ABC: Attributed Bipartite Co-clustering

Junghoon Kim
Nanyang Technological University
Singapore
junghoon001@e.ntu.edu.sg

Kaiyu Feng*
Beijing Institute of Technology
China
fengky@bit.edu.cn

Gao Cong*
Nanyang Technological University
Singapore
gaocong@ntu.edu.sg

Diwen Zhu
Alibaba Group
China
diwen.zdw@alibaba-inc.com

Wenyuan Yu
Alibaba Group
China
wenyuan.ywy@alibaba-inc.com

Chunyan Miao
Nanyang Technological University
Singapore
ascymiao@ntu.edu.sg

ABSTRACT

Finding a set of co-clusters in a bipartite network is a fundamental and important problem. In this paper, we present the Attributed Bipartite Co-clustering (ABC) problem which unifies two main concepts: (i) bipartite modularity optimization, and (ii) attribute cohesiveness. To the best of our knowledge, this is the first work to find co-clusters while considering the attribute cohesiveness. We prove that ABC is NP-hard and is not in APX, unless P=NP. We propose three algorithms: (1) a top-down algorithm; (2) a bottom-up algorithm; (3) a group matching algorithm. Extensive experimental results on real-world attributed bipartite networks demonstrate the efficiency and effectiveness of our algorithms.

PVLDB Reference Format:
Junghoon Kim, Kaiyu Feng, Gao Cong, Diwen Zhu, Wenyuan Yu, and Chunyan Miao. ABC: Attributed Bipartite Co-clustering. PVLDB, 15(10): 2134 - 2147, 2022.
doi:10.14778/3547305.3547318

1 INTRODUCTION

Many systems can be modeled as a bipartite network [32] such as metabolic network [35], human sexual network [44], document-word network [21], and collaboration networks [74]. A bipartite network has two distinct sets of nodes U and V and a set of edges E connecting nodes in U and V. It has many interesting applications. An attributed bipartite network is a bipartite network where each node is associated with a set of attributes. Based on the attributes, we can measure the closeness of the nodes. For instance, a user-location check-in network [39] is an attributed bipartite network, where the nodes are divided into users and locations and an edge represents a user making a check-in at a location; each user is associated with a set of friends, and each location is associated with its coordinates and description.

Co-clustering is an important and fundamental problem for a bipartite network [14, 21, 22]. Generally, co-clustering a bipartite network is to simultaneously group vertices in clusters of each vertex set by performing simultaneous clustering of the rows and columns of a matrix. Co-clustering has been extensively studied in the context of gene expression data and document-word networks. There are many approaches for co-clustering such as bipartite modularity optimization [1], information theoretic co-clustering algorithm [22], deep-learning based co-clustering [72], k-means based co-clustering [34]. Among different approaches, matrix-factorization approaches are arguably most widely used [23, 29, 46, 58, 62, 65]. However, these methods require large memory space and are not scalable to large bipartite graphs. In contrast, bipartite modularity optimization aims to maximize the bipartite modularity [6] via iterative alternating optimization [1] or spectral approximation [41] to find the best partitions in a bipartite network. These co-clustering methods focus on only the bipartite network without considering the attributes of nodes; the two sides of each generated co-cluster are densely connected. However, attributes of nodes often provide valuable information that can be used to generate different types of co-clusters such that each side of a co-cluster has similar attributes and the nodes in each side are cohesive. For example, if one side of the bipartite network is users with a social network, we may require that the user nodes of each co-cluster are densely connected to form a community. Unfortunately, previous work fails to consider this.

In this work, we tackle a new problem of attributed bipartite co-clustering (ABC). We define an attributed co-cluster as a subgraph of an attributed bipartite network, in which each side of a co-cluster has similar attributes (patterns) and the two sides are densely connected with each other. Our goal is to find all the attributed co-clusters in an attributed bipartite network.

The example applications of our problem are as follows.

- **Recommendation.** In a music service, the users and the songs form a bipartite graph, where an edge represents a user favors a song. Each user is associated with attributes such as his/her preferred music genres and each music is associated with attributes like artist and genre. Each attributed co-cluster consists of a set of users with similar favorite genres and a set of favored similar songs. Based on an attributed co-cluster, we can make up a playlist with the selected songs and recommend the playlist to the selected users. Note that users who have not indicated any preference for songs or songs that have never been marked as favorite songs are not included in the attributed co-cluster.
- **Finding fraudsters.** In a social commerce service, users and products can make a bipartite graph, where an edge indicates

*Corresponding author
This work is licensed under the Creative Commons BY-NC-ND 4.0 International License. Visit https://creativecommons.org/licenses/by-nc-nd/4.0/ to view a copy of this license. For any use beyond those covered by this license, obtain permission by emailing info@vldb.org. Copyright is held by the owner/author(s). Publication rights licensed to the VLDB Endowment.
Proceedings of the VLDB Endowment, Vol. 15, No. 10 ISSN 2150-8097.
doi:10.14778/3547305.3547318

a user favors (or disfavors) a product. Each user and product is associated with its attributes. Each resulted co-cluster contains a set of users with similar attributes such as purchase pattern or favorite products, and a set of favored (or disfavored) products having similar properties. The fraudsters often share similar attributes and are connected with products with similar attributes, and co-clusters will help to identify a cluster of fraudsters.

- **Bus planning system.** Consider operating a shuttle bus planning system. The users and the companies form a bipartite network, where each user has a home location and each company is associated with its location, and a bipartite edge represents a user who works for a company. ABC will return a set of co-clusters, in which each co-cluster consists of a set of users whose home locations are close and a set of companies which are located closely. Thus, each co-cluster can be a candidate of a bus route.
- **Finding local experts [15, 40]**. In location/event based social networks (LBSNs), such as meetup, ABC can be utilized for finding a set of local expert candidates by identifying a set of co-clusters. Each co-cluster contains cohesive users and a set of close locations. It is helpful to find a set of local experts by checking the users in the co-cluster. Activities of the experts are concentrated on a specific region. For example, in LBSNs, it is named regionalization, and is important for utilizing marketing strategy [37].

Unfortunately, it is challenging to solve the ABC problem as to be explained. It is not feasible to extend existing bi-clustering methods for attributed bipartite networks. One might think we first find co-clusters without considering the attributes, and then apply the post-processing to guarantee the attribute cohesiveness (i.e., each side of a co-cluster has similar attributes). However, as the existing co-clustering methods do not consider the attributes, the nodes in each side of an identified co-cluster may have total different attributes. It is challenging and unknown how to design a sensible post-processing method to solve the ABC problem.

The first challenge of attributed bipartite co-clustering is to define attributed co-clusters for an attributed bipartite network, which is an open problem. For bipartite networks without attributes, bipartite modularity [6, 30, 56] is proposed and has been used for defining co-clusters [1, 2, 41], which often results in better efficiency and co-clusters of better quality compared with other co-clustering models. Therefore, a natural idea is to extend the bipartite modularity to attributed bipartite networks. However, bipartite modularity optimization [1, 2, 41] may miss some important structures due to its characteristics. In this paper, we formulate a new bipartite k-clique ring resolution limit problem which belongs to the resolution limit problem families, and show that the bipartite modularity suffers from the bipartite k-clique ring resolution limit problem. It implies that when we optimize the bipartite modularity, it fails to identify small-sized co-clusters, and thus misses some important structures. To mitigate the problem, in this paper, we propose a new modularity measure called Excess Bipartite Modularity Density (EBMD). We prove that the proposed EBMD does not suffer from the bipartite k-clique ring resolution limit problem. Furthermore, the previous bipartite modularity definition does not take into account the attributes of nodes. We propose to enforce constraints

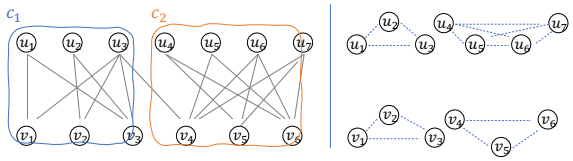


Figure 1: Sample graph with social connection attributes

to incorporate attribute similarity in the definition of co-clusters, which work together with the proposed EBMD.

