# Community Learning by Graph Approximation

Bo Long
SUNY Binghamton
blong1@binghamton.edu

Xiaoyun Wu
Google Inc., USA
xiaoyunwu@google.com

Zhongfei (Mark) Zhang
SUNY Binghamton
zzhang@binghamton.edu

Philip S. Yu
IBM Watson Research Center
psyu@us.ibm.com

## Abstract

*Learning communities from a graph is an important problem in many domains. Different types of communities can be generalized as link-pattern based communities. In this paper, we propose a general model based on graph approximation to learn link-pattern based community structures from a graph. The model generalizes the traditional graph partitioning approaches and is applicable to learning various community structures. Under this model, we derive a family of algorithms which are flexible to learn various community structures and easy to incorporate the prior knowledge of the community structures. Experimental evaluation and theoretical analysis show the effectiveness and great potential of the proposed model and algorithms.*

## 1 Introduction

Learning communities from a graph is an important problem in many domains, such as web mining, social network analysis, bioinformatics, VLSI design, and task scheduling. In many applications, users are interested in strongly intra-connected communities in which the nodes are intra-community close and inter-community loose. Learning this type of communities corresponds to finding strongly connected subgraphs from a graph, which has been studied for decades as graph partitioning problem.

In addition to the strongly intra-connected communities, other types of communities also attract intensive attention in many important applications. For example, in Web mining, we are also interested in the communities of Web pages that sparsely link to each other but all densely link to the same Web pages [20], such as a community of music "fans" Web pages which share the same taste on music and are densely linked to the same set of music Web pages but sparsely linked to each other. Learning this type of communities corresponds to finding dense bipartite subgraphs from a graph,

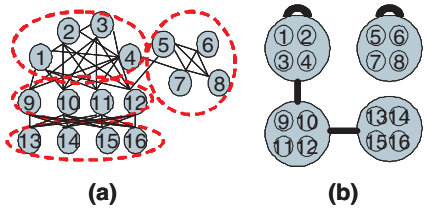


**Figure 1. A graph with mixed community structures (a) and its community prototype graph (b).**

which has been listed as one of the five algorithmic challenges in Web search engines [15].

The strongly intra-connected communities and weakly intra-connected communities are two basic community structures, and various types of communities can be generated based on them. For example, a web community could take on different structures during its development, i.e., in its early stage, it has the form of bipartite graph, since in this stage the members of the community share the same interests (linked to the same web pages) but have not known (linked to) each other; in the later stage, with members of the community start linking to each other, the community becomes a hybrid of the aforementioned two basic community structures; in the final stage it develops into a larger strongly intra-connected community.

These various types of communities can be unified into a general concept, link-pattern based community. *A link-pattern based community is a group of nodes which have the similar link patterns*, i.e., the nodes within a community link to other nodes in similar ways. Let us have an illustrative example. Figure 1(a) shows a graph of mixed types of communities. There are four communities in Figure 1(a), $\mathcal{C}_1 = \{v_1, v_2, v_3, v_4\}$, $\mathcal{C}_2 = \{v_5, v_6, v_7, v_8\}$, $\mathcal{C}_3 = \{v_9, v_{10}, v_{11}, v_{12}\}$, and $\mathcal{C}_4 = \{v_{13}, v_{14}, v_{15}, v_{16}\}$. Within the strongly intra-connected community $\mathcal{C}_1$, the nodes have the similar link patterns, i.e., they all strongly link to the nodes in $\mathcal{C}_1$ (their own community) and $\mathcal{C}_3$, and weakly link to the nodes in $\mathcal{C}_2$ and $\mathcal{C}_4$; Within the weakly intra-connected community $\mathcal{C}_3$, the nodes also have the similar

link patterns, i.e., they all weakly link to the nodes in $\mathcal{C}_3$ (their own community), and $\mathcal{C}_2$, strongly link to the nodes in $\mathcal{C}_1$ and $\mathcal{C}_4$; Similarly for the nodes in community $\mathcal{C}_3$ and the nodes in community $\mathcal{C}_4$. Note that graph partitioning approaches cannot correctly identify the community structure of the graph in Figure 1(a), since they seek only strongly intra-connected communities by cutting a graph into disjoint subgraphs to minimize edge cuts.

In addition to unsupervised community learning applications, the concept of the link-pattern based community also provides a simple approach for semi-supervised learning on graphs. In many applications, graphs are very sparse and there may exist a large mount of isolated or nearly-isolated nodes which do not have community patterns. However, according to extra supervised information (domain knowledge) these nodes may belong to certain communities. To incorporate the supervised information, a common approach is to manually label these nodes. However, for a large graph, manually labeling is labor-intensive and expensive. Furthermore, to make use of these labels, instead of supervised learning algorithms, different semi-supervised algorithms need to be designed. The concept of the link-pattern based community provides a simple way to incorporate supervised information by adding virtual nodes to graphs. The idea is that if the nodes which belong to the same community according to the supervised information, they are linked to the same virtual nodes. Then an algorithm which is able to learn general link-pattern based communities can be directly applied to the graphs with virtual nodes to make use of the supervised information to learn community patterns.

For example, to find the hidden classes from a collection of documents, a common approach is to represent the collection as a graph in which each node denotes a document and each edge weight denotes the similarity between two documents [10, 35]. Usually the similarities are calculated based on the term-frequency vectors of documents. However, there may exist documents which share no or very few words with each other but still belong to the same community according to extra domain information. Let us have an illustrative example. In Figure 2, the dark color nodes (documents) do not share same words and are not linked to each other. However, they all belong to the "vehicle" community. By adding virtual nodes (documents) (light color nodes in Figure 2) which are concept documents consisting of popular words for the "vehicle" community, the originally isolated document nodes are linked to the virtual document nodes and the supervised information is embedded into link patterns.

Therefore, various applications involving unsupervised as well as semi-supervised community learning have presented a great need to link-pattern based community learning algorithms. In this paper, we propose a general model based on graph approximation to learn the link-pattern based community structure from a graph. By unifying the traditional edge cut objectives, the model provides a new

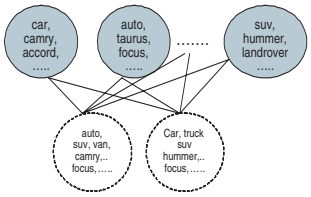


**Figure 2. A graph with virtual nodes.**

view to understand graph partitioning approaches and at the same time it is applicable to learning various community structures. Under this model, we derive three novel algorithms to learn the general community structures from a graph, which cover three main versions of unsupervised learning algorithms, hard, soft and balanced version, to provide a complete family of community learning algorithms. This family of algorithms has the following advantages: they are flexible to learn various types of communities; when applied to learning strongly intra-connected communities, this family evolves to a new family of effective graph partition algorithms; it is easy for the proposed algorithms to incorporate the prior knowledge of the community structure into the algorithms. Experimental evaluation and theoretical analysis show the effectiveness and great potential of the proposed model and algorithms.

