which users are potentially interested to join a newsgroup? Or, which combination of substances is most effective against a certain disease?

Therefore, in recent years the research topic of mining bipartite graphs has attracted much attention, with a large volume of research papers, e.g. [1, 3, 5, 10, 11, 14] to mention a few. To extract knowledge from a large bipartite graph, existing techniques apply one of the following two basic strategies: (1) Clustering. These approaches reduce the complexity by partitioning the large input graph into smaller groups of similar vertices which can be inspected and interpreted by the user. In particular, approaches to co-clustering like [1, 3] cluster both types of vertices, i.e. the rows and the columns of the adjacency matrix simultaneously guided by the idea that the clustering of rows and columns can profit from another. The output of co-clustering is a set of row-clusters and a set of column clusters. (2) Summarization. These approaches reduce the complexity not by grouping but by a global abstraction. The output of summarization techniques like [14] is a bipartite graph which is much smaller than the original input graph. Ideally, it distills the major characteristics from the input data and is small enough to be accessible for the user. A third line of papers focuses on a different, however closely related topic: Link prediction. These approaches like [11] study the question whether there should be an edge between two currently disconnected vertices. Link prediction is closely related to summarization and clustering since a technique for link prediction must capture at least the local structure of the graph to provide reasonable predictions. Also clustering and summarization are closely related since during the process of abstraction, both approaches must identify the relevant major characteristics of the graph.

Contributions

In this paper, we therefore propose *SCMiner*, a technique integrating summarization, clustering and link prediction on bipartite graphs. The name *SCMiner* stands for Summ-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
KDD'12, August 12–16, 2012, Beijing, China.
Copyright 2012 ACM 978-1-4503-1462-6/12/08... $15.00.

arization-Compression Miner and the principle of data compression also known as the Minimum Description Length Principle (MDL) [16] is the basis of our technique. The basic idea is to transform the original graph into a very compact summary graph. During the process of transformation which is controlled by the MDL principle, our technique discovers the major clusters of both vertex types as well as the major connection patterns between those clusters. The result of *SCMiner* comprises a compressed graph, the row- and column-clusters and their link patterns. The major contributions of our approach can be summarized as follows:

- **Clustering plus hidden structure mining.** Like state-of-the-art co-clustering methods, *SCMiner* accurately identifies the row- and column clusters of bipartite graphs and even outperforms them on some data sets. As a key feature of our approach, the result does not only consist of two sets of clusters. *SCMiner* also reveals the relationships between the clusters which are essential for interpretation.
- **Accurate link prediction.** *SCMiner* accurately predicts missing or future links and removes noise edges.
- **Unsupervised graph mining all in one.** *SCMiner* integrates summarization, clustering and link prediction and thus comprehensively supports unsupervised graph mining.
- **Results validated by data compression.** Data compression is an intuitive optimization goal with many benefits: By natural balancing goodness of fit and model complexity, overfitting is avoided. The results are thus simple and interpretable. Moreover, supported by the MDL principle *SCMiner* does not rely on any difficult to estimate input parameters.

The remainder of this paper is organized as follows: In the following section, we give the model formulation. Section 3 presents the algorithm in detail. Section 4 contains an extensive experimental evaluation. Section 5 briefly surveys related work and Section 6 concludes the paper.

2. COMPRESSING A BIPARTITE GRAPH

In this section, we first present a simple example to introduce the terminology of graph summarization. Then an MDL based coding schema is derived to find the best summarization of a bipartite graph. At last, we propose a strategy to discover hidden relations between vertices of different type.

Notation and Example. Figure 1 depicts a simple example for graph summarization of a bipartite graph. $G = (V_1, V_2, E)$ is an unweighted bipartite graph where $V_1 = \{V_{11}, ..., V_{16}\}$ and $V_2 = \{V_{21}, ..., V_{26}\}$ denote two types of vertices and E denotes the edges between them. The summarization of graph G consists of two bipartite graphs, a summary graph G_S and an additional graph G_A. The summary graph $G_S = (S_1, S_2, E')$ is an aggregated graph and is composed of four super nodes $S_{11} = \{V_{11}, V_{12}, V_{13}\}$, $S_{12} = \{V_{14}, V_{15}, V_{16}\}$, $S_{21} = \{V_{21}, V_{22}, V_{23}\}$, $S_{22} = \{V_{24}, V_{25}, V_{26}\}$, and super edges E'. The super nodes themselves are composed of vertices exhibiting the exact same link pattern. They indicate the local cluster structure of each type of vertices, whereas the super edges represent the global relations between local clusters. The additional graph $G_A = (V_1, V_2, E'')$ contains normal nodes and correction edges which are required to reconstruct the bipartite graph G. The + or − symbols above the edges indicate whether it is required

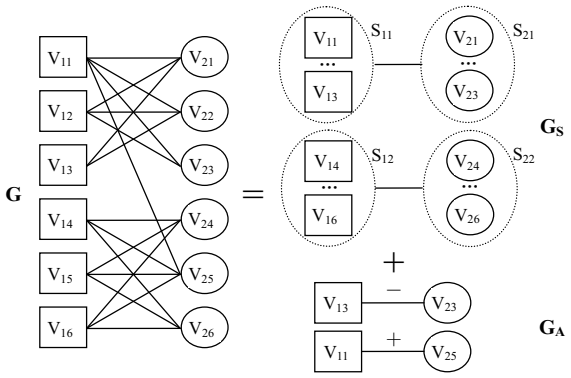


Figure 1: Summarization and Compression.

to add or remove them from G_S to recreate G. The correction edges describe the revealed hidden relations between different types of vertices.