Second, we prove that the attributed bipartite co-clustering problem based on our EBMD definition is NP-hard and not in APX. Thus, it has no PTAS(Polynomial-Time Approximation Scheme) assuming $P \neq NP$, and also no c-approximation. We design three efficient heuristic algorithms: (1) a bottom-up algorithm; (2) a group-matching algorithm; and (3) a top-down algorithm. These algorithms are designed to handle two major challenges to strike a balance differently: (1) maximizing EBMD; and (2) preserving the attribute cohesiveness. The bottom-up algorithm gives priority to addressing the challenge of maximizing EBMD by iteratively finding seed nodes and adding a set of nodes for preserving the attribute cohesiveness. The group-matching algorithm focuses on the attribute cohesiveness as the main concern by finding sets of seed nodes in each side and then combining them based on maximizing EBMD. Lastly, the top-down algorithm simultaneously considers maximizing EBMD and preserving attribute cohesiveness by iteratively splitting larger co-clusters into small-sized co-clusters while maximizing the EBMD.

The contributions of our work are summarized as follows:

(1) **Defining the attributed bipartite co-clustering ABC problem.** To the best of our knowledge, this is the first work to find a set of co-clusters in an attributed bipartite network by considering the bipartite edges and similarity cohesiveness simultaneously. We prove that the ABC problem is NP-hard and not in APX, unless $P = NP$.
(2) **New bipartite modularity and theoretical analysis.** We prove that the existing bipartite modularity suffers from the bipartite k-clique ring resolution limit problem. To overcome the limitation, we propose a new bipartite modularity EBMD, and prove that EBMD can resolve the problem.
(3) **Solutions.** We propose three solutions to solve the attributed bipartite co-clustering problem effectively and efficiently.
(4) **Extensive Evaluation.** We conduct extensive experiments on real-life networks to demonstrate effectiveness and efficiency. We also show the usefulness of our problem with the case study.

2 THEORETICAL ANALYSIS

In this section, we present the bipartite modularity and propose excess bipartite modularity density with theoretical analysis.

2.1 Bipartite modularity

Definition 1. *(Bipartite network).*
Bipartite network $G = (U, V, E)$ is a network where the nodes are divided into two disjoint sets, U and V, and every edge in E connects a node in U to a node in V.

Definition 2. *(Co-cluster).*
Given a set of nodes c, a subgraph $G[c]$ of a bipartite graph G is a co-cluster if c is densely connected in terms of bipartite edges.

Definition 3. *(Bipartite modularity [6, 54]).*
Given a bipartite network $G = (U, V, E)$ and a set of disjoint co-clusters C, bipartite modularity is defined as follows[1].

$$M_B(G, C) = \sum_{c \in C} M_B(G, c) = \sum_{c \in C} [\frac{l_c}{|E|} - (\frac{d_c^U d_c^V}{|E|^2})] \quad (1)$$

In Equation 1, $M_B(G, c)$ is the cluster-modularity of each co-cluster c, $|E|$ is the sum of the bipartite edges, l_c is the number of edges in co-cluster c, $d_c^U = \sum_{v \in c_U} d_v$, where d_v is the degree of the node v and c_U is the set of nodes in the U side of the co-cluster c, and d_c^V is defined similarly as d_c^U. Intuitively, the bipartite modularity in Equation 1 indicates the difference between a fraction of edges within co-clusters and an expected fraction of the edges. Therefore, higher modularity indicates that there are more intra co-cluster edges than randomly distributed edges, which are reflected by $(\frac{d_c^U d_c^V}{|E|})$ in Equation 1. Note that the randomly distributed edges are reflected by a null model of the modularity, which is defined as $\frac{d_u d_v}{|E|}$ for a pair of node.

Example 1. *Figure 1 shows a bipartite network and two co-clusters. The cluster-modularity of each co-cluster is computed as follows.*

- $M_B(G, c_1) = \frac{7}{18} - \frac{8 \cdot 7}{324} = 0.216$
- $M_B(G, c_2) = \frac{10}{18} - \frac{10 \cdot 11}{324} = 0.216$

Therefore, $M_B(G, C) = M_B(G, c_1) + M_B(G, c_2) = 0.432$.

Assume that we have a co-cluster c of G. In the bipartite modularity optimization problem, to maximize the modularity score, the cluster-modularity of the co-cluster c should be positive, i.e., $\frac{l_c}{|E|} - (\frac{d_c^U d_c^V}{|E|^2}) \geq 0$.

2.2 Excess bipartite modularity density(EBMD)

In this section, we propose a new objective function named excess bipartite modularity density(EBMD). We extend excess modularity density [12], which is designed for a unipartite network, to the bipartite network. EBMD is defined as follows.

Definition 4. *(Excess bipartite modularity density(EBMD)).*
Given a bipartite network $G = (U, V, E)$ and a set of disjoint co-clusters $C = \{c_1, c_2, \cdots, c_i\}$, EBMD is defined as follows.

$$X(G, C) = \sum_{c \in C} [\mathcal{X}_c \frac{l_c}{|E|} - \mathcal{X}_c^2 \frac{d_c^U d_c^V}{|E|^2}] \quad (2)$$

where C is a set of co-clusters, l_c is the number of internal edges in the co-cluster c, $|E|$ is the number of edges, c_U (resp. c_V) is a set of nodes in the U (resp. V) side of the co-cluster c, and d_c^U (resp. d_c^V) is the sum of the degree of the nodes in c_U (resp. c_V). The function $\mathcal{X}_c$ is defined as $\mathcal{X}_c = \frac{l_c}{|c_U||c_V|} - \frac{|E|}{|U||V|}$. Note $\mathcal{X}_c$ can be negative when c does not have sufficient internal edges compared with the whole graph. We denote $[\mathcal{X}_c \frac{l_c}{|E|} - \mathcal{X}_c^2 \frac{d_c^U d_c^V}{|E|^2}]$ as the sub-EBMD.

[1]Note that Equation 3 is cluster-level bipartite modularity [6]

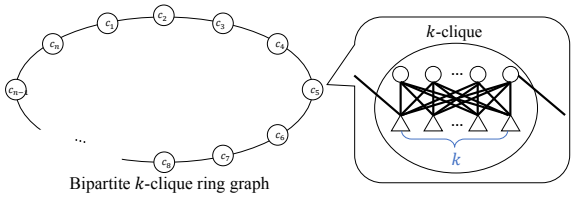


Figure 2: Bipartite k-clique ring graph

Example 2. *The EBMD of the co-clusters in Figure 1 is computed as follows.*

- $\mathcal{X}_{c_1} = \frac{7}{9} - \frac{18}{42} = 0.349$
- $\mathcal{X}_{c_2} = \frac{10}{12} - \frac{18}{42} = 0.404$

Then, EBMD of each co-cluster is as follows.

- $X(G, c_1) = \mathcal{X}_{c_1} \cdot \frac{7}{18} - \mathcal{X}_{c_1}^2 \cdot \frac{8 \cdot 7}{324} = 0.115$
- $X(G, c_2) = \mathcal{X}_{c_2} \cdot \frac{10}{18} - \mathcal{X}_{c_2}^2 \cdot \frac{10 \cdot 11}{324} = 0.169$

Therefore, $X(G, C) = X(G, c_1) + X(G, c_2) = 0.283$.

2.3 Theoretical analysis

We next analyze the classic bipartite modularity and EBMD. We first introduce some definitions.

