PICS: Parameter-free Identification of Cohesive Subgroups in Large Attributed Graphs

Leman Akoglu* Hanghang Tong† Brendan Meeder* Christos Faloutsos*

Abstract

Given a graph with node attributes, how can we find meaningful patterns such as clusters, bridges, and outliers? Attributed graphs appear in real world in the form of social networks with user interests, gene interaction networks with gene expression information, phone call networks with customer demographics, and many others. In effect, we want to group the nodes into clusters with similar connectivity and homogeneous attributes. Most existing graph clustering algorithms either consider only the connectivity structure of the graph and ignore the node attributes, or require several user-defined parameters such as the number of clusters. We propose PICS, a novel, parameter-free method for mining *attributed graphs*. Two key advantages of our method are that (1) it requires *no* user-specified parameters such as the number of clusters and similarity functions, and (2) its running time scales *linearly* with total graph and attribute size. Our experiments show that PICS reveals meaningful and insightful patterns and outliers in both synthetic and real datasets, including call networks, political books, political blogs, and collections from Twitter and YouTube which have more than 70K nodes and 30K attributes.

1 Introduction

Real graphs often have nodes with attributes, in addition to connectivity information. For example, social networks contain both the friendship relations as well as user attributes such as interests and demographics. Both types of information can be described by a graph in which nodes represent the objects, edges represent the relations between them, and attribute vectors associated with the nodes represent their attributes. Such graph data is often referred to as an *attributed graph*.

Given such an attributed graph how can we find meaningful patterns, clusters of nodes, clusters of attributes, and anomalies? For example, consider the case of YouTube in which graph nodes represent users and YouTube-group memberships represent attributes. Given the who-friends-whom and who-belongs-to-which YouTube groups information, how can we summarize and make sense out of it?

We propose PICS for mining attributed graphs. Specifically, PICS finds *cohesive clusters* of nodes that have similar connectivity patterns and exhibit high attribute homogeneity. PICS additionally groups the node attributes into attribute-clusters. These patterns further help us to spot anomalies and bridges. A key characteristic of PICS is that it is *parameter-free*; it can recover the number of necessary clusters without any user intervention. For example, our algorithm automatically finds coherent user clusters and YouTube-group clusters as shown in Figure 1. As a first observation, the algorithm automatically discovers clusters of users who like the 'anime' genre and form a so-called 'core and periphery' pattern: the core consists of the users in the bottom-right dark square of the blue square labeled as '2'; also notice that the anime fans overwhelmingly belong to a coherent set of YouTube-groups (blue square labeled as 'A2'). As we show in Table 3, those groups include *anime4ever*, *narutoholics*, and *crazyforanime*. More discussion on YouTube results is given in experiments, §4.2.1.

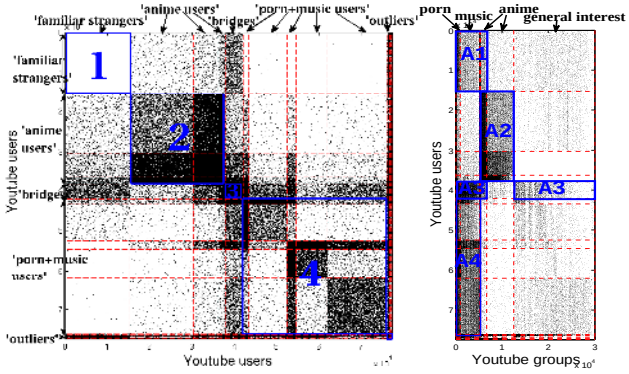


Figure 1: PICS *on* YOUTUBE *finds clusters of users with similar connectivity and attribute coherence. Left: the adjacency matrix (users-to-users); right: the attribute matrix (users-to-YouTubeGroups). Both matrices are carefully arranged by* PICS*, revealing patterns: e.g., the anime fans are heavily connected, and they focus on the same YouTube-groups. See §4.2.1.*

The contributions of this paper are the following:

1. *Algorithm design:* We introduce PICS, a novel clustering method to summarize graphs with node attributes. In effect, it groups the nodes with

*Carnegie Mellon University, Computer Science Department. {lakoglu, bmeeder, christos}@cs.cmu.edu
†IBM T. J. Watson Research. htong@us.ibm.com

similar connectivity patterns into *cohesive clusters* that have high attribute homogeneity. Besides, it also clusters the attributes into attribute-clusters.

2. *Automation:* `PICS` *does not* require any user-specified input such as the number of clusters, similarity functions or any sort of thresholds.
3. *Scalability:* The run time of `PICS` grows *linearly* with the total input graph and attribute size.
4. *Effectiveness:* We evaluate our method on a diverse collection of real data sets with thousands of nodes and attributes. Our results show that `PICS` successfully recovers cohesive node clusters, reveals the bridge and outlier nodes, and groups attributes into meaningful clusters.

2 Related Work

We first highlight the main challenges associated with simply extending traditional clustering algorithms to solve our problem, followed by related work and comparison. An overview is depicted in Table 1.

2.1 Why not simple extensions?

E1 Represent the relations of the objects as additional attributes, and then perform flat clustering on this extended feature space. The challenges with this approach are two-fold: First, it leads to a very large number of features and thus is faced with the curse of dimensionality [3]. Second, it yields two separate types of attributes where it is not clear how to weigh those different sets for clustering.

E2 Represent the attributes of the objects as additional nodes in the graph, introduce new edges from the original nodes to the attribute nodes they exhibit, and then perform graph clustering on this extended graph. Again, there are two challenges with this approach: First, the graph grows with new nodes and edges, which may be quite numerous. Additionally, it is not clear how to do clustering in this heterogeneous graph which contains two types of nodes and edges.

E3 Introduce edges between all pairs of nodes where edges are weighted by the existence of connectivity and attribute similarity: This approach requires quadratic computation of pair-wise similarities and is thus untractable for large graphs. It also requires a careful choice of a similarity function.

In summary, clustering attributed graphs by applying traditional clustering approaches presents several non-trivial challenges.

2.2 Clustering graphs Graph partitioning has been well studied in the literature. The top-performing methods include METIS [17] and spectral clustering [22]. However, these as well as many other graph partitioning methods [2, 10, 11] work with the connectivity structure of the graph and cannot be directly applied to attributed graphs. More importantly, they require the number of partitions and a measure of imbalance between any two partitions as input. These are usually hard for the user to specify, especially for large graphs, and require experimentation to get good results. For example, for spectral partitioning the user needs to choose from several measures such as the ratio cut [6], normalized cut [24], or the min-max cut [8].

The idea of using (lossy) compression for graph clustering is introduced in [7]. The information-theoretic co-clustering algorithm simultaneously clusters rows and the columns of a normalized contingency table which is treated as a two-dimensional joint probability distribution. The algorithm, however, requires the number of row and column clusters as input.

In terms of parameter-free graph clustering algorithms, Autopart [4] and cross-associations [5], and their extension to time-evolving [25] and k-partite graphs [15] are the most representative. These methods use the *minimum description length (MDL)* principle [12] to automatically choose the number of clusters. However, they do not apply to attributed graphs simply because they operate on the adjacency matrix of the graph, and thus consider only the connectivity structure of the graph.