2.1 Coding Scheme

A bipartite graph can be represented by thousands of summarizations, however, the challenge is to find out the one that best represents the data and reflects the hidden structure behind the surface data. The Minimum Description Length (MDL) principle is an intuitive choice for model selection [16]. It follows the assumption that the more we are able to compress the data, the more we have learned about its underlying patterns. Formally, the goodness of the model can be stated as shown in Eq.(1), where $L(M)$ denotes the cost for coding the model parameters and $L(D|M)$ represents the cost of describing the data D under the model M. As the model has to be coded with the data and too complex models result in high compression cost, MDL naturally avoids overfitting.

$$L(M, D) = L(D|M) + L(M). \tag{1}$$

Inspired by the MDL principle, we propose a coding schema to choose the best graph summarization. The most direct and simple way to represent a bipartite graph $G = (V_1, V_2, E)$ is to compress its adjacency matrix $A \in |V_1| \times |V_2|$, with $a_{ij} = 1$ if $(V_{1i}, V_{2j}) \in E$. The coding cost of the adjacency matrix A is lower bounded by its entropy which is provided by:

$$CC(G) = -|V_1| \cdot |V_2| \cdot H(A), \tag{2}$$

where $H(A) = -(p_n(A) \cdot \log_2 p_n(A) + p_r(A) \cdot \log_2 p_r(A))$, $p_n(A)$ and $p_r(A)$ are the probabilities of finding 1 and 0 entries in the adjacency matrix A of G, i.e. the probabilities to observe edges or not.

As mentioned above, the summarization of a bipartite graph G is composed of two bipartite graphs, the summary graph G_S and the additional graph G_A. Instead of transferring G from a sender to a receiver, we can transfer G_S, G_A and the group information of super nodes in G_S. In addition, there is no need to send the symbol information of edges in G_A, since this information can be obtained by comparing G_A and G_S. If the symbol of an edge in G_A is +, this edge is not present in G_S, and vice versa, if the symbol is −, G_S contains the edge. The following equation defines the coding cost of a summarization of the graph.

Definition 1 (Coding Cost of a Graph.).

$$CC(G) = CC(G_S) + CC(G_A) + \sum_{i=1}^{K}\sum_{j=1}^{N_i} |S_{ij}| log_2 \frac{|V_i|}{|S_{ij}|}, \quad (3)$$

where $CC(G_S)$ and $CC(G_A)$ denote the coding cost of summary graph G_S and the additional graph G_A defined by Eq.(2). The third term is the cost of coding the group information, where K is the number of node types, N_i is the number of super nodes in type i, $|S_{ij}|$ is the number of members in super node S_{ij}, $|V_i|$ is the number of original nodes in type i.

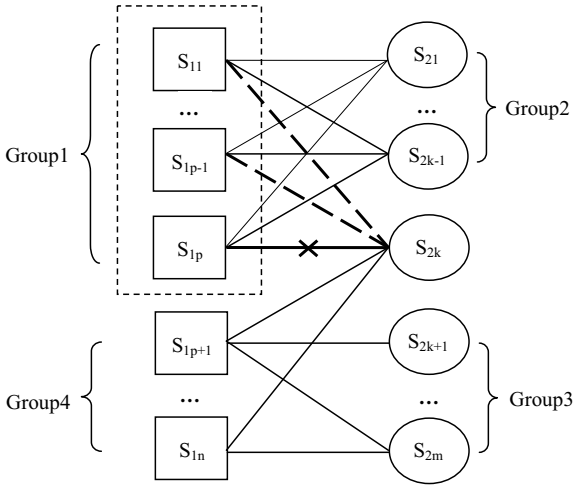
$L(D|M) = CC(G_S)$ is the description of the data under the model M with coding cost $L(M) = CC(G_A) + CC(group)$. The optimization goal of our algorithm *SCMiner* is to find the best model or summarization that minimizes Eq.(3).

2.2 Hidden Relations Between Vertices

The data we record in real life applications does often only approximately represent the ground truth due to inconsistencies and measurement errors. For example, the same user often joins newsgroups under different nicknames and email addresses. Or a protein might show a strong response to a substance simply due to a measurement error. Thus real world graphs are often spoiled by erroneous connections on the one hand and are also missing important links on the other hand.

Take a close look at the graph G in Figure 1, obviously we have the feeling that edge (V_{13}, V_{23}) is probably missing, and edge (V_{11}, V_{25}) maybe presents an artifact of noise. Therefore, if we add edge (V_{13}, V_{23}) to G and remove edge (V_{11}, V_{25}) from G, we can form four super nodes, whose members exhibit the exact same connection patterns. These connections probably reflect the true relationships. Based on these observations, we propose the following strategy to discover the hidden relations between vertices.

Figure 2 describes our strategy for merging nodes, suppose all the nodes are super nodes. The nodes in Group 1 share a similar link pattern and could therefore be merged into a super node. Nodes in Group 2 and Group 3 are both hop two neighbors of node S_{2k}, specifically nodes in Group 2 are common neighbors of nodes in Group 1 and nodes in Group 3 are not. To form a new super node of nodes in Group 1 their link pattern should be exactly the same. However, the link pattern of node S_{1p} in Group 1 is similar to those of the other nodes, but not the same. Concretely, S_{1p} has an extra link with S_{2k} that the other nodes lack. As we can see from Figure 2, two methods can be adopted to make nodes in Group 1 exhibiting the same link pattern: we could either remove edge (S_{1p}, S_{2k}) or add edges $(S_{11}, S_{2k}), (S_{12}, S_{2k}), ..., (S_{1p-1}, S_{2k})$. The question is, how to distinguish whether edges should be removed or added? We can decide if we add or remove edges by calculating some cost function. We define the cost as the number of edges we add or delete, which is highly related to the MDL costs of Eq.(3), in order to make the link patterns of merging nodes equal. Specifically, as shown in Figure 2, there are p nodes in Group 1 $S_{11}, S_{12}, ..., S_{1p}$, among which S_{1p} exhibits a similar but not exact same link pattern as the other nodes. Therefore, the cost of removing edge (S_{1p}, S_{2k}) is 1 and the cost of adding edges $(S_{11}, S_{2k}), (S_{12}, S_{2k}), ..., (S_{1p-1}, S_{2k})$ is $p-1$. Obviously, the former method should be selected because of the lower cost. The pseudocode of modifying the edges of a merging group is shown in Algorithm 1. At first



Group1: Nodes to be merged
Group2: Hop 2 neighbors of S_{2k} and common neighbor of Group1
Group3: Hop 2 neighbors of S_{2k} but not common neighbor of Group1
Group4: Neighbors of S_{2k}