Definition 5. *(A bipartite k-clique graph).*
A bipartite k-clique graph contains k nodes in U and V sides, respectively. For each $v \in V$ and $u \in U$, there exists an edge (u, v). Hence, there are k^2 bipartite edges.

Definition 6. *(Ring-shape graph).*
A ring-shape graph G consists of n disjoint subgraphs $c_1, c_2, \cdots, c_n$ such that for any two subgraphs c_i and $c_{(i\%n)+1}$ where $1 < i \leq n$, there exists exactly one edge in G that connects a node in c_i and a node in $c_{(i\%n)+1}$.

By combining a bipartite k-clique graph and a ring-shape graph, we define a bipartite k-clique ring graph.

Definition 7. *(Bipartite k-clique ring graph).*
A bipartite k-clique ring graph is a sequence of $n \geq 4$ bipartite k-cliques, denoted by $c_1, \ldots, c_n$, such that for any c_i and $c_{(i\%n)+1}^V$ where $1 \leq i \leq n$, there exists only one edge that connects c_i^U and $c_{(i\%n)+1}^V$.

The resolution limit problem implies that the modularity optimization based approaches may fail to find communities that are smaller than a certain size [10, 11, 53]. This problem is considered as a general phenomenon [64] that describes clustering algorithms failing to identify desirable clusters. As this phenomenon is difficult to conceptualize, many papers [12, 27] utilize clique-based network configurations to show the resolution limit problem since merged cliques clearly present a limitation of the modularity-based approaches. Inspired by this idea, we define a type of resolution limit problem, named bipartite k-clique ring resolution limit which belongs to the resolution limit problem families.

Definition 8. *(bipartite k-clique ring resolution limit).*
Given a function f, we call that optimizing f suffers from bipartite k-clique ring resolution limit if it fails to find n bipartite k-cliques.

We proceed to show that optimizing the bipartite modularity suffers from the *bipartite k-clique ring resolution limit* and EBMD

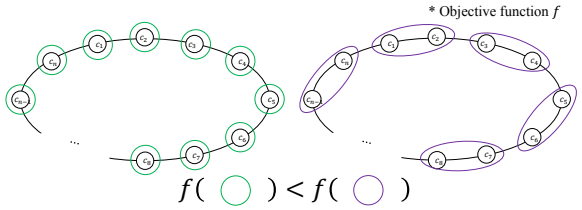


Figure 3: Bipartite k-clique ring resolution limit problem

does not suffers from it, i.e., optimizing EBMD is *bipartite k-clique ring resolution free*. In the proof, we compare two cases: (1) an objective function prefers to find a set of k-cliques; (2) an objective function prefers to find a set of merged cliques, i.e., two adjacent cliques are merged. We define the following notations.

- $|E| = n + nk^2$
- $M_B(S) = n(\frac{k^2}{n+nk^2} - \frac{(k+1)^2}{(n+nk^2)^2})$
- $M_B(M) = \frac{n}{2}(\frac{2k^2+1}{n+nk^2} - \frac{(2k+2)^2}{(n+nk^2)^2})$
- $X(S) = n(\alpha\frac{k^2}{n+nk^2} - \alpha^2\frac{(k+1)^2}{(n+nk^2)^2})$
- $\alpha = \frac{k^2}{k^2} - \frac{n+nk^2}{n^2k^2} = 1 - \frac{n+nk^2}{n^2k^2}$
- $X(M) = \frac{n}{2}(\beta\frac{2k^2+1}{n+nk^2} - \beta^2\frac{(2k+2)^2}{(n+nk^2)^2})$
- $\beta = \frac{2k^2+1}{4k^2} - \frac{n+nk^2}{n^2k^2}$

Theorem 1. *Optimizing the bipartite modularity suffers from the bipartite k-clique ring resolution limit.*

Proof. We assume that bipartite modularity does not suffer from the bipartite k-clique ring resolution limit, and check whether the assumption is contradictory.

$$\begin{aligned} & M_B(S) > M_B(M) \\ \Rightarrow\ & n(\frac{k^2}{n+nk^2} - \frac{(k+1)^2}{(n+nk^2)^2}) > \frac{n}{2}(\frac{2k^2+1}{n+nk^2} - \frac{4(k+1)^2}{(n+nk^2)^2}) \\ \Rightarrow\ & \frac{2k^2}{2(n+nk^2)} - \frac{2k^2+1}{2(n+nk^2)} > -\frac{(k+1)^2}{(n+nk^2)^2} \\ \Rightarrow\ & \frac{1}{2} < \frac{(k+1)^2}{n+nk^2} \end{aligned} \tag{3}$$

We notice that $\frac{1}{2} < \frac{(k+1)^2}{(n+nk^2)}$ does not when n is large. Suppose that $n = k^2$. Then, we get the following result.

$$\frac{1}{2} < \frac{(k+1)^2}{n+nk^2} \Rightarrow \frac{1}{2} < \frac{(k+1)^2}{k^2+k^4} \tag{4}$$

We notice that for any positive value $k \geq 2$, Equation 4 does not hold. It means that our assumption leads to a contradiction, and implies that the bipartite modularity optimization suffers from the bipartite k-clique ring resolution limit problem. □

We next check whether EBMD suffers from the bipartite k-clique ring resolution limit.

Theorem 2. *Optimizing the EBMD is bipartite k-clique ring resolution free.*

Proof. We assume that EBMD prefers merged bipartite cliques, i.e., suffers from bipartite k-clique ring resolution limit problem.

$$\begin{aligned} & X(S) < X(M) \\ \Rightarrow\ & n(\alpha\frac{k^2}{n+nk^2} - \alpha^2\frac{(k+1)^2}{(n+nk^2)^2}) < \frac{n}{2}(\beta\frac{2k^2+1}{n+nk^2} - \beta^2\frac{(2k+2)^2}{(n+nk^2)^2}) \\ \Rightarrow\ & \frac{2k^2(\alpha-\beta)-\beta}{2} < \frac{(k+1)^2(\alpha^2-2\beta^2)}{n+nk^2} \end{aligned} \tag{5}$$

$$\begin{aligned} & \frac{2k^2(1-\frac{2k^2+1}{4k^2}) - (\frac{2k^2+1}{4k^2} - \frac{n+nk^2}{n^2k^2})}{2} < \\ \Rightarrow\ & \frac{(k+1)^2((1-\frac{n+nk^2}{n^2k^2})^2 - 2(\frac{2k^2+1}{4k^2} - \frac{n+nk^2}{n^2k^2})^2)}{n(k^2+1)} \end{aligned} \tag{6}$$

Now, we have two scenarios. The first scenario is to assume that when n is sufficiently large such as $n = \infty$ in Equation 6.

$$\begin{aligned} & \Rightarrow 2k^2(1 - \frac{2k^2+1}{4k^2}) < \frac{2k^2+1}{4k^2} \\ & \Rightarrow 8k^4 - 4k^2 - 2 < 2k^2 + 1 \end{aligned} \tag{7}$$

Equation 7 does not hold when $k \geq 2$. It implies that when $n = \infty$, optimizing EBMD does not suffer from the bipartite k-clique ring resolution limit. We next assume that when n is the smallest value, i.e., $n = 4$ in Equation 6.

$$\begin{aligned} & \Rightarrow \frac{\frac{2k^2-1}{2} - \frac{1}{4}}{2} < \frac{(k+1)^2((\frac{3k^2-1}{4k^2})^2 - \frac{1}{8})}{4(k^2+1)} \\ & \Rightarrow (k^2+1)(2k^2 - \frac{3}{2}) < (k+1)^2((\frac{3k^2-1}{4k^2})^2 - \frac{1}{8}) \end{aligned} \tag{8}$$

We notice that Equation 8 does not always hold if $k \geq 2$. Thus, EBMD is bipartite k-clique ring resolution free. □

Figure 3 shows the bipartite k-clique ring resolution limit problem. If an objective function prefers purple co-clusters to green co-clusters, the objective function can be considered as suffering the bipartite k-clique ring resolution limit problem.