2.3 Clustering attributed graphs Compared to the wide range of work on graph clustering, there has been much less focus on clustering attributed graphs. [14] transforms the graph and the attributes to a combined distance metric and then applies flat clustering. This and other similar methods [26] achieve homogeneity of attributes for the nodes in the same cluster, however they tend to yield low intra-cluster connectivity. [29, 30] transforms the attributes to additional nodes in the graph, where original nodes are linked to attribute nodes if they exhibit the particular attribute. Again, all these methods require the number of clusters to be exclusively specified.

Recently, [21, 13] propose methods to extract cohesive subgraphs from an attributed graph rather than partitioning the entire graph. The subgraphs exhibit high density and homogeneity in a subset of their attributes. In these methods other types of parameters need to be set by the user; these include the subspace dimensionality and density thresholds, as well as the minimum number of nodes in each cluster.

The spectral relational clustering algorithm [19] performs collective factorization of related matrices for multi-type relational data clustering. Although the

Table 1: *Comparison of related work.*

Property	Node Attributes	Graph structure	Parameter-free	Linear scalability
Flat clustering [18, 28] (e.g. k-means)	✓			✓
METIS [17], Spectral [22] and Co-clustering [7]		✓		✓
Spectral relational clustering [19], SA-Cluster [29]	✓	✓		
CoPaM [21], Gamer [13]	✓	✓		?,✓
Autopart [4], Cross-associations [5]		✓	✓	✓
PICS [this paper]	✓	✓	✓	✓

method is applicable to graph data with attributes, the user intervention there is two-fold; besides the number of clusters for each type of nodes, reasonable weights for different type of relations or attributes also need to be specified. In addition, it is not easy to do spectral clustering on directed graphs with attributes (there exists spectral clustering for plain directed graphs, although it is much more complicated than those for un-directed graphs). In contrast, PICS can naturally handle directed graphs.

Despite their success, all existing clustering methods lack one or more of the properties listed in Table 1. PICS is the first to address all at the same time.

3 Proposed Method

3.1 Problem Description In this paper, we address the problem of *finding cohesive clusters* in an attributed graph. Specifically, given a graph with n nodes and their binary connectivity information, where nodes are associated with f binary attributes (interchangeably, features), our goal is to group the nodes into k, and group the features into l disjoint clusters such that the nodes in the same cluster have "similar connectivity" and also exhibit high "feature coherence" (interchangeably we also use the term feature similarity). Informally, a set of nodes have "similar connectivity" if the set of nodes in the graph they connect to "highly" overlap. Similarly, a set of nodes have high "feature coherence" if the set of features they exhibit "highly" overlap.

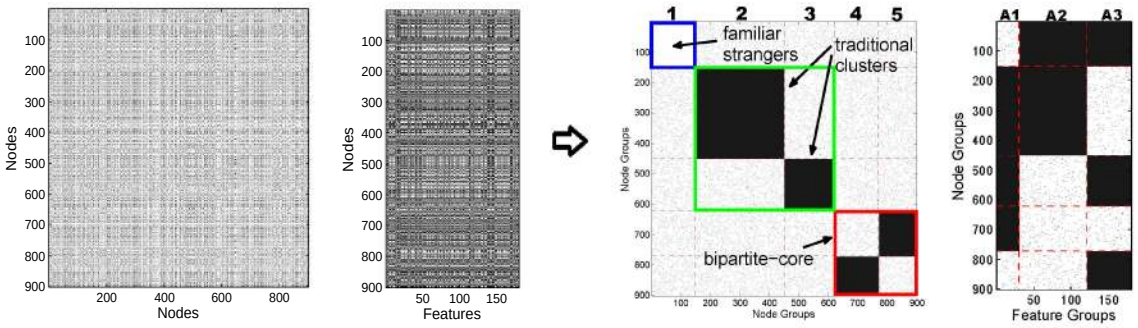
3.1.1 Synthetic Graph Example To elaborate on the terminology, we give an example in Figure 2. PICS detects 5 node-clusters and 3 feature-clusters in this example graph. Notice that the nodes in the same cluster agree on their features to a high extent, i.e. have high feature coherence; for example nodes in node-cluster 1 exhibit features in feature-clusters 2 and 3, nodes in node-cluster 2 exhibit features in feature-clusters 1 and 2, and so on. In addition, notice the nodes in the same cluster having similar connections among themselves as well as to the rest of the graph. For example, nodes in node-cluster 2 (similarly node-cluster 3) are densely connected among themselves but scarcely to the rest, nodes in node-cluster 4 are densely connected to nodes in node-cluster 5 (and vice versa) and scarcely to the rest, and so on.

Note that the nodes in a cluster that PICS finds may not necessarily be densely connected among themselves. In fact, the nodes in node-cluster 1 in the example graph are not connected to each other at all! They, however, share the same set of features and still have similar connectivity to the graph. Simply put, they are "familiar strangers". Similarly, nodes in node-clusters 4 and 5 are not connected within the clusters but across each other, forming a "bipartite core".

We would like to point out that while the traditional graph clustering algorithms would recover node-clusters 2 and 3, PICS can find additional type of clusters such as familiar strangers and bipartite cores, providing a richer analysis of a given graph. This derives from our more general cluster definition of nodes with "similar", rather than "dense", connectivity.

3.2 Problem Formulation The main questions that arise given the above problem definition are: How should we decide the number of node and feature clusters, i.e., k and l, respectively? How can we assign the nodes and the features to their "proper" clusters? How much overlap of the features is "high" enough? In this paper, we address these questions without making the users have to set any parameters such as the number of clusters, feature similarity thresholds or make them have to choose from a large collection of similarity functions. In fact, automation is exactly one of the main contributions of our approach.

Our starting point is data compression. Specifically, we want to compress *two*, *inter-related* matrices *simultaneously*. The first matrix is the $n \times n$ binary connectivity matrix $\mathbf{A}$, and the second is the $n \times f$ binary feature matrix $\mathbf{F}$ (note that although we consider binary graphs in



(a) Original **A** and **F** (b) PICS **A** and **F**

Figure 2: PICS *on a synthetic dataset with* n=900 *nodes and* f=180 *features. Notice* 5 *node and* 3 *feature clusters. Nodes in the same cluster exhibit high feature homogeneity and have similar connectivity patterns. Note that similar connectivity does not only imply but also includes dense connectivity; while node-clusters 2 and 3 are densely connected within the clusters, node-clusters 1, 4, and 5 are not. See §3.1 for more details.*

this paper, similar ideas can also be applied to weighted graphs). Our goal then is to compress (conceptually, summarize) these matrices simultaneously by looking at partitions/clusters/groups (i.e. homogeneous, rectangular "blocks") of both low and high densities. The reason for operating on the matrices simultaneously is simply that the two matrices are inter-related: the node groups should be homogeneous in both the connectivity matrix **A** *as well as* in the feature matrix **F**.

The natural question is, how many rectangular blocks should we have in each matrix **A** and **F**? To compress these matrices efficiently, we need to have several highly homogeneous blocks. On the other hand, having more clusters allows us to obtain more homogeneous blocks (at the very extreme, we can have $n \times n + n \times f$ blocks, each having perfect homogeneity of 0 or 1). The best compression model should achieve a proper trade-off between these two factors of homogeneity (data description complexity) and the number of blocks (model description complexity). To achieve this goal we use the MDL principle [12], a model selection criterion based on lossless compression principles –we design a cost criterion that we aim to minimize in which costs are based on the number of bits required to transmit both the "summary" of the structure (model) as well as each rectangular block (data) within the structure.