: The edge to be deleted
: The edge to be added

Figure 2: The strategy used for merging nodes

Algorithm 1 ***ModifyEdge***

//Modify edges of *group* to make their link same

Input: Group nodes *group*, G_S, G_A
Output: G_S, G_A, *hop2Sim*

alln = Neighbor(*group*);
cn = CommonNeighbor(*group*);
for Each node $S \in alln$ and $S \notin cn$ **do**
  Using Eq.(4) and Eq.(5) to add or remove edge;
  Add or remove edges in G_S;
  Add additional edges to G_A;
end for
Update *hop2Sim* for each $S \in alln$ and $S \notin cn$;
return G_S, G_A, *hop2Sim*;

we combine neighbors and common neighbors of nodes in merging groups, then we need to modify the edges of those nodes which are neighbors but not common neighbors of the merging group to make the link pattern of all group members exactly the same. Suppose the merging group contains $S_{11}, S_{12}, ..., S_{1p}$ and S_{2k} is one of their neighbors but not a common neighbor. The cost of removing and adding edges can be calculated by Eq.(4) and Eq.(5). The cost of a super edge is calculated as $|S_{1i}| \cdot |S_{2k}|$, where $|.|$ is the number of normal nodes contained in a super node. However, in some cases the costs for removing might be the same as for adding edges. If so, we use the similarity proposed in the next section to decide which action to take. Naturally, if S_{2k} is more similar to the nodes of Group 2 compared to those of Group 3, we add edges for merging, while otherwise we

remove edges.

$$Cost_{remove} = \sum_{i=1}^{p} \begin{cases} |S_{1i}| \cdot |S_{2k}| & \text{if } S_{1i} \text{ links to } S_{2k} \\ 0 & \text{otherwise.} \end{cases} \quad (4)$$

$$Cost_{add} = \sum_{i=1}^{p} \begin{cases} 0 & \text{if } S_{1i} \text{ links to } S_{2k} \\ |S_{1i}| \cdot |S_{2k}| & \text{otherwise.} \end{cases} \quad (5)$$

3. ALGORITHM SCMINER

In this section, we propose our algorithm *SCMiner* to find the best summarization of a bipartite graph. It can be proven that finding the global optimal summarization is NP-hard, therefore *SCMiner* follows a heuristic approach to search the local optima.

Basic Idea. The idea behind the algorithm *SCMiner* is that nodes with similar link patterns can be merged to groups, if the similarity between each pair of nodes in the group is bigger than some threshold th. By adding and removing edges as proposed in Section 2 we accomplish that all group nodes share the exact same link pattern and so can form a super node. *SCMiner* iteratively merges groups of nodes or super nodes whose similarities are bigger than th. The initial threshold is 1 and th is reduced stepwise by ϵ when no pair of nodes can be merged. In each iteration we calculate the new summarization coding cost. Then the MDL principle is used to choose the best summarization. The algorithm terminates when th reaches 0.

Finding Super Node Candidates. To find suitable candidate groups of nodes where merging might pay-off, we introduce the notion of hop two similarity. Suppose two super nodes S_{1i} and S_{1j} have n common neighbors $\{S_{21}...S_{2n}\}$ and m neighbors $\{S_{2(n+1)}...S_{2(n+m)}\}$ that are connected to only one of the two super nodes. Intuitively, if the two nodes have more common neighbors than neighbors only linking to one node, they are more similar in link pattern, then different. We can define their similarity as

Definition 2 (Hop Two Similarity.).

$$sim(S_{1i}, S_{1j}) = \frac{\sum_{k=1}^{n} |S_{2k}|}{\sum_{k=1}^{n+m} |S_{2k}|} \quad (6)$$

where $|S_{2i}|$ is the number of normal nodes contained in super node S_{2i}. This similarity measure ranges from 0 to 1.

As described above, we only need to calculate the similarity between two nodes if they share at least one common neighbor and therefore are hop two neighbors of each other. In the following we call the similarity between a node and all its hop two neighbors its hop two similarity. **Algorithm.** Now we describe our algorithm *SCMiner*, the pseudocode is provided in algorithm 2. The input parameters of *SCMiner* are the bipartite graph $G = (V, E)$, where $V = (V_1, V_2)$ and E are the edges between V_1 and V_2, and stepsize ϵ for reducing th. The output of *SCMiner* is the summarization of G, including the summary graph $G_S = (S, E_S)$ composed of super nodes $S = (S_1, S_2)$ and the additional graph $G_A = (V, E')$ composed of single nodes $V = (V_1, V_2)$, which has the minimum coding cost regarding the proposed coding scheme. In the initialization phase, we first set the summary graph G_S to the input graph G, which means that all single nodes are treated as super nodes containing only one normal node, and set the additional graph G_A empty. Then we initialize the coding cost *mincc* using *Eq.*(3) with G_S and G_A. Afterwards we compute the similarities between each node and its hop two neighbors of same node type. In the searching phase, when $th > 0$, we do the following steps: First we search for groups of nodes that have at least one hop two neighbor with similarity larger than th and then we merge every group we found. When there is no more group of nodes that can be merged, we decrease the threshold th by ϵ. In the merging phase, we use the proposed method shown in algorithm 1 to modify the edges of the merging group that are present in G_S to get nodes with exact same link structure. Subsequently we add the corresponding additional edges to G_A and update the hop two similarity for affected nodes, which are neighbors of merging group nodes but not their common neighbors. During the merging phase, nodes with similar link patterns are merged, which decreases the data cost and increases the model cost, while the total cost is reduced in most cases. After each merging step, we calculate the coding cost using Eq.(3) with current G_S and G_A and the best summarization with minimum coding cost is stored in $bestG_S$ and $bestG_A$. Finally, we output $bestG_S$ and $bestG_A$ with coding cost.