## 2 Related Work

Graph partitioning divides the nodes of a graph into communities by finding the best edge cuts of the graph. Several edge cut objectives, such as the average cut [5], average association [30], normalized cut [30], and min-max Cut [11], have been proposed. Various spectral algorithms have been developed for these objective functions [5, 30, 11]. These algorithms use the eigenvectors of a graph affinity matrix, or a matrix derived from the affinity matrix, to partition the graph. Since eigenvectors computed do not correspond directly to individual partitions, a postprocessing approach [34], such as k-means, must be applied to find the final partitions.

Multilevel methods have been used extensively for graph partitioning with the Kernighan-Lin objective, which attempt to minimize the cut in the graph while maintaining equal-sized clusters [4, 14, 17]. In multilevel algorithms, the graph is repeatedly coarsened level by level until only a small number of nodes are left. Then, an initial partitioning on this small graph is performed. Finally, the graph is uncoarsened level by level, and at each level, the partitioning from the previous level is refined using the refinement algorithm.

Recently, graph partitioning with an edge cut objective has been shown to be mathematically equivalent to an appropriate weighted kernel k-means objective function [7, 8]. Based on this equivalence, the weighted kernel k-means algorithm has been proposed for graph partitioning [9, 7, 8].

Learning communities from a graph has also been intensively studied in the context of social network analysis

[29]. Hierarchical clustering [29, 33] has been proposed to learn communities. Recent algorithms [13, 24, 6] address several problems related to the prior knowledge of community size, the precise definition of inter-nodes similarity measure, and improved computational efficiency [25]. However, their main focus is still learning strongly intra-connected communities. Some efforts [12, 31, 16, 16, 2] can be viewed as community learning based on stochastic block modeling.

There are efforts in the literature focusing on finding communities based on dense bipartite graphs [20, 27]. The trawling algorithm [20] extracts communities (which are called emerging communities in [20] as the counterpart concept of strongly intra-connected community) by first applying the the Apriori algorithm to find all possible cores (complete bipartite graphs) and then expanding each core to a full-fledged community with HITS algorithm [19]. [27] proposes a different approach to extract the emerging communities by finding all bipartite graphs instead of finding cores.

In this study, we focus on how to divide the nodes of a graph into disjoint communities based on link patterns.

## 3 Model Formulation

In this section, we propose a general model to learn link-pattern based communities from a graph.

To derive our model to learn latent community structure from a graph, we start from the following simpler problem: if the link-pattern based community structure of any graph is known, can we draw a simple graph with explicit latent community structure (latent link patterns) to represent the original graph? We present the concept of a *community prototype graph* as an answer. A community prototype graph consists of a set of the community nodes and a set of links including self-links for each community node and inter-links for a pair of community nodes.

For example, Figure 1(b) shows a community prototype graph for the graph in Figure 1(a). Note that for convenience, in all the examples, we use 0-1 graphs where the edge weight 0 denotes the absence of edge between two nodes and the edges with weight 0 are not shown in the graphs. However, all the discussions are applicable to a general weighted graph. In Figure 1(b), the top-left community node is associated with the nodes of $\mathcal{C}_1 = \{v_1, v_2, v_3, v_4\}$ from the original graph; the self-link of the top-left community node implies that all its associated nodes are linked to each other; the inter-link between the top-left community node and the bottom-left community node implies that the nodes of $\mathcal{C}_1 = \{v_1, v_2, v_3, v_4\}$ are linked to those of $\mathcal{C}_3 = \{v_9, v_{10}, v_{11}, v_{12}\}$. Hence, the community prototype graph in Figure 1(b) provides a clear view of the community structure and link patterns for the original graph in Figure 1(a). Given the community structure of any graph, we can always draw its community prototype graph.

Therefore, learning the hidden community structures

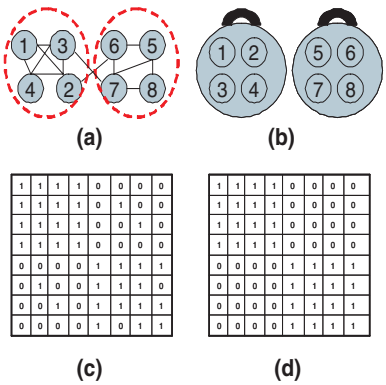


**Figure 3. A graph with strongly-connected communities (a) and its community prototype graph (b); the graph affinity matrices for (a) and (b), (c) and (d), respectively.**

from a graph can be formulated as finding its optimal community prototype graph which is the "closest" to the original graph, i.e., based on this community prototype graph, the original graph can be constructed most precisely. By representing a graph as an affinity matrix, this problem can be formally formulated as an optimization problem of matrix approximation,

$$\arg\min_{A^*} ||A - A^*||^2, \tag{1}$$

where $A \in \mathbb{R}_+^{n \times n}$ denotes the affinity matrix of the original graph and $A^* \in \mathbb{R}_+^{n \times n}$ denotes the affinity matrix of a community prototype graph. The examples of $A$ and $A^*$ are given in Figure 3(c) and 3(d), which are the affinity matrices for the original graph Figure 3(a) and its community prototype graph Figure 3(b), respectively.

Due to the special structure of a community prototype graph, its affinity matrix can be represented as a product of three factors such that $A^* = CBC^T$, where $C \in \{0,1\}^{n \times k}$ such that $k$ is the number of communities and $\sum_j C_{ij} = 1$, i.e., $C$ is an indicator matrix that provides the community membership of each node (without loss of generality, we assume there is no empty community); $B \in \mathbb{R}_+^{k \times k}$ such that $B$ is the *community structure matrix* that provides an intuitive representation of the community structure, since $B_{ii}$ denotes the self-link weight for the $i$th community node and $B_{ij}$ for $i \neq j$ denotes inter-link weight between the $i$th and the $j$th community nodes.

Based on the above observation, formally we define the problem of learning communities from an undirected graph as follows.

**Definition 3.1.** *Given an undirected graph $G = (\mathcal{V}, \mathcal{E}, A)$ where $A \in \mathbb{R}_+^{n \times n}$ is the affinity matrix, and a positive integer $k$, the optimized communities are given by the minimization,*

$$\min_{\substack{C \in \{0,1\}^{n \times k}, B \in \mathbb{R}_+^{k \times k} \\ C\mathbf{1} = \mathbf{1}}} ||A - CBC^T||^2. \tag{2}$$

*where* $\mathbf{1}$ *denotes a vector consisting of 1's (we omit its dimension, since it is clear in the context).*

Definition 3.1 provides a general model, Community Learning by Graph Approximation (CLGA), to learn various community structures from graphs. In the CLGA model, the number of communities $k$ is given. How to decide the optimal $k$ is a non-trivial model selection problem and beyond the scope of this paper.

The problems of finding specific types of communities can be formulated as special cases of the CLGA model. For example, although there are different formulations of graph partitioning with different objective functions, they all can be viewed as special cases of the CLGA model. Since graph partitioning is a very important case of community learning, we propose the following theorem to establish the connection between the CLGA model and the existing graph partitioning objectives.