3.2.1 Notation Let k denote the desired number of disjoint node-clusters and let l denote the desired number of disjoint feature-clusters. Let $R : \{1, 2, \ldots, n\} \rightarrow \{1, 2, \ldots, k\}$ and $C : \{1, 2, \ldots, f\} \rightarrow \{1, 2, \ldots, l\}$ denote the assignments of nodes to node-clusters and features to feature-clusters, respectively. We will refer to (R, C) as a *mapping*. To better understand a mapping, given the node-clusters and feature-clusters, let us rearrange the rows and columns of the connectivity matrix **A** such that all rows corresponding to node-cluster-1 are listed first, followed by rows in node-cluster-2, and so on. We also rearrange the columns in the same fashion. Note that the row and column arrangements for **A** will be the same. One can imagine that such a rearrangement sub-divides the matrix **A** into k^2 two-dimensional, rectangular blocks, which we will refer to as B^A_{ij}, $i, j = 1, \ldots, k$.

Similarly, we can rearrange the rows of the feature matrix **F** such that all rows corresponding to node-cluster-1 are listed first, followed by rows in node-cluster-2, and so on. One can realize that the row arrangements of **A** and **F** matrices will be the same. In a manner different than for **A**, we will rearrange the columns of the feature matrix **F** such that all columns corresponding to feature-cluster-1 are listed first, followed by columns in feature-cluster-2, and so on. As a result, we will obtain rectangular blocks denoted as B^F_{ij}, $i = 1, \ldots, k$ and $j = 1 \ldots, l$ for **F**. Finally, let the dimensions of B^F_{ij} and B^A_{ij} be (r_i, c_j) and (r_i, r_j), respectively.

3.3 Objective Function Formulation We now describe a two-part cost criterion for the (lossless) compression of the connectivity and feature matrices **A** and **F**. The compression cost can be thought of as the total number of bits required to transmit these matrices over a network channel. The first part is the model description cost that consists of describing the mapping (R, C). The second part is the data description cost that consists of encoding the sub-matrices (i.e., the "blocks" B_{ij}), given the mapping. Intuitively, a good choice of (R, C) would simultaneously compress **A** and **F** well,

and as a result yield a low total description cost.

Next, we describe those two parts in more detail and then give the total encoding cost (objective function).

3.3.1 Model Description Cost This part consists of encoding the number of node and feature clusters as well as the corresponding mapping.

- The number of nodes n and the number of features f (i.e., matrix dimensions) require $\log^\star n + \log^\star f$ bits, where $\log^\star$ is the universal code length for integers [23]. This term is independent of any particular mapping.
- The number of node and feature clusters (k, l) require $\log^\star k + \log^\star l$ bits.
- The node and feature cluster assignments with arithmetic coding require $nH(P) + fH(Q)$ bits, where H denotes the Shannon entropy function, P is a multinomial random variable with the probability $p_i = \frac{r_i}{n}$ and r_i is the size of the i-th node cluster, $1 \le i \le k$. Similarly, Q is another multinomial random variable with the probability $q_j = \frac{c_j}{f}$ and c_j is the size of the j-th feature cluster, $1 \le j \le l$.

3.3.2 Data Description Cost This part consists of encoding the matrix blocks.

- For each block B^A_{ij}, $i, j = 1, \ldots, k$ and B^F_{ij}, $i = 1, \ldots, k$, $j = 1, \ldots, l$, $n_1(B_{ij})$, that is the number of 1s in the sub-matrix, requires $\log^\star n_1(B_{ij})$ bits.
- Having encoded the summary information about the rectangular blocks, we next encode the actual blocks B_{ij}. We can calculate the density $P_{ij}(1)$ of 1s in B_{ij} using the description code above as $P_{ij}(1) = n_1(B_{ij})/n(B_{ij})$, where $n(B_{ij}) = n_1(B_{ij}) + n_0(B_{ij}) = r_i c_j$ for $\mathbf{F}$ blocks ($r_i r_j$ for $\mathbf{A}$ blocks), and $n_1(B_{ij})$ and $n_0(B_{ij})$ are the number of 1s and 0s in B_{ij}, respectively. Then, the number of bits required to encode each block using arithmetic coding is

$$\begin{aligned} E(B_{ij}) &= -n_1(B_{ij}) \log_2(P_{ij}(1)) - n_0(B_{ij}) \log_2(P_{ij}(0)) \\ &= n(B_{ij}) H(P_{ij}(1)). \end{aligned}$$

3.3.3 Total Encoding Cost (Length in bits)

$$\begin{aligned} L(\mathbf{A}, \mathbf{F}; R, C) = \log^\star n + \log^\star f + \log^\star k + \log^\star l \\ - \sum_{i=1}^{k} r_i \log_2(\frac{r_i}{n}) - \sum_{j=1}^{l} c_j \log_2(\frac{c_j}{f}) \\ + \sum_{i=1}^{k} \sum_{j=1}^{l} \left(\log^\star n_1(B^F_{ij}) + E(B^F_{ij}) \right) \\ + \sum_{i=1}^{k} \sum_{j=1}^{k} \left(\log^\star n_1(B^A_{ij}) + E(B^A_{ij}) \right). \end{aligned}$$

3.4 Proposed Algorithm PICS The total encoding cost $L(\mathbf{A}, \mathbf{F}; R, C)$ can point out the best model with the minimum cost among many. It does not, however, tell us how to find the best model. In fact, finding the optimal solution that would minimize our cost function is an NP-hard task, since even allowing only column-reordering for a single matrix, a reduction to the traveling salesman problem can be found [16]. As a result, we resort to a greedy iterative heuristic solution. Our experiments show that the proposed heuristic algorithm PICSperforms quite well in practice for the real data sets used. The pseudo-code[1] of PICS is given in Algorithm 1.

Algorithm 1 PICS

Input: $n \times n$ link matrix $\mathbf{A}$, $n \times f$ feature matrix $\mathbf{F}$
Output: A heuristic solution towards minimizing total encoding $L(\mathbf{A}, \mathbf{F}; R, C)$: number of row and column groups (k^*, l^*), associated mapping (R^*, C^*)

1: Set $k^0 = l^0 = 1$ as we start with a single node and feature cluster.
2: Set $R^0 := \{1, 2, \ldots, n\} \to \{1, 1, \ldots, 1\}$
3: Set $C^0 := \{1, 2, \ldots, f\} \to \{1, 1, \ldots, 1\}$
4: Let T denote the outer iteration index. Set $T = 0$.
5: **repeat**
6: $C^{T+1}, l^{T+1} :=$ Split-FeatureGroup($\mathbf{F}$, C^T, l^T)
7: $(R^{T+1}, C^{T+1}) :=$ Shuffle($\mathbf{A}$, $\mathbf{F}$, (R^T, C^{T+1}), (k^T, l^{T+1}))
8: $R^{T+1}, k^{T+1} :=$ Split-NodeGroup($\mathbf{A}$, $\mathbf{F}$, (R^{T+1}, C^{T+1}), (k^T, l^{T+1}))
9: $(R^{T+1}, C^{T+1}) :=$ Shuffle($\mathbf{A}$, $\mathbf{F}$, (R^{T+1}, C^{T+1}), (k^{T+1}, l^{T+1}))
10: **if** $L(\mathbf{A}, \mathbf{F}; R^{T+1}, C^{T+1}) \ge L(\mathbf{A}, \mathbf{F}; R^T, C^T)$ **then**
11: **return** $(k^*, l^*) = (k^T, l^T)$, $(R^*, C^*) = (R^T, C^T)$
12: **else**
13: Set $T = T + 1$
14: **end if**
15: **until** convergence