Algorithm 2 ***SCMiner***

Input: Bipartite graph $G = (V, E)$, Reduce step ϵ
Output: Summary Graph G_S, Addition Graph G_A

```
//Initialization
G_S = G, G_A = (V, ∅);
Compute mincc using Eq.(3) with G_S and G_A;
bestG_S = G_S, bestG_A = G_A;
Compute hop2Sim for each S ∈ G_S using Eq.(6);

//Searching for best Summarization
while th > 0 do
  for each node S ∈ G_S do
    Get SN with S' ∈ SN and hop2Sim(S, S') > th;
  end for
  Combine SN and get non-overlapped groups allgroup;
  for each group ∈ allgroup do
    ModifyEdge(group, G_S, G_A);
    Merge nodes of G_S with same link pattern;
    Compute cc using Eq.(3) with G_S and G_A;
    Record bestG_S, bestG_A, and mincc if cc < mincc;
  end for
  if allgroup == ∅ then
    th = th − ε;
  else
    th = 1.0;
  end if
end while
return bestG_S, bestG_A;
```

Properties. We adjust the parameter ϵ using the MDL principle. Extensive experiments on synthetic and real data showed that a suitable range for ϵ is around 0.01 to 0.1. Therefore we try out every setting with a step size of 0.01 and let MDL select the best result. The runtime complexity for the computation of the similarity between each vertex v and its hop two neighbors is $O(N \cdot d_{av}^3)$, where N is the number of vertices and d_{av} is the average degree of each vertex. During each merging step in *SCMiner*, we only compute the similarities between some affected vertices and their two hop neighbors. Therefore the runtime complexity is roughly $O(d_{av}^4)$. The number of merging steps, affected by the re-

Table 1: Synthetic bipartite graphs

Data Set	S	T
BP1	$\begin{pmatrix} 0.8 & 0.2 \\ 0.1 & 0.9 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$
BP2	$\begin{pmatrix} 0.9 & 0.8 & 0.1 \\ 0.1 & 0.9 & 0.8 \\ 0.1 & 0.2 & 0.9 \end{pmatrix}$	$\begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$
BP3	$\begin{pmatrix} 0.8 & 0.7 & 0.2 & 0.8 \\ 0.9 & 0.3 & 0.8 & 0.2 \\ 0.3 & 0.8 & 0.2 & 0.7 \\ 0.9 & 0.8 & 0.7 & 0.2 \end{pmatrix}$	$\begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$

duction stepsize ϵ, is N in average. So the whole runtime complexity is $O(N \cdot d_{av}^4)$.

4. EXPERIMENTS

This section provides empirical evidence to show the effectiveness of *SCMiner* on synthetic and real data. In particular, we evaluate *SCMiner* three aspects: First, we compare *SCMiner* with state-of-the-art co-clustering algorithms in terms of quality of the clusters detected on each type of nodes. Secondly, we evaluate the hidden structure, i.e. the relationships between both types of nodes found by *SCMiner*. Finally, we compare *SCMiner* with some state-of-the-art link prediction methods to evaluate the validity of hidden relationships discovered by *SCMiner*.

Data Sets. The generated synthetic bipartite graphs with different parameters are shown in Table 1. Both V_1 and V_2 contain the same number of clusters, and each cluster has 100 nodes. The matrix T shows the ground truth links between clusters of different type, where T_{pq} can take the values 1 or 0, which means the pth cluster of V_1 and the qth cluster of V_2 are fully connected or separated. The matrix S introduces link parameters to matrix T, where S_{pq} denotes the percentage of links generated between the pth cluster of V_1 and the qth cluster of V_2. In other words, if link parameters are introduced based on 0, noisy links are added to the ground truth graph. If link parameters are introduced based on 1, links are removed from the ground truth graph, where the percentage is 1 minus the link parameters.

Three real data sets are evaluated in our experiments. World cities[1] data consists of the distribution of offices from 46 global advanced producer service firms over 55 world cities. Global firms are defined as firms owning offices in at least 15 different cities. Service values for a firm in a city are given as 3,2,1 and 0. We binarize the data set such that positive service values become 1 and then generate the bipartite graph. The advanced producer service firms can be categorized into 4 clusters: accountancy firms, advertising firms, banking and finance firms and law firms.

MovieLens[2] data was collected through the MovieLens web site during the seven-month period from September 19th, 1997 to April 22nd, 1998 by the GroupLens Research Project at the University of Minnesota. It consists of 100,000 ratings (1-5) from 943 users on 1682 movies. We preprocess the data by removing movies which are rated by less than 30 users and users that rated less than 30 movies. Therefore, we got a bipartite graph of 361 users and 334 movies. Then we binarize the preprocessed data set such that 3-5 entries become 1 and others become 0.

[1]http://www.lboro.ac.uk/gawc/datasets/da6.html
[2]http://www.grouplens.org/node/12

Table 2: Clustering Performance on Synthetic data.

	BP1	BP2	BP3
SCMiner	**1**	**1**	**0.9949**
CA	0.6683	0.7897	0.8750
ITCC	**1**	**1**	0.8750
GS	0.2568	0.4069	0.5493

Jester[3] is a joke rating data set. The original data set contains over 1.7 million continuous ratings (-10.00 to +10.00, +10.00 best) of 150 jokes from 63974 users collected between April 2006 to May 2009. We remove 22 jokes which are never rated or rated by less than 0.01 of users, and randomly pick 1000 users who rate all the picked jokes. Then we binarize the data set such that 5-10 entries become 1 and others become 0. The ground truth is generated for evaluating link prediction, such that the non-negative entries become 1 and the negative entries become 0.

4.1 Clustering Quality

Firstly, we evaluate the quality of clusters detected by *SCMiner*. *SCMiner* can be used as a parameter-free bipartite graph partition method and therefore we choose the approaches Cross-association (*CA*) [1] and Information-theoretic Co-clustering (*ITCC*) [4] as comparison methods. In addition, we compare *SCMiner* to the graph summarization technique (*GS*) of [14]. The algorithms *SCMiner*, *CA* and *GS* are both parameter-free, *ITCC* requires the number of clusters in rows and columns. The algorithm *GS* does not output a clustering, however, the summarized nodes can also be regarded as clusters. We therefore create a cluster for each summarized node of the algorithm. For synthetic data we set the number of clusters of *ITCC* to the true number of the data set. We use the Normalized Mutual Information (NMI) [21] to measure the clustering performance. The value of NMI ranges between 0 and 1. The higher the value the better the clustering. We further report the Adjusted Mutual Information (AMI) and Adjusted Variation Information (AVI) scores proposed in [21].