Without loss of generality, we first re-define the cluster indicator matric $C$ as the following weighted cluster indicator matrix $\tilde{C}$,

$$\tilde{C}_{ij} = \begin{cases} \frac{1}{|\pi_j|^{\frac{1}{2}}} & \text{if } v_i \in \pi_j \\ 0 & \text{otherwise} \end{cases}$$

where $|\pi_j|$ denotes the number of nodes in the $j$th community. Clearly $\tilde{C}$ still captures the disjoint community memberships and $\tilde{C}^T\tilde{C} = I_k$ where $I_k$ denotes $k \times k$ identity matrix.

**Theorem 3.2.** *The CLGA model in Definition 3.1 with the extra constraint that $B$ is an identity matrix, i.e.,*

$$\min_{\tilde{C}} ||A - \tilde{C}\tilde{C}^T||^2, \tag{3}$$

*is equivalent to the maximization*

$$\max_{\tilde{\mathbf{c}}_1,\ldots,\tilde{\mathbf{c}}_k} \sum_{p=1}^{k} \tilde{\mathbf{c}}_p^T A \tilde{\mathbf{c}}_p \tag{4}$$

*where $\tilde{\mathbf{c}}_p$ denotes the pth column vector of $\tilde{C}$.*

*Proof.* Let $\text{tr}(X)$ denote the trace of a matrix $X$ and $L$ denote the objective function in Eq. 3.

$$\begin{aligned} L &= \text{tr}((A - \tilde{C}\tilde{C}^T)^T(A - \tilde{C}\tilde{C}^T)) && (5)\\ &= \text{tr}(A^TA) - 2\text{tr}(\tilde{C}\tilde{C}^TA) + \text{tr}(\tilde{C}\tilde{C}^T\tilde{C}\tilde{C}^T) && (6)\\ &= \text{tr}(A^TA) - 2\text{tr}(\tilde{C}^TA\tilde{C}) + k && (7)\\ &= \text{tr}(A^TA) - 2\sum_{p=1}^{k} \tilde{\mathbf{c}}_p^T A \tilde{\mathbf{c}}_p + k && (8) \end{aligned}$$

The above deduction uses the property of trace $\text{tr}(XY) = \text{tr}(YX)$ . Based on Eq.(8), since $\text{tr}(A^TA)$ and $k$, the number of communities, are constants, the minimization of $L$ is equivalent to the maximization of $\sum_{p=1}^{k} \tilde{\mathbf{c}}_p^T A \tilde{\mathbf{c}}_p$. The proof is completed. □

**Table 1. A list of variations of the CLGA model.**

| Model | Constraints on B |
|---|---|
| GCL | no cotraints |
| IGP | Identity matrix |
| GGP | Diagonal matrix |
| IBCL | Zero diagonal elements and unit off-diagonal elemtnts |
| GBCL | Zero diagonal elements |

Theorem 3.2 states that if we fix the community structure matrix $B$ as the identity matrix $I_k$ (the more general case is $aI_k$ for any $a \in \mathbb{R}_+$), the CLGA model is reduced to the trace maximization in (4). Since various graph partitioning objectives, such as ratio association [30], normalized cut [30], ratio cut [5], and Kernighan-Lin objective [18], can be formulated as the trace maximization [7], Theorem 3.2 establishes the connection between the CLGA model and the existing graph partitioning objectives.

Therefore, the traditional graph partitioning can be viewed as a special case of the CLGA model in which $B$ is restricted to be an identity matrix. By fixing B as an identity matrix, the traditional graph partitioning objectives make an implicit assumption about community structure of the target graph, i.e., they assume that the nodes within each community are fully connected (the diagonal elements of $B$ are all 1's) and the nodes between communities are disconnected (the off-diagonal elements of $B$ are all 0's), i.e., the community prototype graphs consist of a set of separated community nodes with self-links. This assumption is consistent with our intuition about an ideal partitioning (we call this special case of CLGA model Ideal Graph Partitioning (IGP)). However, it cannot catch the community structures deviating from the ideal case. For example, for a graph that has one strongly intra-connected graph and one relative weak intra-connected graph, assuming $B$ to be $\left[\begin{smallmatrix} 1 & 0 \\ 0 & 0.5 \end{smallmatrix}\right]$ may be better than $\left[\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right]$. Furthermore, the CLGA model provides the flexibility to learn $B$ under various constraints. If B is relaxed from the identity matrix to any diagonal matrix, i.e., if we assume zero connectivity between communities but let the algorithm learn the within-community connectivity, we obtain a new graph partitioning model as another special case of the general CLGA model, which we call General Graph Partitioning (GGP).

Similarly, with the appropriate constraint on $B$, the CLGA model may focus on other specific types of community structures. For example, by restricting $B$ to be the matrix whose diagonal elements are 0 and off-diagonal elements are 1, CLGA learns the ideal weakly intra-connected communities among which each pair of communities forms a dense bi-partite graph (call it Ideal Bi-partite Community Learning (IBCL)); by restricting $B$ to be the matrix whose diagonal elements are 0, CLGA learns communities of general bi-partite subgraphs (call it General Bi-partite Community Learning (GBCL)).

Table 1 summarizes several variations of the CLGA

model. For simplicity, for the general situation without any constraints on $B$, we call it as General Community Learning (GCL). The community structure matrix $B$ plays an important role in the CLGA model. If we have prior knowledge about the community structure of the graph or we are only interested in some special types of communities, it is easy to incorporate it into the model by putting an appropriate constraint on $B$.

# 4 Algorithm Derivation

In this section, we derive algorithms for the basic CLGA model in Definition 3.1 and its extensions, soft CLGA model and balanced CLGA model.

## 4.1 Hard CLGA Algorithm

The CLGA model in Definition 3.1 seeks hard community membership for each node and the problem can be shown to be NP-hard. The proof is easy since based on Theorem 3.2 it can be reduced to the graph partitioning problem, which is NP-hard. We derive an alternative optimization algorithm for the hard CLGA model.

We prove the following theorem which is the basis of our algorithm.

**Theorem 4.1.** *If $C \in \{0,1\}^{n\times k}$ and $B \in \mathbb{R}_+^{k\times k}$ are the optimal solution to the minimization in Definition 3.1, then*

$$B = (C^TC)^{-1}C^TAC(C^TC)^{-1}. \tag{9}$$

*Proof.* The objective function in Definition 3.1 can be expanded as follows.

$$\begin{aligned} L &= ||A - CBC^T||^2 \\ &= \text{tr}((A - CBC^T)^T(A - CBC^T)) \\ &= \text{tr}(A^TA) - 2\text{tr}(CBC^TA) - \text{tr}(CBC^TCBC^T) \end{aligned}$$

Take the derivative with respect to $B$, we obtain

$$\frac{\partial L}{\partial B} = -2C^TAC + 2C^TCBC^TC. \tag{10}$$

Solve $\frac{\partial L}{\partial B} = 0$ to obtain

$$B = (C^TC)^{-1}C^TAC(C^TC)^{-1}; \tag{11}$$

Note that $C^TC$ is a special diagonal matrix such that $[C^TC]_{pp} = |\pi_p|$, the size of the $p$th community, and since $A$ is a non-negative symmetric matrix, so is $B$. This completes the proof of the theorem. □

Based on Theorem 4.1, we propose an iterative algorithm which alternatively updates $B$ and $C$ and converges to a local optimum. First, we fix $C$ and update $B$. Eq (9) in Theorem 4.1 provides an updating rule for $B$. This updating rule can be implemented more efficiently than it looks like. First, it does not really involve computing inverse matrices, since $C^TC$ is a special diagonal matrix with the size of each community on its diagonal; second, the product of $C^TAC$ can be calculated without normal matrix multiplication, since $C$ is an indicator matrix.