Example 3. *Suppose that we have a bipartite k-clique ring graph with $n = 12$ and $k = 2$. Note that each clique has four nodes and has four internal edges and two outgoing edges. Let suppose two scenarios:*

(1) co-clusters S : each clique is a co-cluster;
(2) co-clusters M : two adjacent cliques form a co-cluster, i.e., $c_1 \cup c_2$ is a co-cluster

Then, the bipartite modularity is as follows.

- $B_M(S) = 12 \cdot (\frac{4}{60} - \frac{5 \cdot 5}{60^2}) = 0.716$
- $B_M(M) = 6 \cdot (\frac{9}{60} - \frac{10 \cdot 10}{60^2}) = 0.733$

We notice that co-clusters S have a small bipartite modularity score even if each co-cluster is a bipartite clique. It indicates that the bipartite modularity optimization suffers from the resolution limit problem.

Next, to compute EBMD, let y denote a co-cluster in S and Y denote a co-cluster in M. We first calculate $\mathcal{X}_y$ and $\mathcal{X}_Y$:

- $\mathcal{X}_y = \frac{4}{4} - \frac{60}{24 \cdot 24} = 0.895$
- $\mathcal{X}_Y = \frac{9}{16} - \frac{60}{24 \cdot 24} = 0.458$

Therefore, we get the following result. We notice that EBMD does not miss an important structure in this example.

Table 1: Notations

Notation	Description
G	Attributed bipartite network
$G[H]$	the subgraph G induced by nodes H
μ	threshold for minimum number of close nodes
ϵ	threshold to determine similar nodes
E_U	$\bigcup_{\forall u,v \in U}\{x,y\}$if $F(x,y) \geq \epsilon$
$F(u,v)$	similarity between nodes u and v
$N(u)$	A set of neighbor nodes based on bipartite edges
$S(u)$	A set of similar nodes based on ϵ, A, F

- $X(S) = 12 \cdot (0.895 \cdot \frac{4}{60} - (0.895^2)\frac{5 \cdot 5}{60^2}) = 0.649$
- $X(M) = 6 \cdot (0.458 \cdot \frac{9}{60} - (0.458^2)\frac{10 \cdot 10}{60^2}) = 0.377$.

3 PROBLEM DEFINITION

We propose a new bipartite modularity to overcome the limitation of the classic bipartite modularity. But we still face the challenge of incorporating attributes in the definition of co-cluster. We next present how we address this challenge. The main notations used in this paper are summarized in Table 1.

Definition 9. *(Attributed bipartite network(ABN)). Attributed bipartite network $G = (U, V, E, A, F)$ is a bipartite network in which every node $u \in U$ (or $v \in V$) is associated with a set of attributes $A(u)$. Given two nodes u_1, u_2 from U (or V), function F evaluates the similarity $F(u_1, u_2)$ based on their attributes.*

Our idea. Previous work does not take similarity of nodes or their connection into consideration in defining co-clusters although the attributes of nodes provide valuable information for defining co-clusters. For example, if the nodes U of bipartite network G are users with social network links, we expect that each co-cluster has two properties: 1) the U side and V side are densely connected by bipartite edges as it is considered in the previous work; and 2) the nodes in U side (resp. V) are densely connected or share similar attributes. Note that our proposed solutions should be orthogonal to the similarity function, which can be any function to be specified by end users. This is important to cope with various types of data to meet the needs in different scenarios. For example, the similarity function could simply correspond to the social link if U is a social network, i.e., the similarity is 1 if there is a link between u_1 and u_2, and 0 otherwise.

Definition 10. *(Similarity constraint). Given a set of nodes $H \subseteq V$ (or $H \subseteq U$), function F, and similarity thresholds ϵ and μ, H satisfies the similarity constraint if*

- *$\forall h \in H$ of G, there exist at least μ nodes $\{h_1, \ldots, h_\mu\}$ in H, such that $F(h, h_i) \geq \epsilon$ for every $i \in [1, \mu]$.*
- *$\forall h, h' \in H$, there is a sequence of nodes $h_1, h_2, \cdots h_x$ where $h_1 = h$ and $h_x = h'$, s.t. $F(h_i, h_{i+1}) \geq \epsilon$ for $i \in [1, x-1]$.*

Intuitively, the first condition requests that each node should have at least μ similar nodes in a co-cluster; The second condition is to guarantee that each pair of nodes in a co-cluster is similarity-based reachable The two conditions can be understood from a different angle as follows: We construct a similarity graph $G_V = (V, E_V)$, where the edges E_V are formed based on the ϵ and function F, i.e., $(v_1, v_2) \in E_V$ iff $F(v_1, v_2) \geq \epsilon$. Then, a set $H \subseteq V$ of nodes satisfies the similarity constraint if the subgraph induced by H is connected (similarity-based reachable) and its minimum degree is at least μ. We call a maximal subgraph satisfying the similarity constraint as μ-core which is a k-core [63] in a similarity graph.

Definition 11. *(Attributed Co-cluster). Let $H = (H_U, H_V)$. A subgraph $G[H]$ of an attributed bipartite network G is an attributed co-cluster if $G[H]$ is a co-cluster and H_U and H_V satisfy similarity constraint.*

Problem definition 1. *(Attributed Bipartite Co-Clustering(ABC)). Given an attributed bipartite network $G = (U, V, E, A, F)$, similarity thresholds μ, ϵ, attributed bipartite co-clustering (ABC) is to find a set C of attributed co-cluster while maximizing EBMD $X(G, C)$.*

Example 4. *In Figure 1, we notice that there are two attributed bipartite co-clusters c_1 and c_2. Each attributed bipartite co-cluster is densely connected and satisfy the similarity constraint when $\mu = 2$.*

Remark 1. *Note that our ABC problem is not a graph partitioning problem, i.e., we aim to find a set of meaningful attributed co-clusters. It is possible that some entities may not belong to a specific attributed co-cluster if it does not satisfy the similarity constraint or it does not have a benefit to maximize EBMD.*

Theorem 3. *The problem of solving ABC is NP-hard when $\mu \geq 3$*

Proof. We first introduce $MSMD_k$ problem [3]. Given a graph $G = (V, E)$ and positive integer k, the goal of $MSMD_k$ is to find a minimum subgraph of which its minimum degree is larger than or equal to k. It is known that the problem is NP-hard and not in APX-hard for any $k \geq 3$. Here we introduce a reduction from an instance of $MSMD_k$ to the instance of ABC problem. Note that we consider a similarity graph in U and V sides, i.e., we consider both sides as a graph. An edge of U side (or V side) indicates that two nodes are similar. Otherwise, two nodes are not connected. Initially, we construct a new graph G' from G. The procedure is as follows.

- Make $U = G_1 \cup G_2 \cup G_3$ where $G_1 = G_2 = G_3 = G$ and $V = G_4 \cup G_5 \cup G_6$ where $G_4 = G_5 = G_6 = G$
- From G_1 to G_4, we make a bipartite clique in G'.
- From G_3 to G_6, we make a bipartite clique in G'.
- For both G_1 and G_4, we make a clique.
- For both G_2 and G_5, they have the same edges E of G.
- For both G_3 and G_6, there is no edge.
- From G_1 to G_2, we make a bipartite clique (any nodes in G_1 is similar to any nodes in G^2).
- From G_4 to G_5, we make a bipartite clique (any nodes in G_4 is similar to any nodes in G^5).