Synthetic Data. Table 2 depicts the clustering performance comparison on synthetic data sets evaluated by NMI, which shows that *SCMiner* yields better results than *CA*, *ITCC* and *GS*. For BP1 and BP2 both *ITCC* and *SCMiner* give perfect results, however *ITCC* needs the true number of clusters as input parameter, whereas *SCMiner* determines the number of clusters automatically without user input. *CA* outputs worse NMI results, because it splits the clusters into some smaller ones. Worst NMI results yields *GS*, this is mainly because *GS* is designed for summarization but not for clustering. For space limitations, Table 2 reports only NMI but the other scores are similar.

For *SCMiner* we try out several ϵ and choose the best results with the minimum MDL. Figure 3 shows the clustering results for all synthetic data sets with different ϵ and one can see that the results are quite stable on a wide range of ϵ. We also test the relationship between NMI and MDL, for space limitation we only mark several MDL values for synthetic data BP3 in Figure 3. The coding costs are 127102 bits when $\epsilon = 0.01$ and $NMI = 0.89$, and it costs 123723 bits

[3]http://eigentaste.berkeley.edu/dataset/

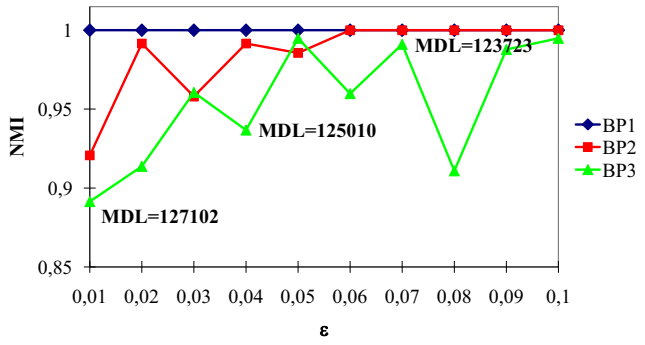


Figure 3: Results for Various ϵ.

when $\epsilon = 0.07$ and $NMI = 0.99$, which proofs that smaller MDL values lead to better NMI results.

World Cities Data. The World cities real data set contains cluster labels of global firms and thus we can use NMI to evaluate clustering quality of this type of nodes. We run *SCMiner*, *CA*, *ITCC* and *GS* on this data set. *SCMiner*, *CA* and *GS* are all parameter-free methods (the chosen ϵ of *SCMiner* for cities is 0.01). We set the true number of clusters of global firms and 4 as the number of clusters of world cities as input parameter for *ITCC*. The NMI results for global firms are depicted in Table 3 and the table shows that *SCMiner* clearly outperforms *CA*, *ITCC* and *GS* in all clustering quality scores. The evaluation of cities type clusters is much more difficult, since the cluster labels are not provided along with the data set. Looking in detail at the cluster contents, *SCMiner* finds 3 clusters and a separated city Washington, DC. The first cluster contains 13 cities, including Atlanta, Boston, Dallas, Munich, Montreal etc. and all these cities are strong in economics. In detail, all 5 accountancy firms have services in these cities, and there are 3 advertising, 5 banking and 2 law companies in average owning offices in these 13 cities. The second cluster contains 17 cities, including Toronto, Paris, London, New York, and Beijing. All these cities are metropolis in the world. Moreover most of these cities are capital of their country. In terms of service, all 5 accountancy firms have services in these cities, and there are 9 advertising, 11 banking and 7 law companies in average having offices in these 17 cities. The third cluster contains 24 cities, including Amsterdam, Barcelona, Seoul, Shanghai etc., which are all kind of financial cities. In terms of service, all 5 accountancy firms have services in these cities as well, and there are 7 advertising, 8 banking and 2 law companies in average having offices in these 24 cities. The cluster analysis shows that *SCMiner* outputs reasonable clusters regarding the cities type nodes. To sum up, the cities can be categorized into 3 types, capital metropolis, financial cities and economic cities. Capital metropolis are the economic, financial and politic center of a country, therefore all kinds of companies have offices in these cities. Financial cities offer a lot of banking and advertising company services, but lack law services, because they are not politically oriented. Some local manufactories are located in economic cities, whereas advertising, banking and law companies are not. Washington, DC is quite different compared to the other cities, since it is a politic, but not a financial city and therefore owns lots of law firms' offices but fewer advertising and banking firms' offices. Thus it is reasonable that this city is separated in its own cluster.

Table 3: Results on World Cities Data.

	NMI	AMI	AVI
SCMiner	**0.4345**	**0.3807**	**0.3824**
CA	0.3109	0.2447	0.2604
ITCC	0.2522	0.1845	0.1891
GS	0.2515	0.0467	0.0683

Movielens Data. MovieLens data set does not provide cluster labels, however, it provides movie categories and personal information of users that can be used to analyze the cluster contents. We separately perform *SCMiner*, *CA*, *ITCC* and *GS* on this data set, *SCMiner*, *CA* and *GS* are both parameter-free (the chosen ϵ of *SCMiner* for MovieLens is 0.05). *CA* outputs 9 user clusters and 8 movie clusters. In addition some isolated nodes and small clusters with less than 5 instances are found. *SCMiner* gives 10 user clusters and 15 movie clusters, whereas all the clusters found by *GS* are single nodes or just contain a few nodes (< 5). To be fair, we set the number of user and movie clusters to 10 and 15 for *ITCC*, which corresponds to the number of clusters found by *SCMiner*. In this data set, movies are classified into 19 genres, such as action, adventure, animation, etc, and a movie can be categorized into several genres at once. According to the basic concept of clustering that objects in the same cluster are similar, we define purity P to evaluate the movie clusters. By counting genres of movies, we can acquire the genre that dominates the cluster and let this genre be the cluster representative. The purity of a cluster is defined as the percentage of movies belonging to the most represented genre. We calculate the purity P for each movie cluster detected by *SCMiner*, *CA* and *ITCC*, and sort them by p value, which is shown in Figure 4. The figure shows that *SCMiner* outputs more pure movie clusters than the two comparison methods. In terms of user clusters, it is difficult to compare the results. Analyzing the content of user clusters detected by *SCMiner*, reveals that some clusters contain old users, some are composed of young users, some groups contain only males, while others own mainly women. This reflects that *SCMiner* outputs meaningful user clusters.