**Algorithm 1** Hard CLGA algorithm

**Input:** A graph affinity matrix $A$ and a positive integer $k$.
**Output:** A community membership matrix $C$ and a community structure matrix $B$.
**Method:**
1: Initialize $B$.
2: **repeat**
3: **for** $h = 1$ to $n$ **do**
4: $C_{hp^*} = 1$ for $p^* = \arg\min_p L_p$ where $L_p$ denotes $||A - CBC^T||^2$ for $C_{hp} = 1$.
5: **end for**
6: $B = (C^TC)^{-1}C^TAC(C^TC)^{-1}$.
7: **until** convergence

Second, we fix $B$ and update $C$. Since each row of $C$ is an indicator vector with only one element equal to 1, we adopt the re-assignment procedure to update $C$ row by row. To determine which element of the $h$th row of $C$ is equal to 1, for $p = 1, \ldots, k$, each time we let $C_{hp} = 1$ and compute the objective function $L = ||A - CBC^T||^2$, which is denoted as $L_p$, then

$$C_{hp^*} = 1 \text{ for } p^* = \arg\min_p L_p \tag{12}$$

Note that when we update the $h$th row of $C$, the necessary computation involves only the $h$th row or column of $A$ and $CBC^T$.

The algorithm, Hard CLGA, is summarized in Algorithm 1. The hard CLGA algorithm learns the general community structures from a graph, since it does not put any constraint on $B$. However, it is trivial to modify it to solve the variations of the general CLGA model, such as IGP, GGP, IBCL and GBCL in Table 1. For example, the hard IGP algorithm works as follows: fix $B$ as the identity matrix and simply update $C$ by updating rule (12) until convergence; the hard GGP algorithm works as follows: fix the off-diagonal elements of $B$ as zero; update the diagonal elements of $B$ by updating rule (9) and update $C$ by updating rule (12) until convergence. Besides the variations in Table 1, it is easy for Algorithm 1 to incorporate other prior knowledge through $B$.

The complexity of hard CLGA can be shown to be $O(tn^2k)$ where $t$ is the number of iterations. It can be further reduced for sparse graphs. When applied to graph partitioning task, the hard CLGA algorithm is computationally more efficient than the popular spectral approaches which involve expensive eigenvector computation and extra post-processing on eigenvectors to obtain the partitioning. Compared with the multi-level approaches such as METIS [17], CLGA does not restrict communities to have an equal size.

The proof of the convergence of Algorithm 1 is easy due to the following facts. First, based on Theorem 4.1, the objective function is non-increasing under updating rule (9); second, by the criteria for reassignment in updating rule

(12), it is trivial to show that the objective function is non-increasing under updating rule (12).

### 4.2 Soft CLGA algorithm

In the hard CLGA model of Definition 3.1, each node belongs to only one community. It is natural to extend it to a soft version, in which each node could belong to more than one community with certain degrees. Formally, we define the soft CLGA model as follows.

**Definition 4.2.** *Given an undirected graph $G = (\mathcal{V}, \mathcal{E}, A)$ where $A \in \mathbb{R}_+^{n\times n}$ is the affinity matrix, and a positive integer $k$, the optimized communities are given by the minimization,*

$$\min_{\substack{C\in\mathbb{R}_+^{n\times k}, B\in\mathbb{R}_+^{k\times k}\\ C\mathbf{1}=\mathbf{1}}} ||A - CBC^T||^2. \tag{13}$$

In the soft CLGA model, $C$ is a soft membership matrix such that $C_{ij}$ denotes the degree that the $i$th node is associated with the $j$th community and the sum of all the degrees for each node equals to 1. Since the soft CLGA model defines a constrained non-convex optimization, it is not realistic to expect an algorithm to find a global optimum. We propose an alternative optimization algorithm which converges to a local optimum.

Deriving the updating rules for $C$ and $B$ in the soft CLGA model is more difficult than in the hard CLGA model. First, it is difficult to deal with the constraint $\sum_j C_{ij} = 1$ efficiently. Hence, we transform it to a "soft" constraint, i.e., we implicitly enforce the constraint by adding a penalty term, $\alpha||C\mathbf{1} - \mathbf{1}||^2$ where $\mathbf{1}$ is a k-dimension vector consisting of 1's, $\alpha$ is a positive constant. Therefore, we obtain the following optimization.

$$\min_{C\in\mathbb{R}_+^{n\times k}, B\in\mathbb{R}_+^{k\times k}} ||A - CBC^T||^2 + \alpha||C\mathbf{1} - \mathbf{1}||^2. \tag{14}$$

Fixing $B$, the objective function in (14) is quartic with respect to $C$. We derive a simple and efficient updating rule for $C$ based on the bound optimization procedure [28, 22]. The basic idea is to construct an auxiliary function which is a convex upper bound for the original objective function based on the solution obtained from the previous iteration. Then, a new solution for the current iteration is obtained by minimizing this upper bound.

**Definition 4.3.** *$G(S, S^t)$ is an auxiliary function for $F(S)$ if $G(S, S^t) \geq F(S)$ and $G(S, S) = F(S)$.*

The auxiliary function is useful due to the following lemma.

**Lemma 4.4.** *If $G$ is an auxiliary function, then $F$ is non-increasing under the updating rule $S^{t+1} = \arg\min_S G(S, S^t)$.*

*Proof.* $F(S^{t+1}) \leq G(S^{t+1}, S^t) \leq G(S^t, S^t) \leq F(S^t)$. □

We propose an auxiliary function for $C$ in the following theorem.