PICS starts with a single node and a single feature cluster (Lines 1-3) and iterates between two steps. At each iteration, it first tries to increase the number of feature clusters l by 1, by *splitting* the feature cluster with the maximum entropy per feature into two clusters (Line 6). Then, it *shuffles* the rows and columns of $\mathbf{A}$ and $\mathbf{F}$ such that the new ordering (mapping) yields a lower total encoding cost for the current number of clusters (k, l) (Line 7). Next, it tries to increase the number of node clusters k by 1, followed by another shuffle step (Lines 8 and 9, respectively). The algorithm halts when the total cost can not be reduced any further.

[1] Source code of PICS: www.cs.cmu.edu/~lakoglu/#pics

The implementation details for the procedures `Split-FeatureGroup`, `Split-NodeGroup` and `Shuffle` are given in pseudo-code in the Appendix, § 6.

3.4.1 Synthetic Graph Example In Figure 3, we show the step-by-step operation of `PICS` on the synthetic dataset in Figure 2. In Fig. 3(a), we see the output $\mathbf{A}$ and $\mathbf{F}$ after `Split-FeatureGroup` and `Shuffle` (Steps 6 and 7) are executed on the original $\mathbf{A}$ and $\mathbf{F}$ shown in Fig. 2(a). Here, `Split-FeatureGroup` increases the number of feature groups to 2, and then `Shuffle` reorders the rows and columns in both matrices. Next, `Split-NodeGroup` increases the number of row groups to 2 (Step 8) and `Shuffle` reorders the rows and columns that yields a lower encoding cost (Step 9). This is also visually clear in Fig. 3(b). `PICS` repeats the same steps in Fig. 3(c) and (d). Notice that `Split-FeatureGroup` cannot increase the number of feature groups above 3, and thus `Shuffle` is called only after `Split-NodeGroup` in Fig. 3(e) and (f), after which `Split-NodeGroup` also stops finding new node groups for reduced cost and the algorithm converges.

3.5 Convergence The stopping criterion for `Shuffle` is satisfied if and only if the total encoding cost cannot be reduced any further by the new ordering. Therefore, lines 7 and 9 in Algorithm 1 decreases the objective criterion $L(\mathbf{A}, \mathbf{F}; R, C)$. Since the objective criterion, i.e. the total encoding cost in bits, has the lower bound zero and the number of node and feature clusters have respective upper bounds n and f, the algorithm is guaranteed to converge.

3.6 Computational Complexity The computationally most demanding component of `PICS` is the `Shuffle` procedure, which takes as input the $\mathbf{A}$ and $\mathbf{F}$ matrices and the number of clusters (k, l), and finds a new ordering of the rows and columns that gives a lower encoding cost. It achieves this goal by iterating between two steps: (1) shuffling the columns in $\mathbf{F}$, and (2) shuffling the rows in $\mathbf{A}$ and $\mathbf{F}$, simultaneously.

One iteration of step (1) above is $O(n_1(F) * l)$, where $n_1(F)$ denotes the number of non-zeros in $\mathbf{F}$, as we access each column and compute its number of non-zeros and consider l possible feature groups to place it into. Similarly, one iteration of step (2) is $O((n_1(F)+2n_1(A))*k)$. Therefore, the total complexity of `Shuffle` is $O([2n_1(A)k + n_1(F)(k + l)] * t)$, where t is the total number of inner iterations for `Shuffle`.

`PICS` calls `Shuffle` two times at each outer iteration T (Lines 7 and 9 in Algorithm 1). T is equal to $\max(k^*, l^*)$. Thus, the overall complexity of `PICS` is $O(\max(k^*, l^*) * [2n_1(A)k^* + n_1(F)(k^* + l^*)] * \hat{t})$, where $\hat{t}$ denotes the maximum number of inner iterations. Note that `PICS` scales linearly with respect to the total number of non-zeros in $\mathbf{A}$ and $\mathbf{F}$, the total number of `Shuffle` iterations $\hat{t}$ (typically $\hat{t} \ll n_1(A) + n_1(F)$), and quadratically with respect to the number of clusters (k^*, l^*), where k^* and l^* are usually small.

4 Empirical Study

4.1 Datasets In our experiments we studied six real-world datasets from various domains which we describe next. A summary is given in Table 2. In each case we are able to use `PICS` to automatically discover interesting node and attribute structures, with further investigation providing explanations for their presence.

`YOUTUBE` data consists of users and Youtube-groups. The links represent user-user friendships and node attributes are the group memberships of each user. The data is compiled by [20] in 2007.

`TWITTER` graph contains early-adopter users, with links representing the who-mentioned-whom interactions between them. The attributes are the hashtags (tokens starting with a #) that the users used in their messages (Tweets).

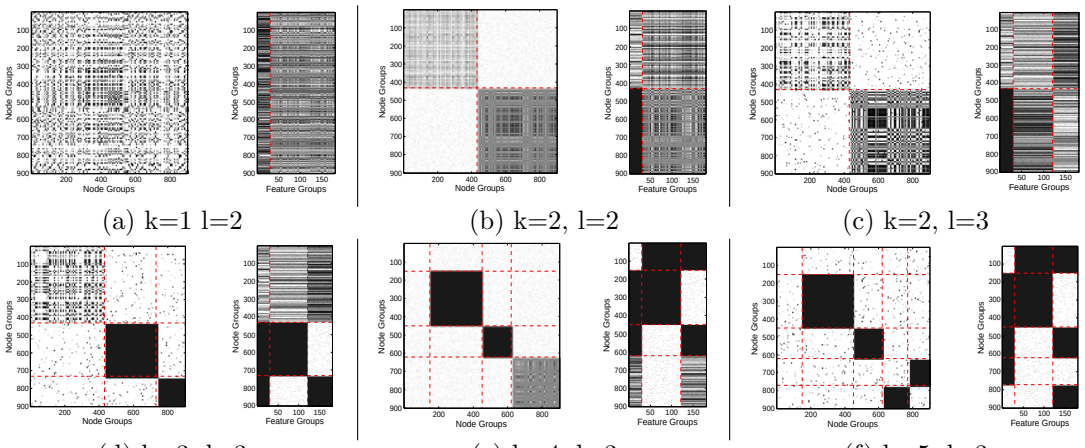
The `PHONECALL` and `DEVICE` graphs are constructed using the Reality Mining data sets provided by the MIT Media Lab [9]. The Reality Mining project was conducted in 2004-05 with 94 human subjects using mobile phones pre-installed with special software that recorded data. `PHONECALL` is built using the call logs, and represents the phone-call interactions between the subjects. `DEVICE` is constructed using Bluetooth device scans; an edge from i to j indicates person i's device discovered person j's device within five meters proximity. The affiliations of the subjects (e.g. grad, undergrad, business student) are the node attributes.