4.2 Hidden Structure

For bipartite graph data, we do only want to analyze the clusters in each type, but we also want to evaluate the relationship between clusters of different types. Besides our *SCMiner* algorithm, *CA* is the only comparison method providing information about these relationships (in the form of a re-arranged adjacency matrix) to users.

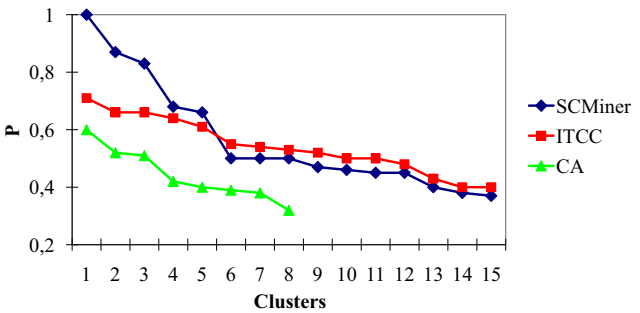


Figure 4: Cluster Purity on Movielens Data.

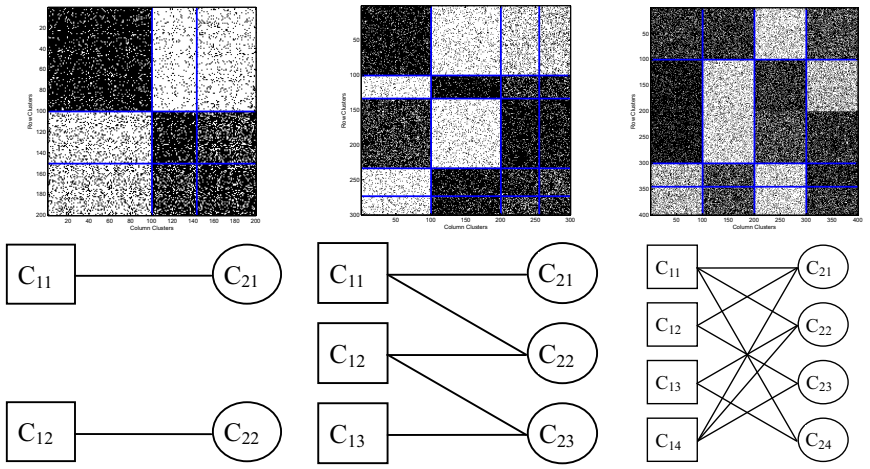


Figure 5: Hidden Structure Detected by *SCMiner* and *CA*. From Left to Right: Data Set BP1, BP2, BP3.

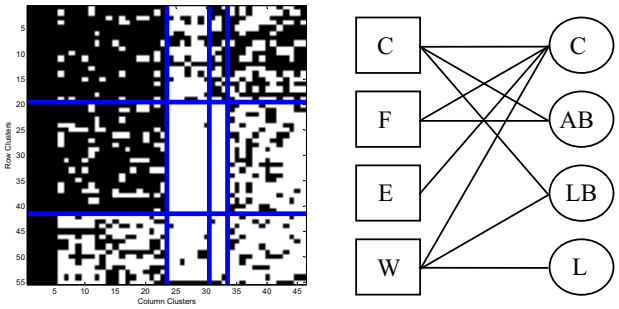


Figure 6: Hidden structure of cities detected by *CA* (left) and *SCMiner* (right). For *SCMiner* results, Square C represents a cluster consisting of capitals, Square F financial cities, Square E economic cities and Square W is the isolated city Washington, DC. On the other side, the cluster represented by Circle C mainly contains accountancy firms, Circle AB advertising and banking companies, Circle LB banking and law firms, and Circle L law firms.

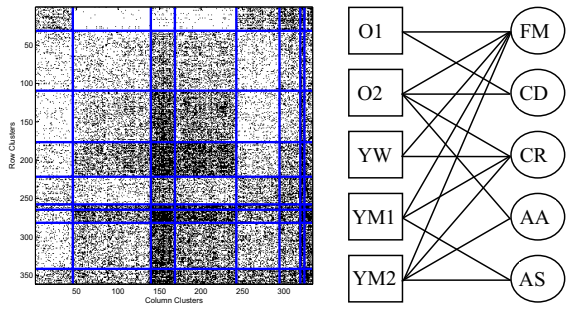


Figure 7: Hidden structure of MovieLens detected by *CA* (left) and *SCMiner* (right). For *SCMiner* results, Square $O1$ and $O2$ denote clusters containing old users, Square YW represents a cluster of young women, and Square $YM1$ and $YM2$ young man. On the other side, the cluster represented by Circle FM represents high scored movies, Circle CD comedy and drama, Circle CR action, romance and comedy. Circles AA and AS represent adventure, action, thriller movies and action, sci-fi, thriller movies, respectively.

Synthetic Data. Figure 5 depicts the hidden structure detected by *SCMiner* and *CA*, the top row shows the re-arranged adjacency matrix of data output by *CA*, the bottom row depicts the hidden structure found by *SCMiner*, where squares and circles denote the two different types of clusters. For BP1 and BP2 *SCMiner* gives perfect results and for BP3 just one node is categorized incorrectly, as cluster C_{12} contains 99 nodes and clusterC_{14} contains 101 nodes, but the whole structure is correct, whereas the cross association output by *CA* for BP3 is incorrect and it is really hard to tell the relationship between the second row and fourth column cluster. In summary, the visualization of cross-associations is only clear and interpretable for data having a relatively easy structure.