**Lemma 4.5.**

$$\begin{aligned} G(C, \tilde{C}) &= \sum_{ij}(A_{ij} + \frac{\alpha}{n} - 2\sum_{gh}(A_{ij}\tilde{C}_{ig}B_{gh}\tilde{C}_{jh}(1 + 2\log C_{jh} \\ &\quad -2\log\tilde{C}_{jh}) + \frac{\alpha}{nk}\tilde{C}_{jh}(1 + \log C_{jh} - \log\tilde{C}_{jh})) + \\ &\quad \sum_{gh}([\tilde{C}B\tilde{C}^T]_{ij}\tilde{C}_{ig}B_{gh}\tilde{C}_{jh}\frac{C_{jh}^4}{\tilde{C}_{jh}^4} + \\ &\quad \frac{\alpha}{2nk}[\tilde{C}\mathbf{1}]_j\tilde{C}_{jh}(\frac{C_{jh}^4}{\tilde{C}_{jh}^4} + 1))) \end{aligned}$$

*is an auxiliary function for*

$$F(C) = ||A - CBC^T||^2 + \alpha||C\mathbf{1} - \mathbf{1}||^2. \tag{15}$$

*Proof.* For convenience, we let $\beta = \frac{\alpha}{nk}$.

$$\begin{aligned} F(C) &= \sum_{ij}((A_{ij} - \sum_{gh}C_{ig}B_{gh}C_{jh})^2 + \beta\sum_{gh}(C_{jh} - 1)^2) \\ &\leq \sum_{ij}(\sum_{gh}\frac{\tilde{C}_{ig}B_{gh}\tilde{C}_{jh}}{[\tilde{C}B\tilde{C}^T]_{ij}}(A_{ij} - \frac{[\tilde{C}B\tilde{C}^T]_{ij}}{\tilde{C}_{ig}B_{gh}\tilde{C}_{jh}}C_{ig}B_{gh}C_{jh})^2 \\ &\quad +\beta\sum_{gh}\frac{\tilde{C}_{jh}}{[\tilde{C}\mathbf{1}]_j}(\frac{[\tilde{C}\mathbf{1}]_j}{\tilde{C}_{jh}}C_{jh} - 1)^2) \\ &= \sum_{ij}(A_{ij} - 2\sum_{gh}A_{ij}C_{ig}B_{gh}C_{jh} + \\ &\quad \sum_{gh}\frac{[\tilde{C}B\tilde{C}^T]_{ij}}{\tilde{C}_{ig}B_{gh}\tilde{C}_{jh}}C_{ig}^2B_{gh}^2C_{jh}^2 + \beta\sum_{gh}\frac{[\tilde{C}\mathbf{1}]_j}{\tilde{C}_{jh}}C_{jh}^2 \\ &\quad -2\beta\sum_{gh}C_{jh} + k\beta) \\ &= \sum_{ij}(A_{ij} + k\beta - 2\sum_{gh}(A_{ij}\tilde{C}_{ig}B_{gh}\tilde{C}_{jh}\frac{C_{ig}C_{jh}}{\tilde{C}_{ig}\tilde{C}_{jh}} + \\ &\quad \beta\tilde{C}_{jh}\frac{C_{jh}}{\tilde{C}_{jh}}) + \sum_{gh}([\tilde{C}B\tilde{C}^T]_{ij}\tilde{C}_{ig}B_{gh}\tilde{C}_{jh}\frac{C_{ig}^2C_{jh}^2}{\tilde{C}_{ig}^2\tilde{C}_{jh}^2} \\ &\quad +\beta[\tilde{C}\mathbf{1}]_j\tilde{C}_{jh}\frac{C_{jh}^2}{\tilde{C}_{jh}^2})) \\ &\leq \sum_{ij}(A_{ij} + k\beta - 2\sum_{gh}(A_{ij}\tilde{C}_{ig}B_{gh}\tilde{C}_{jh}(1 + \log C_{ig} \\ &\quad + \log C_{jh} - \log\tilde{C}_{ig} - \log\tilde{C}_{jh}) + \beta\tilde{C}_{jh}(1 + \log C_{jh} - \\ &\quad \log\tilde{C}_{jh})) + \sum_{gh}(\frac{1}{2}[\tilde{C}B\tilde{C}^T]_{ij}\tilde{C}_{ig}B_{gh}\tilde{C}_{jh}(\frac{C_{ig}^4}{\tilde{C}_{ig}^4} + \frac{C_{jh}^4}{\tilde{C}_{jh}^4}) \\ &\quad +\frac{1}{2}\beta[\tilde{C}\mathbf{1}]_j\tilde{C}_{jh}(\frac{C_{jh}^4}{\tilde{C}_{jh}^4} + 1))) \\ &= \sum_{ij}(A_{ij} + k\beta - 2\sum_{gh}(A_{ij}\tilde{C}_{ig}B_{gh}\tilde{C}_{jh}(1 + 2\log C_{jh} \\ &\quad -2\log\tilde{C}_{jh}) + \beta\tilde{C}_{jh}(1 + \log C_{jh} - \log\tilde{C}_{jh})) + \\ &\quad \sum_{gh}([\tilde{C}B\tilde{C}^T]_{ij}\tilde{C}_{ig}B_{gh}\tilde{C}_{jh}\frac{C_{jh}^4}{\tilde{C}_{jh}^4} + \\ &\quad \frac{1}{2}\beta[\tilde{C}\mathbf{1}]_j\tilde{C}_{jh}(\frac{C_{jh}^4}{\tilde{C}_{jh}^4} + 1))) \end{aligned}$$

During the above deduction, the second step uses Jensen's inequality and the fifth step uses the inequalities $x \geq 1 + \log x$ and $x^2 + y^2 \geq 2xy$. □

The following theorem provides the updating rule for $C$.

**Theorem 4.6.** *The objective function $F(C)$ in Eq.*(15) *is nonincreasing under the updating rule,*

$$C = \tilde{C} \odot (\frac{A\tilde{C}B + \frac{\alpha}{2}}{\tilde{C}B\tilde{C}^T\tilde{C}B + \frac{\alpha}{2}\tilde{C}E})^{\frac{1}{4}} \quad (16)$$

*where $\tilde{C}$ denotes the solution from the previous iteration, $E$ denotes a $k \times k$ matrix of 1's, $\odot$ denotes entry-wise product, and the division between two matrix is entry-wise division.*

The Theorem can be proved by solving $\frac{\partial G(C,\tilde{C})}{\partial C_{jh}} = 0$ and using Lemma 4.4 (details are omitted due to the space limit).

Similarly, we present the following theorems to derive updating rules for $B$. Note that Theorem 4.1 cannot be used to update $B$, since $C$ does not have the special structure of the indicator matrix in this case; updating rule (9) cannot guarantee that $B$ is non-negative. To guarantee that $B$ can be updated appropriately, we have the following theorems.

**Lemma 4.7.**

$$G(B, \tilde{B}) = \sum_{ij}(A_{ij} - 2\sum_{gh} A_{ij}C_{ig}B_{gh}C_{jh} + \sum_{gh}[C\tilde{B}C]_{ij}C_{ig}C_{jh}\frac{B_{gh}^2}{\tilde{B}_{gh}})$$

*is an auxiliary function for*

$$F(B) = ||A - CBC^T||^2. \quad (17)$$

**Theorem 4.8.** *The objective function $F(B)$ in Eq.*(17) *is nonincreasing under the updating rule*

$$B = \tilde{B} \odot \frac{C^TAC}{C^TC\tilde{B}C^TC}. \quad (18)$$

Following the way to prove Lemma 4.5 and Theorem 4.6, it is easy to prove the above theorems. We omit details here.