`POLBOOKS` is a graph of books about U.S. politics published during 2004 presidential election [27]; an edge from i to j indicates that book i was frequently co-purchased with book j by the same buyers. Similarly, `POLBLOGS` is a directed graph of hyperlinks between weblogs on U.S. politics, compiled by [1] in 2005. In both graphs, the nodes (books or blogs) are attributed as being liberal or conservative.

The `YOUTUBE` and `POLBOOKS` graphs are undirected and the rest of the graphs are directed.

4.2 Clusters, bridges, and outliers Next, as for our real datasets no ground truth (if any) for "true" clustering exists, we provide anecdotal and visual study.

4.2.1 `YOUTUBE` Dataset: `PICS` finds node and feature clusters of various sizes in our largest dataset `YOUTUBE`, as shown in Figure 1, § 1. In Table 3, we show example



(a) k=1 l=2 (b) k=2, l=2 (c) k=2, l=3

(d) k=3, l=3 (e) k=4, l=3 (f) k=5, l=3

Figure 3: *Step-by-step operation of* PICS *on the synthetic attributed graph in Figure 2.*

Table 2: *Dataset summary. n: number of nodes, f: number of features, $n_1(A) + n_1(F)$: number of non-zeros (nnz), i.e. edges, in the connectivity matrix* **A** *plus the nnz in the feature matrix* **F**. *See §4.1 for more details.*

Dataset	n	f	$n_1(A)+n_1(F)$	**Connectivity and Attribute Description**
YOUTUBE	77381	30087	994542	User friendships and group memberships
TWITTER	9654	10000	81770	User 'mention's and hashtag usages
PHONECALL	94	7	391	Phone calls and affiliations
DEVICE	94	7	5233	Bluetooth device scans and affiliations
POLBOOKS	92	2	840	Book co-purchases and politic inclinations
POLBLOGS	1490	2	20580	Blog citations and political inclinations

YouTube-groups in major feature clusters; notice that these clusters can be human-labeled as 'porn', 'music', 'anime', and 'special interest'.

With respect to node clusters, PICS finds a very sparse group of 'familiar strangers' in cluster '1' with high feature coherence but scarce connections to the rest of the graph. These users belong to arbitrarily many and mostly same YouTube-groups labeled as 'A1', yet are not well-connected among themselves. Node clusters in the blue square labeled as '2' exhibit dense connectivity among each other as well as high feature homogeneity as seen in the 'anime' groups labeled as 'A2'. The node clusters in the blue square labeled as '4' mostly belong to YouTube groups associated with 'porn and music' labeled as 'A4'. Notice that the nodes in each of these clusters have quite similar connectivity to the graph. The node cluster '3' contains nodes with connections across many clusters, behaving like 'bridges'. They also mostly belong to the same YouTube-groups, labeled as 'A3'. Finally, the small node clusters constitute the outliers, with arbitrarily many connections across clusters. All in all, by using PICS we are able to understand and summarize the YOUTUBE graph in a completely unsupervised fashion.

Table 3: *Examples of YouTube-groups in feature clusters found by* PICS. *See Figure 1.*

porn	**music**	**anime**	**interest**
hotmodels	poptastic	anime4ever	streetboarders
upskirt	raphiphop	narutoholics	modelcooks
lesbokiss	guitarsolos	crazyforanime	poetryandmusic
men4men	metallovers	animefreak1	mexicogrupero
gayestgay	classicalmusic	AnimeDaisuki	bodypainting
bootyshake	xtinaaguilera	SailorMoon	nflfans
sexysolo	heavymetal	Tsyukomi	chelseafcfans

4.2.2 TWITTER Dataset: Our Twitter dataset consists of directed edges (i, j) between users if i directs a message to j at least three times. Attributes in our network are the hashtags used at least three times by a user. In Figure 4, PICS finds a group of 'casual users' in node cluster '1' with few connections and few number of hastags used. Node cluster '2' is the most prominent: this is a dense group of users (mostly tech bloggers) all from Italy who extensively mention each other but do not frequently message other users. They also use common and distinctive hashtags in Italian such as '#terremoto' and '#partigi'. The two clusters in blue square (labeled as '4') are the so-called 'heavy-hitters'

who use overwhelmingly many different hashtags (labeled as 'A4'). PICS also reveals a 'core-periphery' pattern within these users. The clusters in blue square labeled as '5' form dense diagonal blocks with dense connections within. The smaller clusters also correspond to the so-called 'bridges', with many connections across clusters. Notice that the bridge-nodes are mostly mentioned *by* others but do not themselves mention others. Upon further inspection of users in this group, we discover Jeffrey Zeldman, a well known author, as well as Jack Dorsey and Ev Williams, the two founders of Twitter. We also observe that users in groups labeled as '3' and '5' have never used the hashtags in the rightmost feature cluster of Figure 4.

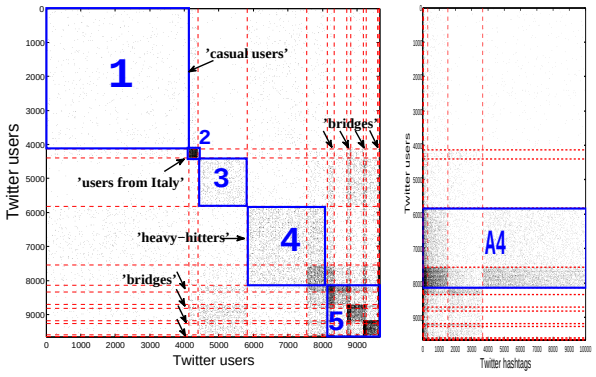


Figure 4: PICS *on* TWITTER *finds a tight group of users from Italy and reveals groups of casual users, heavy-hitters, and bridges. Left: user-mentions-user adjacency matrix; right: users-to-hashtags feature matrix.*

4.2.3 PHONECALL Dataset: PICS finds 3 major node clusters as shown in Figure 5. The first cluster corresponds to the group of *casual* subjects who make phone calls to only a few people. The next two clusters, which are relatively densely connected, are a group of *business* and a group of *grad* students, respectively.

With respect to outliers, notice a cluster of size 1, an outlier subject who does not belong to any of the clusters. This subject, whose affiliation is not given in the dataset, does not make any outgoing calls but receives calls from almost everyone, which is presumably a call service center in the campus. Another outlier we can spot is a 1st year grad student who is in the same cluster with the business students in cluster 2; he/she neither calls nor gets called by other grad students but some business students.

With respect to bridges, we notice one grad student in cluster 3 who is in mutual contact with two business students. Notice that none of the other subjects has phone call interactions with the business students.

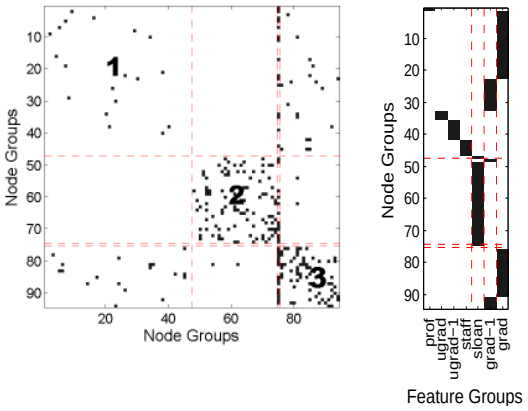


Figure 5: PICS *on* PHONECALL. *Notice* 3 *major node groups of casual users, business and grad students, respectively as well as a group of size* 1*, receiving many calls, probably a call service center.*

4.2.4 DEVICE Dataset: In Figure 6, we observe 3 major dense clusters; clusters 1, 3 and 5. These three clusters involve mostly the *grad*, *business* and *undergrad* students, respectively. The reason these clusters form near-cliques, i.e. are almost fully connected, may be due to the Bluetooth scans occurring when these groups of students sit in the same classroom all within five meters.