Figure 4 shows a new bipartite graph. Here, we have an instance of ABC problem, i.e., $I_{ABC} = (G', \mu = |V| + k)$ where $|V|$ is the number of nodes in the original graph G. We can easily notice that a solution must contain G_1 and G_4 and include at least $k+1$ nodes in G_2 and G_5 to satisfy the similarity constraint since $\mu = |V| + k$. Note that G_3 and G_6 cannot form a co-cluster since it does not satisfy a similarity constraint. Here, let denote Y_{G_2} (or Y_{G_5} as a subgraph in G_2 (and G_5) to satisfy the similarity constraint and $N = |V|$. To specify a solution, we denote $S = \{G_1 \cup Y_{G_2}, G_4 \cup Y_{G_5}\}$. We often use Y instead of using Y_{G_2} or Y_{G_5} since $|Y_{G_2}| = |Y_{G_5}|$.

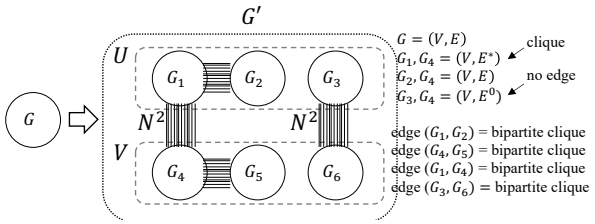


Figure 4: Reduction setting

$$\mathcal{X} = \frac{N^2}{(N+Y)^2} - \frac{2N^2}{(3N)^2} = \frac{N^2}{(N+Y)^2} - \frac{2}{9} \tag{9}$$

$$X(G', S) = (\frac{N^2}{(N+Y)^2} - \frac{2}{9})\frac{1}{2} - (\frac{N^2}{(N+Y)^2} - \frac{2}{9})^2\frac{1}{4} \tag{10}$$

Here, we wonder when the size of Y is increased, how is EBMD changed? To answer the question, we check the derivative of $X(G', S)$.

$$\frac{\mathrm{d}X(G', S)}{\mathrm{d}Y} = -\frac{N^2(11Y^2 + 22NY + 2N^2)}{9(Y+N)^5} < 0 \tag{11}$$

Since the derivative of $\frac{\mathrm{d}X(G', S)}{\mathrm{d}Y} < 0$, $X(G', S)$ is always decreasing, i.e., it prefers small-size subgraph to maximize the EBMD. It implies a set of nodes Y in G^2 can be a solution of $MSMD_k$ since they satisfy the degree constraint and the size is minimized for maximizing EBMD. Since finding an exact solution of $MSMD_k$ is NP-hard and the reduction process can be done in polynomial time, solving our ABC problem is also NP-hard when $\mu \geq 3$. □

Theorem 4. *ABC problem is not in APX for $\mu \geq 3$, unless $P = NP$.*

Proof. We give an L-reduction [18] from MSMD_k(Minimum Subgraph of Minimum Degree) [3, 4] to an instance of ABC problem in polynomial time. MSMD_k problem is a minimization problem and is not in APX, when $k \geq 3$ and unless P=NP [3]. For convenience, we consider the inverse problem of ABC to make the minimization problem. Since the problem MSMD_k is not in APX when $k \geq 3$ and unless $P = NP$, our claim is that our ABC is also not in APX, unless $P = NP$. In the proof of the theorem 3, we present a reduction from MSMD_k to our problem. Now, we are ready to show L-reduction from MSMD_k to ABC. We switch ABC for convenience from maximization to minimization for preserving APX in the reduction. New problem $\overline{\mathrm{ABC}}$ is to minimize the inverse EBMD while EBMD is positive. This is reasonable since we can find only one subgraph as a result (See Proof of Theorem 3). In this proof, we use the following notations (See Figure 5).

- $N = |V|$.
- x is an instance of MSMD_k.
- $f(x)$ is an instance of $\overline{\mathrm{ABC}}$.
- if y' is a solution to $f(x)$, then $g(y')$ is a solution to x.
- $OPT(.)$ defines the cost of the optimal solution for the given instance of a problem.
- $Sol'_{\mathrm{MSMD}_k}(g(y'))$ is the cost of the solution $g(y')$ for the instance x of MSMD_k.
- $Sol'_{\overline{\mathrm{ABC}}}(y')$ is the cost of the solution y' for the instance $f(x)$ of $\overline{\mathrm{ABC}}$.

Problem	Instance	Solution	Cost
$MSMD_k$	x	$g(y)$	Size
ABC	$f(x)$	y'	EBMD

Figure 5: L-reduction notations

There is an L-reduction (f, g) if two positive constants α and β exist such that

$$OPT_{\overline{\mathrm{ABC}}}(f(x)) \leq \alpha OPT_{\mathrm{MSMD}_k}(x) \tag{12}$$

$$OPT_{\mathrm{MSMD}_k}(x) - Sol'_{\mathrm{MSMD}_k}(g(y')) \leq \beta(OPT_{\overline{\mathrm{ABC}}}(f(x)) - Sol'_{\overline{\mathrm{ABC}}}(y')) \tag{13}$$

We find that there exist such $\alpha = 4N^2$ and $\beta = 0.0007$. Equation 12 holds when we use $\alpha = 4N^2$. For $OPT_{MSMD_k}(x)$, see Equation 10.

$$\frac{1}{\frac{1}{2}(\frac{N^2}{(N+OPT_{\mathrm{MSMD}_k}(x))^2} - \frac{2}{9}) - \frac{1}{4}((\frac{N^2}{N+OPT_{\mathrm{MSMD}_k}(x))^2} - \frac{2}{9})^2} \leq \alpha OPT_{\mathrm{MSMD}_k}(x) \tag{14}$$

Let us replace $OPT_{MSMD_k}(x)$ by P for convenience.

$$2(\frac{N^2}{(N+P)^2} - \frac{2}{9}) - (\frac{N^2}{(N+P)^2} - \frac{2}{9})^2 \geq \frac{4}{\alpha P} \tag{15}$$

In Equation 15, since $\frac{N^2}{(N+P)^2} - \frac{2}{9}$ is positive and less than 1, $\frac{N^2}{(N+P)^2} - \frac{2}{9} > (\frac{N^2}{(N+P)^2} - \frac{2}{9})^2$. Thus, we can rewrite Equation 15 as follows.

$$2(\frac{N^2}{(N+P)^2} - \frac{2}{9}) - (\frac{N^2}{(N+P)^2} - \frac{2}{9})^2 \geq \frac{N^2}{(N+P)^2} - \frac{2}{9} \geq \frac{4}{\alpha P} \tag{16}$$

When we check $\frac{N^2}{(N+P)^2} - \frac{2}{9} \geq \frac{4}{\alpha P}$, we notice that the range of the left term is from $\frac{1}{36}$ to $\frac{7}{9}$ since $P \leq N$. The right term is $\frac{1}{N^2 P}$ when we replace $\alpha = 4N^2$. The right term is always smaller than the left term if $N \geq 4$.