The soft CLGA algorithm is summarized in Algorithm 2. The implementation of the algorithm is simple and it is easy to take the advantage of the distributed computation for very large data. The complexity of the algorithm is still $O(tn^2k)$ for $t$ iterations and it can be further reduced for sparse graphs. The convergence of the soft CLGA algorithm is guaranteed by Theorem 4.6 and 4.8.

Like the hard CLGA algorithm, the soft CLGA can be applied to learning the specific types of community structures by enforcing the corresponding constraint on $B$. For example, soft IGP and GGP provide another two new graph partitioning algorithms and they deal with the soft graph partitioning problem which has not been addressed extensively in the literature of graph partitioning.

**Algorithm 2** Soft CLGA algorithm

**Input:** A graph affinity matrix $A$ and a positive integer $k$.
**Output:** A community membership matrix $C$ and a community structure matrix $B$.
**Method:**

1: Initialize $B$ and $C$.
2: **repeat**
3:
$$B = B \odot \frac{C^TAC}{C^TCBC^TC}.$$
4:
$$C = C \odot (\frac{ACB + \frac{\alpha}{2}}{CBC^TCB + \frac{\alpha}{2}CE})^{\frac{1}{4}}$$
5: **until** convergence

## 4.3 Balanced CLGA Algorithm

In some applications, users may be interested in communities of a balance size. We propose the balanced CLGA model as follows.

**Definition 4.9.** *Given an undirected graph $G = (\mathcal{V}, \mathcal{E}, A)$ where $A \in \mathbb{R}_+^{n\times n}$ is the affinity matrix, and a positive integer $k$, the optimized communities are given by the minimization,*

$$\min_{\substack{C\in\mathbb{R}_+^{n\times k}, B\in\mathbb{R}_+^{k\times k} \\ C^T\mathbf{1}=\mathbf{1}}} ||A - CBC^T||^2. \quad (19)$$

Unlike $C\mathbf{1} = \mathbf{1}$ in the soft CLGA model, we have $C^T\mathbf{1} = \mathbf{1}$ in the balanced CLGA model, i.e., the sum of elements in each column of the community membership matrix equals to 1. This constraint enforces that for each community, the sum of degrees that each node is associated with this community equals to 1. As a result, the more the nodes in a certain community, the smaller the average degree that each node is associated with this community, the more a node tends to belong to other communities with a relative larger degree. Therefore, compared with the other two CLGA models, the balanced CLGA model tends to provide more balanced communities. Note that this constraint does not enforce strictly balanced communities of equal sizes and it just pushes the model to provide communities as balanced as possible.

Due to the following lemma, the balanced CLGA model can be simplified by dropping the constraint $C^T\mathbf{1} = \mathbf{1}$.

**Lemma 4.10.** *If $C \in \mathbb{R}_+^{n\times k}$, $B \in \mathbb{R}_+^{k\times k}$, and $D \in \mathbb{R}_+^{k\times k}$ is a diagonal matrix s.t. $D_{jj} = \frac{1}{\sum_i C_{ij}}$, then $CBC^T = CDD^{-1}BD^{-1}(CD)^T$ and $(CD)^T\mathbf{1} = \mathbf{1}$.*

*Proof.* omitted. □

Lemma 4.10 implies that we can always normalize $C$ to satisfy the constraint without changing the value of the objective function. Hence, the balanced CLGA can be reduced to the following optimization,

$$\min_{C\in\mathbb{R}_+^{n\times k}, B\in\mathbb{R}_+^{k\times k}} ||A - CBC^T||^2. \quad (20)$$

**Algorithm 3** Balanced CLGA algorithm

**Input:** A graph affinity matrix $A$ and a positive integer $k$.
**Output:** A community membership matrix $C$ and a community structure matrix $B$.
**Method:**

1: Initialize $B$ and $C$.
2: **repeat**
3:
$$B = B \odot \frac{C^T AC}{C^T CBC^T C}.$$
4:
$$C = C \odot (\frac{ACB}{CBC^T CB})^{\frac{1}{4}} \quad (21)$$
5: **until** convergence
6: Let $D$ be a diagonal matrix s.t. $D_{jj} = \frac{1}{\sum_i C_{ij}}$.
7: $C = CD$
8: $B = D^{-1}BD^{-1}$

Following the way to derive the soft CLGA algorithm, we derive the balanced CLGA algorithm. Since Lemma 4.7 and Theorem 4.6 still hold true for the optimization in (20), the balanced CLGA has the same updating rule for $B$ as the soft CLGA algorithm. By dropping off the term $\alpha||C\mathbf{1} - \mathbf{1}||^2$ in (14), the theorems similar to lemmas 4.5 and 4.6 can be obtained to derive a simpler updating rule for $C$ (details are omitted due to the space limit). The balanced CLGA algorithm is summarized in Algorithm 3. Similarly, by enforcing constraints on $B$, it is easy to obtain other versions of the balanced CLGA algorithm, such as balanced IGP and balanced GGP.

## 5 Experimental Results

In this section, we present experimental results to show the effectiveness of various CLGA algorithms.

### 5.1 Data Sets and Parameter Setting

The data sets used in the experiments include synthetic graphs with different types of community structures and real graphs for text mining and social network analysis.

The synthetic graphs are 0-1 graphs generated based on Bernoulli distribution. The distribution parameters to generate the graphs are listed in the second column of Table 2 as matrices. In a parameter matrix $P$, $P_{ij}$ denotes the probability that the nodes in the $i$th community are connected to the nodes in the $j$th community. For example, in graph W1, the nodes in community 1 are connected to the nodes in community 2 with probability $0.1$ and the nodes within communities are connected to each other with probability 0. The graph G2 has ten communities mixing with strongly intra-connected and weakly intra-connected communities. Due to the space limit, its distribution parameters are omitted here. Totally G2 has $5000$ nodes and about $2.1$ million edges.

The graphs based on the text data have been widely used to test graph learning algorithms [11, 10, 35]. We use various data sets from 20-newsgroups [21], WebACE and TREC [1] to construct the real graphs. The data are preprocessed by removing the stop words and each document is represented by a term-frequency vector using TF-IDF weights. Then we construct a graph for each data set such that each node denotes a document and the edge weight denotes the cosine similarity between documents. A summary of all the data sets to construct graphs used in this paper is shown in Table 3, in which $n$ denotes the number of nodes in a graph, $k$ denotes the number of true communities, and *balance* denotes the size ratio of the smallest community to the largest community. Besides the graphs constructed directly from these data sets, we also construct three graphs with virtual nodes, v-tr23, v-tr45 and v-NG1-20, for three relatively difficult data sets. $5\%n$ virtual nodes are added into the original graphs to incorporate supervised information. To simulate the concept documents provided by domain expert in real applications, the "mean" documents of the communities, which contain popular words of the corresponding communities, are used as virtual documents.

For performance measure, we elect to use the Normalized Mutual Information (NMI) [32] between the resulting community labels and the true community labels, which is a standard way to measure the cluster quality. For the number of communities $k$, we simply use the number of the true communities, since how to choose the optimal number of communities is a nontrivial model selection problem and beyond the scope of this paper.