In this dataset, the sparser clusters are also of interest. For instance, cluster 2 is a group of subjects whose devices seem to report far less scan results. This might be due to powered-off devices or turned-off Bluetooth functionality.

In cluster 3 of business students, we also notice a column consisting of almost all zeros. This corresponds to a subject whose device can scan other devices, but somehow it does not get detected by others. This is interesting, and may be due to a malfunctioning device or missing data.

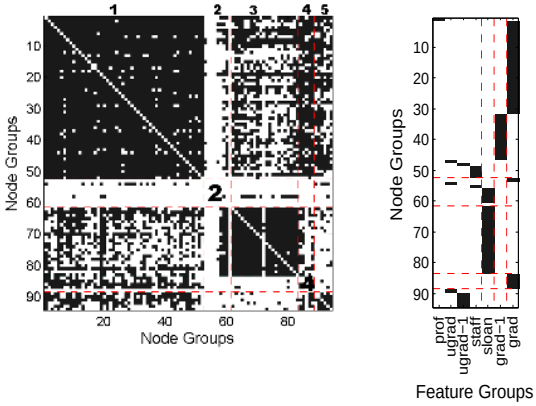


Figure 6: PICS *on* DEVICE *finds* 3 *major dense node groups of grad, business and undergrad students, respectively, as well as anomalies, probably missing data.*

Table 4: *Examples of "core" liberal and conservative books.*

Liberal	Conservative
–Lies and the Lying Liars Who Tell Them: *A Fair and Balanced Look at the Right*	–Persecution: *How Liberals Are Waging War Against Christianity*
–Big Lies: *The Right-Wing Propaganda Machine and How It Distorts the Truth*	–Deliver Us from Evil: *Defeating Terrorism, Despotism, and Liberalism*
–The Lies of George W. Bush	–Tales from the Left Coast
–Dude, Where's My Country?	–A National Party No More

4.2.5 POLBOOKS Dataset: The POLBOOKS dataset consists of liberal and conservative books which might be thought as two major clusters. In Figure 7, PICS gives more information about the cluster structure by finding 4 node clusters. The denser clusters (clusters 2 and 4) correspond to the "core" conservative and liberal books, respectively, which are often purchased together. Clusters 1 and 3 are then the corresponding "peripheral" books. Table 4 gives a list of several books in each "core". Notice that the "core" books do seem to lie at the two extremes of the political spectrum.

In cluster 1, we also observe 3 bridging books, namely *Bush at War*, *The Bushes: Portrait of a Dynasty*, and *Rise of the Vulcans: The History of Bush's War Cabinet*, which are co-purchased with both some liberal and conservative books. These books are human-labeled as conservative although they seem to have more of a historical perspective. Notice that the bridge nodes reside in the "periphery" rather than in the "core".

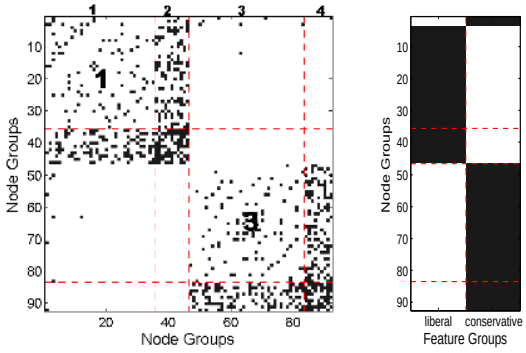


Figure 7: PICS *on* POLBOOKS *finds* 4 *node groups corresponding to "core" and "peripheral" liberal and conservative books, as well as several bridge-books, with historical content and not extremely liberal or conservative.*

4.2.6 POLBLOGS Dataset: In Figure 9, we observe 7 major node clusters in POLBLOGS. The first and the largest cluster contains liberal and conservative blogs which do not have many citations. Clusters 2-4 consist of conservative and 5-7 consist of liberal blogs. Here, PICS seems to also reveal the "core" and "periphery" structure for the political blogs. In particular, cluster 3 is a core conservative group with a fanatic follower group (cluster 4), and a less fanatic follower group (cluster 2) of other conservative blogs, which often cite the blogs in cluster 3. Examples to "core" conservative blogs include *rightwingnews.com*, *georgewbush.com*, and *conservativeeyes.blogspot.com*. Similarly, cluster 6 is a core liberal group with cluster 7 being its more fanatic, and cluster 5 being its less fanatic follower group of other liberal blogs, with many citations to cluster 6. The blogs in the liberal "core" include *talkleft.com*, *liberaloasis.com*, and *democrats.org/blog*.

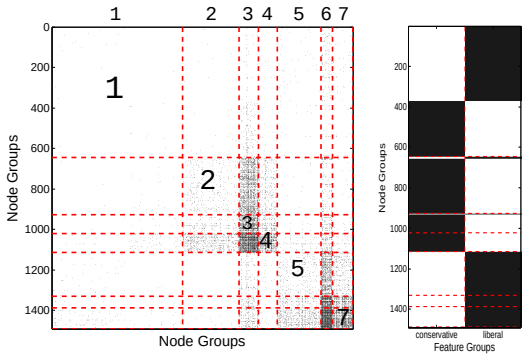
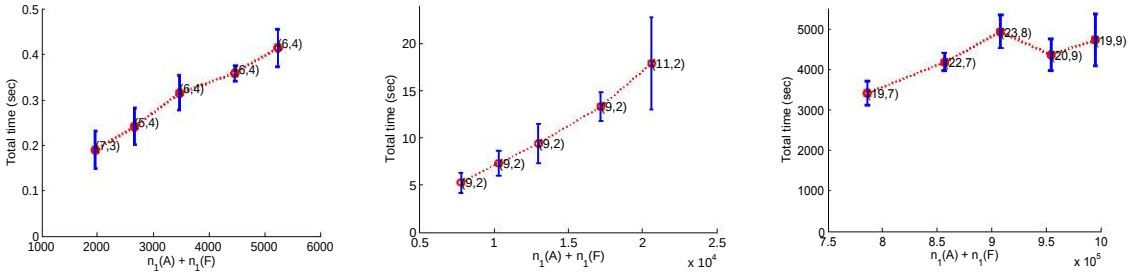


Figure 9: PICS *on* POLBLOGS. *Notice the "core" conservative and liberal blogs (clusters 3 and 6), each with two sets of "peripheral" groups with many citations.*

4.3 Scalability In § 3.6 we theoretically showed that the computational complexity of PICS is linear in the total graph and attribute size. In this section, we also demonstrate the time complexity of PICS experimentally. Figure 8 shows the running time with respect to increasing total size for several datasets we studied (the total size is the total number of non-zeros (nnz) in the **A** and **F** matrices). Notice that the run time grows linearly with respect to the total nnz. (Recall that the run time also depends on the number of clusters found, hence the slight dip for YOUTUBE at total size 950K as fewer clusters (20 vs 23) are found.) Experiments were performed on a 4-CPU Intel 3.0GHz Xeon server with 16GB RAM. All code was written in Matlab.