Next, we check whether Equation 13 holds when $\beta = 0.0007$. Note that $Sol'_{\overline{\mathrm{ABC}}}(y') = \frac{1}{\frac{1}{2}(\frac{N^2}{(N+Sol'_{\mathrm{MSMD}_k}(y'))^2} - \frac{2}{9}) - \frac{1}{4}((\frac{N^2}{N+Sol'_{\mathrm{MSMD}_k}(y'))^2} - \frac{2}{9})^2}$.
We next check whether Equation 13 holds.

$$OPT_{\mathrm{MSMD}_k}(x) - Sol'_{\mathrm{MSMD}_k}(g(y')) \leq \beta(\frac{1}{\frac{1}{2}(\frac{N^2}{(N+OPT_{\mathrm{MSMD}_k}(x))^2} - \frac{2}{9}) - \frac{1}{4}((\frac{N^2}{N+OPT_{\mathrm{MSMD}_k}(x))^2} - \frac{2}{9})^2} - \frac{1}{\frac{1}{2}(\frac{N^2}{(N+Sol'_{\mathrm{MSMD}_k}(y'))^2} - \frac{2}{9}) - \frac{1}{4}((\frac{N^2}{N+Sol'_{\mathrm{MSMD}_k}(y'))^2} - \frac{2}{9})^2}) \tag{17}$$

For convenience, let us replace $OPT_{\mathrm{MSMD}_k}(x)$ by A, $Sol'_{\mathrm{MSMD}_k}(g(y'))$ by B, $\frac{1}{2}(\frac{N^2}{(N+OPT_{\mathrm{MSMD}_k}(y'))^2} - \frac{2}{9}) - \frac{1}{4}((\frac{N^2}{N+OPT_{\mathrm{MSMD}_k}(y'))^2} - \frac{2}{9})^2$ by X, and $\frac{1}{2}(\frac{N^2}{(N+Sol_{\mathrm{MSMD}_k}(y'))^2} - \frac{2}{9}) - \frac{1}{4}((\frac{N^2}{N+Sol_{\mathrm{MSMD}_k}(y'))^2} - \frac{2}{9})^2$ by Y. We know that $X > Y$ (See Theorem 3) and $A < B$. We then rewrite Equation 17.

$$A - B \leq \beta(\frac{1}{X} - \frac{1}{Y}) \Rightarrow XY(A - B) \leq \beta(Y - X) \tag{18}$$

Since $A-B$ and $Y-X$ are negative, we multiply -1 on both sides for convenience.

$$XY(B-A) \geq \beta(X-Y) \Rightarrow \frac{XY(B-A)}{(X-Y)} \geq \beta \tag{19}$$

Our goal is to find a positive constant β which satisfies Equation 19. We can set $B-A$ as 1 which is the minimum value.

$$\frac{XY(B-A)}{(X-Y)} \geq \frac{XY}{(X-Y)} \geq \frac{Y^2}{(X-Y)} \geq \beta \tag{20}$$

Here, β can be positive, and we aim to find the minimum value of $\frac{Y^2}{(X-Y)}$ for selecting a proper β value. Let us denote the minimum value of $\frac{Y^2}{(X-Y)}$ by T^{min}. To find T^{min}, we set $OPT_{\mathsf{MSMD}_k}(x) = 4$ and $Sol'_{\mathsf{MSMD}_k}(y') = N$ since this is the minimum and maximum value of A and B. The value T^{min} is as follows.

$$\Rightarrow \frac{(\frac{1}{2}(\frac{1}{4}-\frac{2}{9})-\frac{1}{4}((\frac{1}{4}-\frac{2}{9})^2)^2}{(\frac{1}{2}(\frac{N^2}{(N+4)^2}-\frac{2}{9})-\frac{1}{4}((\frac{N^2}{(N+4)^2}-\frac{2}{9})^2)-(\frac{1}{2}(\frac{1}{4}-\frac{2}{9})-\frac{1}{4}((\frac{1}{4}-\frac{2}{9})^2)}$$

$$\Rightarrow \frac{(\frac{1}{72}-(\frac{1}{4\cdot 36^2}))^2}{(\frac{1}{2}(\frac{N^2}{(N+4)^2}-\frac{2}{9})-\frac{1}{4}(\frac{N^2}{(N+4)^2}-\frac{2}{9})^2-(\frac{1}{72}-\frac{1}{4\cdot 36^2})} \tag{21}$$

We notice that T^{min} decreases monotonically when N increases. When N is 5, T^{min} is 0.006785, and when N is sufficiently large such as 1 quadrillion(10^{15}), $T^{min} \approx 0.000838$. It implies that when N is less than 1 quadrillion, $\beta = 0.0007$ satisfies Equation 20. When N becomes large, we can choose smaller β value. □

We also notice that ABC problem is not in PTAS for $\mu \geq 3$, unless $P = NP$ since PTAS is a sub-class of APX.

Overview of our solutions Due to the NP-hardness of the ABC problem, it is prohibitively expensive to find an exact solution. To solve the problem efficiently, there are two main challenges in finding good co-clusters: (1) *we need to maximize EBMD*; and (2) *we need to preserve the attribute cohesiveness in each co-cluster*. Inspired by this, we design three efficient algorithms that strike a balance of the two aspects differently. Firstly, we propose a bottom-up algorithm that puts maximizing EBMD as the prime concern. We first select an edge that most likely belongs to an attributed co-cluster with a high EBMD. Then we build up co-clusters from this seed edge by iteratively adding nodes to ensure the similarity constraint. Secondly, we propose a group-based matching algorithm that puts preserving the attribute cohesiveness as the main concern. Specifically, in each side of the bipartite network, we find a group of node sets that satisfy the similarity constraint. Then we pair up the node sets from both sides to maximize EBMD. Lastly, we propose a top-down algorithm that simultaneously considers EBMD and attribute cohesiveness. It iteratively deletes edges to split the bipartite network into smaller attribute cohesive co-clusters. In the remaining paper, we elaborate on the three algorithms in turn.

4 BOTTOM-UP ALGORITHM

In this section, we present the bottom-up algorithm (BUA) that focuses on maximizing EBMD as the prime concern. A natural idea works as follows: We first select an edge that most likely belongs to an attributed co-cluster with a high EBMD. Starting from this seed edge, we build up a co-cluster by iteratively adding nodes that ensure the similarity constraint is met. However, this is not trivial. Two questions are raised in designing the algorithm: (1) How to select the seed edge? and (2) Which nodes should be included to ensure the similarity constraint?

To answer the first question, we notice that edge centrality measures the importance of an edge in a graph. Informally, an edge with a large edge centrality implies that it may connect different clusters of nodes [28]. In contrast, an edge having a small edge centrality is likely to be in the center of a cluster since removing the edge may not have an important effect on the graph structure. Inspired by this, we propose to select the edge with the minimum edge centrality as the seed edge. There are many edge centrality measures [5, 7, 9, 20, 57] for graph analysis. However, most of these measures are inefficient and have to be recomputed when the graph structure is changed. We use four edge centrality measures and evaluate their suitability empirically. Two measures are degree-related measures by extending traditional degree centrality [7] for measuring the importance of edges without considering the attributes. We also develop two additional measures to incorporate the attributes based on the Jaccard similarity $\Gamma(.)$. (1) Given a bipartite edge $e = \{u, v\}$, attributed bipartite centrality computes the Jaccard similarity of a set of bipartite neighbor nodes of u in V (resp. v in U) and a set of similar nodes of v in V(resp. u in U). If the average value of the two Jaccard similarities is large, u and v are closely related, i.e., the edge $e = \{u, v\}$ is considered as an unimportant edge. (2) Second order neighbor centrality uses Jaccard similarity of a set of similar nodes of u(resp. (v)) and 2-hop neighbor nodes of u(resp. (v)). Definition 12 defines the four edge centrality measures.