Three versions of algorithms under different CLGA models as listed in Table 1 are tested in the experiments. We use "H-", "S-" and "B-" to represent hard, soft and balanced versions, respectively. Two representative graph

**Table 2. Summary of the graphs with general communities**

| Graph | Parameter | n | k |
|---|---|---|---|
| S1 | $\begin{bmatrix} 0.6 & 0.3 & 0.3 \\ 0.3 & 0.5 & 0.3 \\ 0.3 & 0.3 & 0.5 \end{bmatrix}$ | 1500 | 3 |
| W1 | $\begin{bmatrix} 0 & 0.1 & 0.1 \\ 0.1 & 0 & 0.3 \\ 0.1 & 0.3 & 0 \end{bmatrix}$ | 1500 | 3 |
| G1 | $\begin{bmatrix} 0.3 & 0.2 & 0.3 \\ 0.2 & 0 & 0.2 \\ 0.3 & 0.2 & 0 \end{bmatrix}$ | 1500 | 3 |
| G2 | $[0, 1]^{10\times 10}$ | 5000 | 10 |

**Table 3. Summary of graphs based on text datasets**

| Name | n | k | Balance | Source |
|---|---|---|---|---|
| tr11 | 414 | 9 | 0.046 | TREC |
| tr23 | 204 | 6 | 0.066 | TREC |
| tr45 | 690 | 10 | 0.0856 | TREC |
| NG1-3 | 1600 | 3 | 0.5 | 20-newsgroups |
| NG1-20 | 14000 | 20 | 1.0 | 20-newsgroups |
| k1b | 2340 | 6 | 0.043 | WebACE |

**Table 4. NMI scores on graphs of general communities**

| Algorithm | S1 | W1 | G1 | G2 |
|---|---|---|---|---|
| SGC | 0.9902 | 0.4551 | 0.5086 | 0.6125 |
| METIS | 0.9811 | 0.0108 | 0.1495 | 0.6391 |
| H-GGP | **1.0000** | 0.0582 | 0.0151 | 0.6965 |
| H-GBCL | 0.0031 | **1.000** | 0.7932 | 0.8837 |
| H-GCL | 0.9103 | 0.9306 | 0.6531 | **0.9133** |
| S-GGP | **1.0000** | 0.0006 | 0.0009 | 0.5802 |
| S-GBCL | 0.0025 | 0.9270 | 0.8100 | 0.6494 |
| S-GCL | 0.9211 | 0.9295 | 0.8144 | 0.7314 |
| B-GGP | **1.0000** | 0.0002 | 0.0008 | 0.5832 |
| B-GBCL | 0.0011 | 0.9091 | 0.9976 | 0.7128 |
| B-GCL | 0.9666 | 0.9085 | **1.0000** | 0.7695 |

learning algorithms are selected as comparisons. Spectral approaches have been applied to a wide range of graph learning tasks from regular graph to bi-partite[10] and k-partite graph learning [23]. In this study, we select to use Spectral Graph Clustering (SGC) [26] that is generalization of a number of graph learning algorithms. SGC learns link-pattern based community structure by embedding community structures into eigen-space and discovering them by the eigenvectors. Another comparison is the classic multilevel graph partitioning algorithm, METIS [17].

### 5.2 Results and Discussion

Table 4 shows the NMI scores of the eleven algorithms on the graphs listed in Table 2. Each NMI score is the average of ten test runs. The graph S1 has three strongly intra-connected communities. We observe that all the GGP algorithms provide perfect scores on S1. However, the GBCL algorithms totally fail, since their goal is to learn weakly intra-connected communities which do not exist in S1. The GCL algorithms perform not as good as the GGP algorithms. The possible reason is that they do not focus on strongly intra-connected communities and hence have more local optimal solutions to converge. The graph W1 consists of three weakly intra-connected communities. This time the GBCL algorithms provide perfect or nearly perfect performance scores. METIS totally fails, since it only looks for strongly intra-connected communities. SGC identifies part of community structures but its performance is not satisfactory. Both G1 and G2 are graphs of mixed-type community structures. Since there is only one strongly intra-connected community in G1, it is still difficult for the GGP algorithms and METIS to identify it. The graph G2 is a large graph of ten communities consisting of four strongly intra-connected communities and six weakly intra-connected communities. The GGP and METIS algorithms do not totally fail on G2, since they could learn strongly intra-connected part from G2. The GBCL algorithms could learn weakly intra-connected part from G2. However, overall the GCL algorithms perform better especially on G1 and G2, since they are capable of learning general communities of various types. In summary, the CLGA model provides a family of algorithms for effectively learning general link-pattern based community as well as specific types of community.

Table 5 shows the NMI scores of the nine algorithms. Since the graphs mainly consist of strongly intra-connected communities, GBCL algorithms are not appropriate. Hence IGP (GGP works similarly to IGP and their result are omitted) and GCL results are reported in comparison with SGC, METIS and a state-of-the-art document clustering algorithm, VMF, which is based on von Mises-Fisher distribution and was reported as one of the best document clustering algorithm [3]. We observe that although there is no single winner on all the graphs, for most graphs CLGA algorithms perform better than or close to SGC, METIS and VMF. The CLGA algorithms provide the best performance on eight out of the nine graphs. By adding virtual nodes into the graphs, the community structures are reinforced by link patterns induced by the virtual nodes. Although all the algorithms benefit from the virtual nodes, CLGA algorithms always achieve the best performance on the graphs with virtual nodes. For example, S-GCL provides the best performance on v-tr23 and increases the performance about $50\%$ by making use of the virtual nodes. Hence, when learning the communities of documents (with or without supervised information), the CLGA model provides a family of new algorithms which are competitive compared with the existing state-of-the-art graph learning algorithms and document clustering algorithm.

We also run the GCL algorithm on the actor graph based on IMDB movie data set for a case study of social network analysis. We formulate a graph of 20000 nodes, in which each node represents an actor and the edges denote collaboration between actors. However, we delete all the links between the actors of the same gender, i.e., the links between male actors and the links between female actors. Although there is no ground truth for the community structure of this graph, the GCL algorithm provides a large number of meaningful communities, which represent the major cast members of one movie or movie series. For example, Table 6 shows Community 11 consisting of 10 male actors and Community 87 consisting of 12 female actors and the community structure shows that the two communities are strongly related to each other. In fact, all these actor/actress are from the movies series "American Pie". Although the links within the communities are missing, these communities are still identified by the GCL algorithm.