(a) DEVICE (b) POLBLOGS (c) YOUTUBE

Figure 8: *Run time of* PICS *versus the total number of non-zeros (nnz) in* $\mathbf{A}$ *and* $\mathbf{F}$ *(averaged over 10 runs, bars depict* $\pm$ *one standard deviation). The numbers in parentheses denote the number of node and feature clusters* (k^*, l^*) *found. Notice that the run time grows linearly w.r.t. total nnz.*

5 Conclusion

We proposed PICS, which is to the best of the authors' knowledge, the first parameter-free method for finding cohesive clusters in attributed graphs. The contributions of our work include:

- *Algorithm design:* We introduce a novel clustering model; PICS finds groups of nodes in an attributed graph with (1) *similar connectivity*, and (2) *attribute homogeneity*. By definition, clusters with similar connectivity include but are not limited to dense clusters. It also groups the node attributes into meaningful clusters. The nodes deviating from the discovered patterns correspond to bridge-nodes with connections across clusters or outlier-nodes that do not belong well to any cluster.
- *Parameter-free nature:* PICS is *fully automatic*. It works without any user-specified input, such as the number of clusters, choice of density or similarity functions and thresholds.
- *Scalability:* The run time of the proposed algorithms grows *linearly* with respect to the total graph and attribute size.
- *Effectiveness:* We show that PICS discovers quality clusters, bridges and outliers in diverse real-world datasets including YouTube and Twitter.

6 Appendix

In this section, we give the pseudo-code and describe the procedures called by PICS in § 3.4 in more detail. Simply put, Split-FeatureGroup (similarly Split-NodeGroup) increases the number of attribute (node) groups by 1 by first finding the attribute (node) group with the maximum entropy per-column (row) (Line 1), and then moving those attributes (nodes) in that group whose removal reduce the per-column (row) entropy to the new group (Lines 2-9). If no column (row) can be moved, the procedures return the original mapping (Lines 10-14).

Procedure 1.1 Split-FeatureGroup

Input: $n \times f$ feature matrix $\mathbf{F}$, C^T, l^T
Output: C^{T+1}, l^{T+1}

1: Split the column group g with the maximum entropy per-column using Equation (6.1)
2: **for** each column y in column group g **do**
3: **if** removal of y from group g decreases the per-column entropy of g as in Equation (6.2) **then**
4: Place y into the new group: $C_y^{T+1} = l^T + 1$
5: Update B_{ig}^F $\forall i$, $1 \le i \le k$.
6: **else**
7: $C_y^{T+1} = C_y^T = g$
8: **end if**
9: **end for**
10: **if** size of new feature-group > 0 **then**
11: $l^{T+1} = l^T + 1$
12: **else**
13: $l^{T+1} = l^T$
14: **end if**

Procedure 1.2 Split-NodeGroup

Input: $n \times n$ connectivity matrix $\mathbf{A}$, $n \times f$ feature matrix $\mathbf{F}$, (R^T, C^T), (k^T, l^T)
Output: R^{T+1}, k^{T+1}

1: Split the row group g with the maximum entropy per-row using Equation (6.3)
2: **for** each row x in row group g **do**
3: **if** removal of x from group g decreases the per-row entropy of g as in Equation (6.4) **then**
4: Place x into the new group: $R_x^{T+1} = k^T + 1$
5: Update B_{gj}^F $\forall j$, $1 \le j \le l$ as well as B_{gj}^A and B_{jg}^A $\forall j$, $1 \le j \le k$.
6: **else**
7: $R_x^{T+1} = R_x^T = g$
8: **end if**
9: **end for**
10: **if** size of new node-group > 0 **then**

11: $k^{T+1} = k^T + 1$
12: **else**
13: $k^{T+1} = k^T$
14: **end if**

When the number of node or attribute groups changes, `Shuffle` finds a lower-cost mapping by reassigning the nodes and attributes to the existing groups. It does so by first iterating over all the rows (nodes) (Lines 4-16) and then the columns (attributes) (Lines 17-25), assigning each row (column) to the node (attribute) group that minimizes its encoding cost (Lines 15 and 24, resp.). It repeats the same process until the total cost cannot be reduced any further (Lines 26-30).

Procedure 1.3 `Shuffle`

Input: $n \times n$ connectivity matrix $\mathbf{A}$, $n \times f$ feature matrix $\mathbf{F}$, (R^T, C^T), (k^T, l^T)
Output: (R^{T+1}, C^{T+1})

1: Let t denote the inner iteration index. Set $t = T$
2: Compute B_{ij}^t and P_{ij}^t with respect to (R^t, C^t, k^t, l^t)
3: **repeat**
4: *Shuffle rows*: Fix column assignments C^t
5: **for** each row x **do**
6: Splice x in $\mathbf{F}$ into l^t parts (each corresponding to one of the column groups in $\mathbf{F}$) Denote them as $x^1, \ldots, x^{l^t}$.
7: **for** each of these parts **do**
8: Compute the number of 1s and 0s, that is, $n_u^F(x^j)$, $u = 0, 1$ and $j = 1, \ldots, l^t$
9: **end for**
10: Splice x in $\mathbf{A}$ and the corresponding column in $\mathbf{A}$ into k^t parts. Denote them as $x_r^1, \ldots, x_r^{k^t}$ and $x_c^1, \ldots, x_c^{k^t}$, respectively.
11: **for** each of these parts **do**
12: Compute the number of 1s and 0s, that is, $n_u^A(x_r^j)$ and $n_u^A(x_c^j)$, $u = 0, 1$ and $j = 1, \ldots, k^t$
13: **end for**
14: **end for**
15: Assign each row (i.e., node) x into the node group R_x^{t+1} that yields the minimum encoding cost for x using Equation (6.5)
16: Re-compute B_{ij}^{t+1} and P_{ij}^{t+1} with respect to (R^{t+1}, C^t, k^t, l^t)
17: *Shuffle columns*: Fix row assignments R^{t+1} in $\mathbf{A}$ and $\mathbf{F}$, (also fixes the column assignment in $\mathbf{A}$, so we operate only on $\mathbf{F}$ here)
18: **for** each column y **do**
19: Splice y in $\mathbf{F}$ into k^t parts (each corresponding to one of the row groups in $\mathbf{F}$) Denote them as $y^1, \ldots, y^{k^t}$.
20: **for** each of these parts **do**
21: Compute the number of 1s and 0s, that is, $n_u^F(y^i)$, $u = 0, 1$ and $i = 1, \ldots, k^t$
22: **end for**
23: **end for**
24: Assign each column (i.e., feature) y to the feature group C_y^{t+1} that yields the minimum encoding cost for y using Equation (6.6)
25: Recompute B_{ij}^{t+1} and P_{ij}^{t+1} w.r.t. $(R^{t+1}, C^{t+1}, k^t, l^t)$
26: **if** there is no decrease in total cost **then**
27: **return** (R^t, C^t)
28: **else**
29: Set $t = t + 1$
30: **end if**
31: **until** convergence

6.1 Equations

(6.1)

$$g := \arg\max_{1 \le j \le l} \frac{1}{c_j} \quad \left(\sum_{i=1}^{k} n^F(B_{ij}) H(P_{ij}^F(1)) \right)$$

(6.2)

$$\frac{1}{c_g - 1} \sum_{i=1}^{k} n^F(B'_{ig}) H(P'^F_{ig}(1)) < \frac{1}{c_g} \sum_{i=1}^{k} n^F(B_{ig}) H(P_{ig}^F(1))$$

where B'_{ig} denotes the B_{ig} without column y.