Definition 12. *(edge centrality). Let $e = (u, v) \in E$ be an edge in a bipartite network. We consider the following centrality measures.*

- *degree multiply centrality :* $b(e) = |N(u)| * |N(v)|$.
- *degree sum centrality :* $b(e) = |N(u)| + |N(v)|$.
- *attributed bipartite centrality :* $b(e) = 1 - \frac{\Gamma(N(u),S(v))+\Gamma(N(v),S(u))}{2}$ *where* $\Gamma(.)$ *is a Jaccard similarity between two sets.*
- *second order neighbor centrality :* $b(e) = 1 - \frac{\Gamma(S(u),\bigcup_{x\in N(u)} N(x))+\Gamma(S(v),\bigcup_{x\in N(v)} N(x))}{2}$

Example 5. *In Figure 1, the edge centrality of edges $b(e_{1,1})\{u_1, v_1\}$ and $b(e_{3,4})\{u_3, v_4\}$ are as follows:*

- *attributed bipartite centrality :* $b(e_{1,1}) = 1 - \frac{0.5+0.5}{2} = 0$ *and* $b(e_{3,4}) = 1 - \frac{0+0}{2} = 1$;
- *degree multiply centrality :* $b(e_{1,1}) = 4$ *and* $b(e_{3,4}) = 16$;
- *degree sum centrality :* $b(e_{1,1}) = 4$ *and* $b(e_{3,4}) = 8$;
- *second order neighbor centrality :* $b(e_{1,1}) = 1 - \frac{0.67+0.5}{2} = 0.415$ *and* $b(e_{3,4}) = 1 - \frac{0.2+0.2}{2} = 0.8$

To address the second challenge "which nodes should we include to ensure the similarity constraint?", suppose that a bipartite edge $e_{seed} = \{u, v\}$ has the smallest edge centrality in attributed bipartite graph G. For each node u and v in e_{seed}, we iteratively compute μ-core [63] to satisfy the similarity constraint in both sides :

(1) Initialize a set of nodes T^u(resp. T^v) as the ego network [25] of node u(resp. v), i.e., $T^u \leftarrow N(u) \cup u$ (resp. $T^v \leftarrow N(v) \cup v$).
(2) Compute μ-core D_T^u (resp. D_T^v) of T^u(resp. T^v).
(3) If D_T^u and D_T^v contains node u and v, respectively, return D_T^u and D_T^v as a result; Otherwise, go to Step (4).

(4) For all the nodes $R \subseteq T^u$(resp. T^v) that are in T^u, but not in D_T^u(resp. D_T^v), we add all the neighbors of R to T^u(resp. T^v), i.e., $T^u \leftarrow T^u \cup N(x), \forall x \in R$ (resp. $T^v \leftarrow T^v \cup N(x), \forall x \in R$). Then, go back to the Step (2).

Step (4) is to expand T since from current nodes T we cannot find any subgraph satisfying the similarity constraint and containing node u (resp. v). Thus, we iteratively expand nodes T for considering more nodes to satisfy the similarity constraint. As the result of the above procedures, we have both D_T^u and D_T^v. We combine D_T^u and D_T^v to get co-cluster C and put C in the result. Next, to find other co-clusters, we delete D_T^u and D_T^v from the attributed bipartite network, and find the next seed edge and repeat the above procedure until there is no more co-cluster.

Example 6. *In Figure 1, when we compute attributed bipartite centrality, the edge* (u_4, v_6) *is selected since its centrality score is the smallest (*0.292*). It then identifies co-cluster* $\{u_4, u_5, u_6, u_7, v_4, v_5, v_6\}$. *Next, the edge* (u_3, v_3) *is selected because its centrality score is* 0.41. *It also generates a co-cluster* $\{u_1, u_2, u_3, v_1, v_2, v_3\}$.

Time complexity. Let n denote $|U| + |V|$, $|G_U| = |U| + |E_U|$, M as $|G_U| + |G_V|$. We have

- $O(\frac{min(|U|,|V|)}{\mu+1})$ as the maximum number of co-clusters.
- $O(B)$ to compute edge centrality measure.
- $O(|G_U|)$(or $|G_V|$) to compute μ-core and find a set of connected components in a U(or V) side.
- $O(|U|^2 + |V|^2)$ to compute the similarity of the entities.

Therefore, the time complexity of BUA is $O(|U|^2 + |V|^2 + B + nM\frac{min(|U|,|V|)}{\mu+1})$.

5 GROUP-BASED MATCHING ALGORITHM

In this section, we present the group-based matching algorithm (GMA). Different from BUA, GMA takes ensuring the similarity constraint as the prime concern. The high-level idea of GMA is as follows: We first find a group of node sets from U and V sides of the bipartite network, respectively, such that every node set satisfies the similarity constraint. Then we pair up the node sets from both sides to form a co-cluster to maximize EBMD. This idea raises two challenges to be addressed: (1) How can we find node sets that satisfy the similarity constraint? and (2) Given the node sets, how can we pair them up to form co-clusters with high EBMD?

To address the first challenge, recall that if we build a similarity graph for the nodes in the U (or V) side of the bipartite network, a node set satisfies the similarity constraint if the subgraph induced by the node set is connected and has a minimum degree at least μ. Inspired by this, a naive idea for finding the node sets that satisfy the similarity constraint is to use μ-core. A μ-core is a maximal connected subgraph in which all vertices have degree at least μ. A μ-core can be identified in linear time by removing all nodes that have degree less than μ. However, this idea has a limitation. Due to the maximality constraint of μ-core, it usually finds a large giant node set. For instance, in Gowalla dataset [17], 3-core contains 12 connected components, and one single giant connected component contains 99.96% nodes. Similarly, 3-core of Brightkite [17] contains 16 connected components, and one of them contains 99.6% nodes. It is impossible to pair up node sets to form a co-cluster with high EBMD if there exists only a few candidate node sets.

To overcome such a limitation, we propose to use two techniques: (1) hierarchy of core decomposition and (2) graph partitioning technique. First, recall that to find a μ-core, we have to remove all nodes that have degree less than μ cascadingly. This naturally organize the μ-core, $(\mu+1)$-core, ..., into a hierarchy, i.e., $(\mu+1)$-core $\subseteq$ μ-core when $\mu \geq 1$. Apparently, the set of $(\mu+1)$-cores also satisfy the similarity constraint. Therefore, instead of just considering μ-core as the node sets to be paired up, we consider all μ-core, $(\mu+1)$-core, ..., μ_{max}-core, where μ_{max} is the maximum core value.

Second, to get as many candidate node sets as possible, we adopt the graph partitioning technique to divide the similarity graph into smaller partitions. Then for each partition we compute μ-core, $(\mu+1)$-core, ..., μ_{max}-core. Each core is a candidate node set to be paired up later. With these two techniques, we find a group of node sets that satisfy the similarity constraint from both sides of the bipartite network.

To address the second challenge, i.e., how to pair up the node sets to form a co-cluster with high EBMD, we model this as the maximum weight matching problem [50]. Specifically, we build a bipartite graph $B = (U', V', W')$, where U' (resp. V') is the set of node sets from the U (resp. V) side of the original bipartite network. For a pair of node $u \in U', v \in V'$, the edge weight of (u, v) is the EBMD of the co-cluster formed by pairing u and v up. The maximum weight matching problem [50] is to find a matching in which the sum of the weights is maximized. Then we can use the existing LEDA-algorithm [50] to determine which node sets pair should be paired up to form a co-cluster.

Time complexity. Let $|G_U| = |U| + |E_U|$. Each component takes:

- $O(P)$ to run the graph partitioning algorithm. We utilize pMETIS for our algorithm, and it takes $O(G_U \log k + G_V \log k)$, where k is the number of partitions.
- $O(W)$ to run the maximum weight bipartite matching algorithm. We utilize LEDA's algorithm, and it takes $O((|U| + |V|)(|E| + (|U| + |V|) \log (|U| + |V|)))$
- $O(|G_U| + |G_V|)$ to form shell structures in U and V sides.
- $|E|$ is to setup s-EBMDM.
- $O(|U|^2 + |V|^2)$ to compute the similarity of the entities.

Therefore, its time complexity is $O(|U|^2+|V|^2+G_U \log k+G_V \log k+((|U| + |V|)(|E| + (|U| + |V|) \log (|U| + |V|))) + |G_U| + |G_V| + |E|)$.

6 TOP-DOWN ALGORITHM

We have presented two algorithms that take maximizing EBMD and ensuring similarity constraint as the prime objective, respectively. We proceed to present a top-down algorithm (TDA) that considers both objectives iteratively.