World Cities Data. Figure 6 depicts the hidden structure of World cities data set detected by *SCMiner* and *CA*. The left shows the re-arranged adjacency matrix of data output by *CA*. It is hard to identify the relationship between two types of nodes from the matrix itself. The right depicts the hidden structure found by *SCMiner*, the squares represent the cities clusters and circles denote clusters of companies. The figure shows that capital metropolis have connections to all kind of firms, financial cities do not have law companies services, most service in economic cities are from accountancy companies, and Washington, DC owns lots of law companies but fewer banking and advertising firms. From the figure, the major structure of the complex network becomes obvious at first glance. The result of *SCMiner* is a highly compact bipartite graph, a data representation which is easy to understand for the user. In contrast, the re-arranged adjacency matrix is still quite noisy and it is very difficult to infer the structure from this representation.

Movielens Data. Figure 7 depicts the hidden structure of MovieLens data set detected by *SCMiner* and *CA*, the left shows the re-arranged adjacency matrix of data output by *CA*, which is hard to understand, the right depicts the hidden structure found by *SCMiner* which is much easier to analyze. For clarity we only show the relation between the 5 biggest clusters in each type. Squares and circles represent the user and movies cluster. Cluster $O1$ and $O2$ contain user groups that are older than the average regarding the whole data set, and their male-to-female ratio is just about whole

data set average. Square YW represents a user group with a much larger female ratio compared to the average, and the users are younger than the average as well. Square $YM1$ and $YM2$ denote clusters with groups of users containing almost all young man. On the other side, circle FM represents a cluster of famous movies, containing Schindler's List, Shawshank Redemption, and Seven etc. that all have high rating scores in IMDB, which shows that all groups of users like it. This fact is embodied in the revealed hidden structure, since this cluster has connections to all user groups. Circle CD is mainly consisting of movies from comedy and drama category, so it makes sense that older people like it. The movies in circle CR are mostly from action, romance and comedy genre, so young women and young men like it. Circle AA and AS represent adventure action thriller movies and action sci-fi thriller movies respectively, therefore young man like them. The above analysis shows that the hidden structure found by *SCMiner* is reasonable.

4.3 Link Prediction Accuracy

SCMiner outputs the summarization of the input graph that yields the minimum coding cost. The summarization itself consists of a summary graph and an additional graph, where the "-" edges in the additional graph denote missing links which should be added to the original graph, and "+" edges might be noisy links which should be removed from the original graph. Since it is hard to determine whether a "-" edge represents noise, we only use link prediction to evaluate whether a "+" edge is a missing link.

The link prediction problem has been studied on graphs by several approaches which can be divided into two categories, supervised, e.g. [11], [6] and unsupervised methods, e.g. [10]. Since *SCMiner* is unsupervised, we only compare our results to unsupervised approaches. There are two types of unsupervised methods: based on node neighbors and based on paths between nodes. However, some of these methods, like common neighbor, are not suitable for link prediction on bipartite graph data and therefore we choose two methods, *PA* preferential attachment [15] and *Katz* [8], that are suitable for bipartite graphs. We use precision and recall to evaluate the accuracy. *SCMiner* automatically outputs K predicted edges and for reasons of fairness we choose the top K predictions of the ranking list of *PA* and *Katz* to compare the precision and recall of the prediction results. We also compare our approach to *GS* Graph Summarization, since the algorithm changes the edges of the original graph as well. Table 4 shows the link prediction comparison on our synthetic data sets as well as on the real data set Jester. All the results are averaged over runs on 10 randomly generated or the selected data sets. *SCMiner* predicts the missing links of BP1 and BP2 completely and those of BP3 nearly completely correct and therefore yields better results on all synthetic data sets than all comparison methods. Moreover it outperforms the other approaches on the Jester data set. Interestingly, *GS* performs better on the synthetic data sets than on Jester. The reason might be that the Jester data set is sparser than the synthetic data sets, and therefore *GS* reaches a local minimum very fast, which results in fewer predicted edges.

5. RELATED WORK

During past decades, many algorithms were proposed for graph clustering, graph compression, graph summarization and link prediction. Due to space limitations, we only provide a very brief survey on some related research directions.

Co-clustering. Bi-clustering or co-clustering is a creative approach which simultaneously clusters rows and columns of a data matrix. It avoids the problems caused by sparseness and high-dimensionality that traditional one way clustering algorithms suffer from. There are some state-of-the-art co-clustering algorithms, such as Information- theoretic Co-clustering [4], Cross Association [1] etc. Specifically, Information theoretic co-clustering [4] simultaneously maps row elements to row clusters and column elements to column clusters, mutual information of each clustering state is calculated and compared to the initial state. Thus, the optimal co-clustering result is obtained when the mutual information loss is minimal. However, the drawback of the method is that the number of both row and column clusters must be predetermined. Cross Association [1] is a MDL-based parameter-free co-clustering method that processes a binary matrix and seeks clusters of rows and columns alternately. Then the matrix is divided into homogeneous rectangles which represent underlying structure of the data. However, compared with our algorithm, it only addresses the clustering problem. Moreover, Long et al. [13] proposed a framework for co-clustering named block value decomposition (BVD), which formulates co-clustering as an optimization problem of matrix decomposition. Similarly, in [12], Long et al. reconstruct a bipartite graph based on some hidden nodes and thus an optimal co-clustering result is obtained from the new bipartite graph which mostly approximates the origin graph. However, these two algorithms both need the number of clusters as input parameter. Besides, Dhillon [3] proposed a spectral algorithm for bipartite graph partition. Shan and Banerjee [17] proposed a Bayesian co-clustering model and considered co-clustering as a generative mixture modeling problem. George and Merugu proposed a co-clustering algorithm based on the collaborative filtering framework [5]. In terms of real life data sets, [2] applied co-clustering on gene expression data and [3] effectively processed documents and words data.