## 6 Conclusions

In this paper, we propose a general model based on graph approximation to learn link-pattern based community structures from a graph. The model demonstrates a good theoretic generalization by unifying the traditional graph partitioning objectives and providing a new view to understand

**Table 5. NMI scores on graphs of text data**

| Data | VMF | SGC | METIS | H-IGP | H-GCL | S-IGP | S-GCL | B-IGP | B-GCL |
|---|---|---|---|---|---|---|---|---|---|
| tr11 | 0.5990 | 0.5848 | 0.5708 | 0.5902 | 0.6048 | **0.6070** | 0.5941 | 0.6016 | 0.5959 |
| tr23 | 0.2420 | 0.3003 | 0.2454 | **0.3223** | 0.2158 | 0.3065 | 0.3011 | 0.2905 | 0.2894 |
| tr45 | 0.4821 | 0.5224 | 0.4290 | **0.5570** | 0.4991 | 0.4753 | 0.4793 | 0.5068 | 0.5107 |
| NG1-3 | 0.5611 | 0.5515 | 0.4998 | 0.4574 | 0.5101 | **0.6132** | 0.6104 | 0.5840 | 0.5799 |
| NG1-20 | 0.5102 | 0.4770 | **0.5343** | 0.4977 | 0.4015 | 0.5131 | 0.5102 | 0.5124 | 0.5211 |
| k1b | 0.4998 | 0.4758 | 0.5058 | 0.5066 | 0.5007 | **0.5146** | 0.5087 | 0.4946 | 0.4903 |
| v-tr23 | 0.4001 | 0.3805 | 0.3530 | 0.3873 | 0.4147 | 0.4606 | **0.4648** | 0.4265 | 0.4492 |
| v-tr45 | 0.6095 | 0.6225 | 0.6050 | 0.6184 | 0.6205 | 0.6163 | **0.6304** | 0.5953 | 0.5967 |
| v-NG1-20 | 0.5468 | 0.5597 | 0.5684 | 0.5873 | 0.5954 | 0.6031 | 0.6208 | 0.6177 | **0.6336** |

**Table 6. Two communities from the actor graph.**

| Community 11 |
|---|
| Jason Biggs, James DeBello,Chris Klein, Thomas Ian Nicholas, Seann William Scott, Eugene Levy, Chris Owen, Eric Lively, Joseph D. Reitman, James B. Rogers |
| Community 87 |
| Alyson Hannigan, Shannon Elizabeth, Tara Reid, Mena Suvari, Jennifer Coolidge, Natasha Lyonne, Lisa Arturo, Joelle Carter, Christina Milian, Eden Riegel, Lee Garlington, Joanna Garcia |

the graph partitioning problem. Under this model, we derive three novel algorithms to learn the general community structures from a graph, which cover three main versions of unsupervised learning algorithms, hard, soft and balanced versions, to provide a complete family of community learning algorithms. Besides the theoretic analysis, extensive experiments on both real and synthetic graphs also demonstrate the effectiveness and the great potential of the proposed model and algorithms.

## 7 Acknowledgement

This work is supported in part by NSF (IIS-0535162), AFRL (FA8750-05-2-0284), and AFOSR (FA9550-06-1-0327), and a research internship at Google Research.

## References

[1] http://trec.nist.gov/.
[2] E. Airoldi, D. Blei, E. Xing, and S. Fienberg. Mixed membership stochastic block models for relational data with application to protein-protein interactions. In *ENAR-2006*.
[3] A. Banerjee, I. S. Dhillon, J. Ghosh, and S. Sra. Generative model-based clustering of directional data. In *KDD'03)*, pages 19–28.
[4] T. N. Bui and C. Jones. A heuristic for reducing fill-in in sparse matrix factorization. In *PPSC*, pages 445–452, 1993.
[5] P. K. Chan, M. D. F. Schlag, and J. Y. Zien. Spectral k-way ratio-cut partitioning and clustering. In *DAC '93*, pages 749–754, 1993.
[6] A. Clauset, M. E. J. Newman, and C. Moore. Finding community structure in very large networks. *Eur. Phys*, 2004.
[7] I. Dhillon, Y. Guan, and B. Kulis. A unified view of kernel k-means, spectral clustering and graph cuts. Technical Report TR-04-25, University of Texas at Austin, 2004.
[8] I. Dhillon, Y. Guan, and B. Kulis. A fast kernel-based multilevel algorithm for graph clustering. In *KDD '05*, 2005.
[9] I. Dhillon, Y. Guan, and B. Kulis. A fast kernel-based multilevel algorithm for graph clustering. In *KDD '05*, pages 629–634, 2005.
[10] I. S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *KDD*, pages 269–274, 2001.
[11] 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 *Proceedings of ICDM 2001*, pages 107–114, 2001.
[12] S. E. Fienberg, M. M. Meyer, and S. Wasserman. Satistical analysis of multiple cociometric relations. *Journal of American Satistical Association*, 80:51–87, 1985.
[13] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. In *National Academic Science*, 2001.
[14] B. Hendrickson and R. Leland. A multilevel algorithm for partitioning graphs. In *Supercomputing '95*, page 28, 1995.
[15] M. Henzinger, R. Motwani, and C. Silverstein. Challenges in web search engines. In *Proc. of the 18th International Joint Conference on Artificial Intelligence*, pages 1573–1579, 2003.
[16] P. Hoff, A. Rafery, and M. Handcock. Latent space approaches to social network analysis. *Journal of American Satistical Association*, 97:1090–1098, 2002.
[17] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput.*, 20(1):359–392, 1998.
[18] B. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, 49(2):291–307, 1970.
[19] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.
[20] R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Trawling the Web for emerging cyber-communities. *Computer Networks*, 31(11–16), 1999.
[21] K. Lang. News weeder: Learning to filter netnews. In *ICML*, 1995.
[22] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *NIPS*, pages 556–562, 2000.
[23] B. Long, Z. M. Zhang, X. Wu, and P. S. Yu. Spectral clustering for multi-type relational data. In *ICML'06*, 2006.
[24] M. E. J. Newman. Detecting community structure in networks. *Eur. Phys*, 38:321–330, 2003.
[25] M. E. J. Newman. Fast algorithm for detecting community structure in networks. *Phys. Rev., E,*, 2004.
[26] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 14*, 2001.
[27] P. Reddy and M. Kitsuregawa. Inferring web communities through relaxed cocitation and dense bipartite graphs. In *Data Basis Engineering Workshop*, 2001.
[28] R. Salakhutdinov and S. Roweis. Adaptive overrelaxed bound optimization methods. In *ICML'03*, 2003.
[29] J. P. Scott. *Social Network Analysis: A Handbook*. SAGE Publications, January 2000.
[30] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.
[31] T. Snijders. Markov chain monte carlo estimation of exponential random graph models. *Journal of Ssocial Structure*, 2002.
[32] A. Strehl and J. Ghosh. Cluster ensembles – a knowledge reuse framework for combining partitionings. In *AAAI 2002*, pages 93–98, 2002.
[33] A. Y. Wu, M. Garland, and J. Han. Mining scale-free networks using geodesic clustering. In *KDD '04*, pages 719–724. ACM Press, 2004.
[34] S. Yu and J. Shi. Multiclass spectral clustering. In *ICCV'03.*, 2003.
[35] H. Zha, C. Ding, M. Gu, X. He, and H. Simon. Bi-partite graph partitioning and data clustering. In *ACM CIKM'01*, 2001.