(6.3)

$$g := \arg\max_{1 \le i \le k} \frac{1}{r_i} \Bigg(\sum_{j=1}^{l} n^F(B_{ij}) H(P_{ij}^F(1)) + \sum_{j=1}^{k} n^A(B_{ij}) H(P_{ij}^A(1)) + \sum_{j=1}^{k} n^A(B_{ji}) H(P_{ji}^A(1)) \Bigg)$$

(6.4)

$$\frac{1}{r_g - 1} \Bigg(\sum_{j=1}^{l} n^F(B'_{gj}) H(P'^F_{gj}(1)) + \sum_{j=1}^{k} \Big(n^A(B'_{gj}) H(P'^A_{gj}(1)) + n^A(B'_{jg}) H(P'^A_{jg}(1)) \Big) \Bigg) < \frac{1}{r_g} \Bigg(\sum_{j=1}^{l} n^F(B_{gj}) H(P_{gj}^F(1)) + \sum_{j=1}^{k} \Big(n^A(B_{gj}) H(P_{gj}^A(1)) + n^A(B_{jg}) H(P_{jg}^A(1)) \Big) \Bigg)$$

where B'_{gj} denotes the B_{gj} without row x.

(6.5)

$$R_x^{t+1} := \arg\min_{1 \le i \le k} \Bigg\{ \sum_{j=1}^{l} \sum_{u=0}^{1} n_u^F(x^j) \log \frac{1}{P_{ij}^F(u)}$$

$$+\sum_{j=1}^{k}\sum_{u=0}^{1}\left(n_u^A(x_r^j)\log\frac{1}{P_{ji}^A(u)}+n_u^A(x_c^j)\log\frac{1}{P_{ij}^A(u)}\right)$$
$$+d_{xx}(\log P^A_{iR_x^t}(1)+\log P^A_{R_x^t i}(1)-\log P^A_{ii}(1))$$
$$+(1-d_{xx})(\log P^A_{iR_x^t}(0)+\log P^A_{R_x^t i}(0)-\log P^A_{ii}(0))\Big\}$$

where first two lines respectively denote the cost of shifting row x in $\mathbf{F}$, and the same row x and its corresponding column in $\mathbf{A}$ to a new group i. Last two lines account for the double-counting of cell d_{xx} in $\mathbf{A}$.

(6.6)
$$C_y^{t+1} := \quad \arg\min_{1\le j\le l}\left\{\sum_{i=1}^{k}\sum_{u=0}^{1} n_u^F(x^i)\log\frac{1}{P_{ij}^F(u)}\right\}$$

Acknowledgements

Research was sponsored by the National Science Foundation under Grant No. IIS1017415 and the Army Research Laboratory under Cooperative Agreement No. W911NF-09-2-0053. It is continuing through participation in the Anomaly Detection at Multiple Scales (ADAMS) program sponsored by the U.S. Defense Advanced Research Projects Agency (DARPA) under Agreements No. W911NF-11-C-0200 and W911NF-11-C-0088. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory, of the National Science Foundation, of the U.S. Government, or any other funding parties. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

[1] L. Adamic and N. Glance. The political blogosphere and the 2004 u.s. election: Divided they blog. In *LinkKDD*, pages 36–43, 2005.

[2] R. Andersen, F. Chung, and K. Lang. Local graph partitioning using pagerank vectors. In *FOCS*, pages 475–486, 2006.

[3] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is "nearest neighbor" meaningful? In *ICDT*, pages 217–235, 1999.

[4] D. Chakrabarti. Autopart: Parameter-free graph partitioning and outlier detection. In *PKDD*, 2004.

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

[6] P. K. Chan, M. D. F. Schlag, and J. Y. Zien. Spectral k-way ratio-cut partitioning and clustering. In *DAC*, pages 749–754, 1993.

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

[8] C. H. Q. Ding, X. He, H. Zha, M. Gu, and H. D. Simon. A min-max cut algorithm for graph partitioning and data clustering. In *ICDM*, 2001.

[9] N. Eagle, A. Pentland, and D. Lazer. Inferring social network structure using mobile phone data. *PNAS*, 2007.

[10] G. W. Flake, S. Lawrence, and C. L. Giles. Efficient identification of web communities. In *KDD*. 2000.

[11] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proc. of Nat. Acad. of Sci.*, 99(12):7821–7826, 2002.

[12] P. D. Grünwald. *The Minimum Description Length Principle*. MIT Press, 2007.

[13] S. Günnemann, I. Färber, B. Boden, and T. Seidl. Subspace clustering meets dense subgraph mining: A synthesis of two paradigms. In *ICDM*, 2010.

[14] D. Hanisch, A. Zien, R. Zimmer, and T. Lengauer. Co-clustering of biological networks and gene expression data. In *ISMB*, pages 145–154, 2002.

[15] J. He, H. Tong, S. Papadimitriou, T. Eliassi-Rad, C. Faloutsos, and J. Carbonell. Pack: Scalable parameter-free clustering on k-partite graphs. In *SDM Work. on Link Anal., Cntr.terror. and Secur.*, 2009.

[16] D. S. Johnson, S. Krishnan, J. Chhugani, S. Kumar, and S. Venkatasubramanian. Compressing large boolean matrices using reordering techniques. In *VLDB*, pages 13–23, 2004.

[17] G. Karypis and V. Kumar. Multilevel algorithms for multi-constraint graph partitioning. In *Proc. of Supercomputing*, pages 1–13, 1998.

[18] H.-P. Kriegel, P. Kröger, and A. Zimek. Clustering high-dimensional data: A survey. *ACM TKDD*, 3(1):1–58, 2009.

[19] B. Long, Z. Zhang, X. Wu, and P. S. Yu. Spectral clustering for multi-type relational data. In *ICML*, volume 148, pages 585–592, 2006.

[20] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and analysis of online social networks. In *IMC*, 2007.

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

[22] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *NIPS*, 2001.

[23] J. Rissanen. A universal prior for integers and estimation by minimum description length. *The Annals of Statistics*, 11(2):416–431, 1983.

[24] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):888–905, 2000.

[25] J. Sun, C. Faloutsos, S. Papadimitriou, and P. S. Yu. Graphscope: parameter-free mining of large time-evolving graphs. In *KDD*, pages 687–696, 2007.

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

[27] `http://www-personal.umich.edu/~mejn/netdata/`. *Book citations on U.S. politics*.

[28] M. van Leeuwen, J. Vreeken, and A. Siebes. Identifying the components. In *ECML/PKDD*, 2009.

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

[30] Y. Zhou, H. Cheng, and J. X. Yu. Clustering large attributed graphs: An efficient incremental approach. In *ICDM*, pages 689–698, 2010.