Graph Compression and Summarization. In real life, it is common that a graph consists of tens of thousands nodes and complex linkages. Graph compression and graph summarization [14, 20, 22] are effective ways to simplify the original large graph and to extract useful information. Many algorithms are based on Subdue [7], which is a classical graph compression system. It makes use of the MDL Principle to find a subgraph suitable for compression. Bayesian Model Merging (BMM) [18, 19] also is a kind of compression method which is based on Bayesian formula. The best compressed model can be achieved by maximizing the posterior probability. Navlakha et al. [14] represented a graph by a summary graph and a correction matrix, and summarization is implemented by greedy merging and randomized algorithms. MDL is used to find the local optimum. However, the algorithm is designed for compressing only and performs poorly in clustering or link prediction. In [20], according to group nodes based on their attributes and relationships, which are selected by the user, Tian et al. propose a graph summarization algorithm.

Link Prediction. Predicting whether two disconnected nodes will be connected in the future, based on available network information is known as the challenge of link prediction. Liben-Nowell and Kleinberg [10] summarize some

Table 4: Link Prediction Performances.

Algorithm	BP1		BP2		BP3		Jester	
	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
PA	0.084	0.084	0.436	0.436	0.442	0.443	0.406	0.369
Katz	0.984	0.984	0.943	0.943	0.547	0.548	0.406	0.369
GS	0.965	**1**	0.981	1	0.994	0.973	0.356	0.05
SCMiner	**1**	**1**	**1**	**1**	**0.997**	**1**	**0.431**	**0.389**

unsupervised link prediction approaches for social networks. For example, common neighbor, Jaccard's coefficient, preferential attachment [15], *Katz* [8] etc. Specifically, preferential attachment is based on the idea that two nodes with higher degree have higher probability to be connected. *Katz* [8] defines a score that sums up the number of all paths between two nodes where short paths are weighted stronger. By ranking these scores, the probability of whether two disconnected nodes will be linked in the future can be acquired. However, when dealing with bipartite graphs, Kunegis et al. [9] stated that scores based on common neighbor are not suitable, because two connected nodes do not have any common neighbors in a bipartite graph. But scores like preferential attachment and *Katz*, which are used in this paper as comparison methods are reasonable.

6. CONCLUSION

In this paper, we propose *SCMiner*, a technique which integrates graph summarization, bipartite graph clustering, link prediction and hidden structure mining. Based on the sound optimization goal of data compression, our algorithm finds a compact summary representation of a large graph. In addition, while compressing the graph *SCMiner* detects the truly relevant clusters of both node types and their hidden relationships. Thus, *SCMiner* is a framework comprehensively supporting the major tasks in unsupervised graph mining. Our experiments have demonstrated that *SCMiner* outperforms specialized state-of-the-art techniques for co-clustering and link prediction. In ongoing and future work we want to extend this idea to support multi-partite graphs as well as graphs with different edge types.

Acknowledgments

Jing Feng and Xiao He are supported by the China Scholarship Council (CSC). Claudia Plant is supported by the Alexander von Humboldt Foundation.

7. REFERENCES

[1] D. Chakrabarti, S. Papadimitriou, D. S. Modha, and C. Faloutsos. Fully automatic cross-associations. In *KDD*, pages 79–88, 2004.

[2] H. Cho, I. S. Dhillon, Y. Guan, and S. Sra. Minimum sum-squared residue co-clustering of gene expression data. In *SDM*, 2004.

[3] I. S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *KDD*, pages 269–274, 2001.

[4] I. S. Dhillon, S. Mallela, and D. S. Modha. Information-theoretic co-clustering. In *KDD*, pages 89–98, 2003.

[5] T. George and S. Merugu. A scalable collaborative filtering framework based on co-clustering. In *ICDM*, pages 625–628, 2005.

[6] M. A. Hasan, V. Chaoji, S. Salem, and M. Zaki. Link prediction using supervised learning. In *Proc. of SDM Workshop on Link Analysis*, 2006.

[7] L. B. Holder, D. J. Cook, and S. Djoko. Substucture discovery in the subdue system. In *KDD Workshop*, pages 169–180, 1994.

[8] L. Katz. A new status index derived from sociometric analysis. *PSYCHOMETRIKA*, 18(1):39–43, 1953.

[9] J. Kunegis, E. W. D. Luca, and S. Albayrak. The link prediction problem in bipartite networks. *CoRR*, abs/1006.5367, 2010.

[10] D. Liben-Nowell and J. M. Kleinberg. The link prediction problem for social networks. In *CIKM*, pages 556–559, 2003.

[11] R. Lichtenwalter, J. T. Lussier, and N. V. Chawla. New perspectives and methods in link prediction. In *KDD*, pages 243–252, 2010.

[12] B. Long, X. Wu, Z. M. Zhang, P. S. Yu, and P. S. Yu. Unsupervised learning on k-partite graphs. In *KDD*, pages 317–326, 2006.

[13] B. Long, Z. M. Zhang, P. S. Yu, and P. S. Yu. Co-clustering by block value decomposition. In *KDD*, pages 635–640, 2005.

[14] S. Navlakha, R. Rastogi, and N. Shrivastava. Graph summarization with bounded error. In *SIGMOD*, pages 419–432, 2008.

[15] M. E. J. Newman. Clustering and preferential attachment in growing networks. *PHYS.REV.E*, 64:025102, 2001.

[16] J. Rissanen. *Information and Complexity in Statistical Modeling*. Springer, 2007.

[17] H. Shan and A. Banerjee. Bayesian co-clustering. In *ICDM*, pages 530–539, 2008.

[18] A. Stolcke and S. M. Omohundro. Hidden markov model induction by bayesian model merging. In *NIPS*, pages 11–18, 1992.

[19] A. Stolcke and S. M. Omohundro. Inducing probabilistic grammars by bayesian model merging. In *ICGI*, pages 106–118, 1994.

[20] Y. Tian, R. A. Hankins, and J. M. Patel. Efficient aggregation for graph summarization. In *SIGMOD*, pages 567–580, 2008.

[21] N. X. Vinh, J. Epps, and J. Bailey. Information theoretic measures for clusterings comparison: is a correction for chance necessary? In *ICML*, pages 1073–1080, 2009.

[22] N. Zhang, Y. Tian, and J. M. Patel. Discovery-driven graph summarization. In *ICDE*, pages 880–891, 2010.