The high-level idea of TDA is as follows: Given a set of initial co-clusters, we iteratively split them into smaller co-clusters until it can no longer be split or we cannot get any EBMD gain when we split the co-clusters. We proceed to present the idea in more detail.

To get the initial set of co-clusters, we follow the idea of GMA. Specifically, we partition the similarity network of U and V for the original bipartite network and compute the μ-core of each partition to ensure the similarity constraint. We invoke LEDA-algorithm [50], which is designed for the maximum weight matching problem, to find the set of co-clusters. To find initial co-cluster, we do not use the result of GMA. GMA uses a set of (μ')-cores where $\mu' \geq \mu$,

and it may return densely connected co-clusters by not considering loosely connected nodes. It indicates that there is no enough space to maximize EBMD by splitting the co-clusters. Thus, we use μ-core to find initial co-clusters.

To split a co-cluster into two smaller co-clusters, we first divide the co-cluster into two subgraphs with graph partition techniques. As graph partition aims to minimize the number of crossing edges, the resulted subgraphs are likely to contain co-clusters with a higher EBMD. Next, we compute μ-core on the similarity graph of each side of the subgraphs to ensure the similarity constraint. Since μ-core can generate multiple connected components, we invoke LEDA-algorithm [50] to find best co-clusters. If newly identified co-clusters C_{new} have larger EBMD compared with the co-cluster C_{old} before being split, we discard C_{old} and add C_{new} to the current solution. Otherwise, we keep C_{old} and check other co-clusters whether we can improve EBMD by splitting them. If there is no more co-clusters to be split, the algorithm is terminated.

Time complexity. Let n be $|V| + |U|$, $|G_U| = |U| + |E_U|$. Time complexity of each component in TDA is as follows.

- $O(t)$ is the maximum number of co-cluster splitting.
- $O(P)$ to run the graph partitioning algorithm. Since we utilize pMETIS for our algorithm, it takes $O(G_U \log k + G_V \log k)$ and $O(G_C \log 2)$, where k is the number of partitions.
- $O(W)$ to run the maximum weight bipartite matching algorithm. We utilize LEDA's algorithm, which takes $O(n(|E| + n \log n))$
- $|G_U|$(or $|G_V|$) to compute μ-core and find a set of connected components in U(or V) side.
- $O(|U|^2 + |V|^2)$ to compute the similarity of the entities.

Therefore, the time complexity of TDA is $O(|U|^2 + |V|^2 + G_U \log k + G_V \log k + t(G_C \log 2)(n(|E| + n \log n))(|G_U| + |G_V|))$.

Summary. We observe that BUA can be utilized when the graph size is relatively small and the clustering coefficient is sufficiently large. In contrast, GMA is preferable when the graph size is large due to its high scalability. Finally, our TDA is capable of handling data of any size and producing high quality results.

Selecting proper parameters. Selecting proper parameters is an open question and is very challenging. For μ, we consider μ as an additional degree of freedom available to users to specify the cohesiveness level. If a user chooses a larger μ, the resultant co-cluster has more attribute cohesiveness and the size will be small. For ϵ, we can use the elbow method by utilizing μ-similar plot [45] to select a proper ϵ value. For $nPart$, we suggest to choose small $nPart$ since selecting a large $nPart$ may return undesired result. For the similarity function, our solution can support any similarity function, which can be selected based on problem domain.

7 EXPERIMENTS

We evaluate our proposed algorithms over real-world attributed bipartite graphs. All the experiments are conducted on CentOS 7.6.1810 with 2.60GHz Xeon CPU E5-4627 v4 and 32GB memory. For implementation, we use JgraphT library [51], Apache Lucene [49], METIS package [38], Coclust package [61], and LEDA package [50].

Dataset. Table 2 shows four categories of datasets that are used in the experimental study:

Table 2: Datasets

Category	Name	$\|U\|$	$\|V\|$	$\|E\|$
LBSN without g.t	YELP [47]	23K	19K	691K
	Brightkite [17]	58K	773K	831K
	Gowalla [17]	197K	1.3M	4M
EBSN without g.t	Meetup [26]	1.2M	3.5M	4.5M
Synthetic network [42]		1M	1M	32M
Doc-word with g.t	20news [43]	36K	19K	2.4M
	pubmed10 [13]	36K	16K	1.8M

- Location-based social network (LBSN) including YELP, Brightkite, and Gowalla. They have no ground-truth co-cluster information. For LBSN, it consists of a set of users and a set of locations. A bipartite edge represents that a user makes a check-in at a location. Each user has a set of friends, and each location has a coordinate. We define two users are similar if they are friends. Two locations are similar if their distance is within a threshold ϵ.
- Event-based social network (EBSN) including Meetup. The Meetup dataset consists of a set of users and a set of events. We consider two users to be similar if they attend at least three common events. For each event, we consider the top-k events with the highest textual similarity to be similar events. It has no ground-truth information.
- Synthetic network [42] consists of $1M$ nodes in each side and $32M$ bipartite edges. We utilize LFR benchmark dataset [42] to generate a synthetic unipartite network $G = (V_G, E_G)$. We then generate a symmetric bipartite network from the unipartite graph. We firstly set $U = G$ and $V = G$, and connect a node $u \in U$ to a node $v \in U$ if $u = v$ (named self-edge). Next, if $(u, v) \in E_G$, we make two edges from $u \in U$ to $v \in V$ and $v \in U$ to $u \in V$. We next define a noise parameter $\tau = 0.01$, and pick $\tau|E|$ edges and randomly rewire them to make a noise of a bipartite network.
- Doc-word dataset contains 20news and Pubmed10 dataset. It consists of a set of words and documents. A bipartite edge represents a word that appears in a document. We remove stopwords [48], words with frequency smaller than 5, and find the top-10 similar words using Word2Vec [52]. For each document, we find the top 10 similar documents with the highest textual similarity [49].

Algorithms. To the best of our knowledge, there is no direct competitor in the literature for solving the ABC problem. Therefore, we compare our algorithms with several representative co-clustering algorithms including the recent work using deep learning, but they do not consider the attributes and the returned co-clusters may not satisfy the similarity constraint (We tried to do a post-processing of applying k-core or bipartite weight matching, but it does not work as it often returns empty results): CCMOD [1], SpecMOD [41], Info [22], and DeepCC [72]. For CCMOD, SpecMOD, and Info, we use coclust package [61] with default parameters. For DeepCC [2], we use the default parameters. For BiMLPA, we use $\theta = 0.3$ and $\lambda = 5$ where θ indicates the weight ratio threshold, and λ indicates the maximum number of labels.

- BUA : Bottom-up algorithm.
- GMA : Group-based matching algorithm.

[2] https://github.com/DerronXu/Deep-Co-Clustering

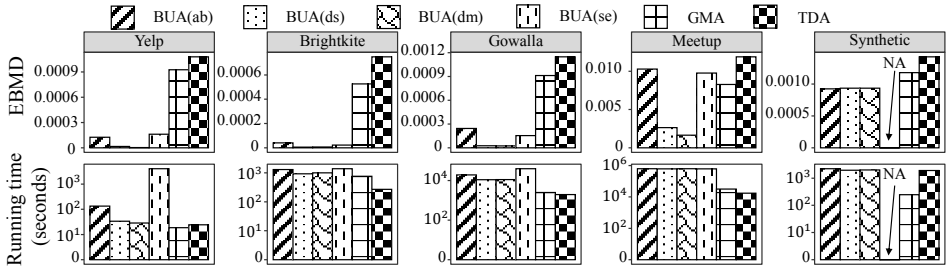


Figure 6: EBMD and running time

- GMA* : Group-based matching algorithm without considering similarity constraint.
- TDA : Top-down algorithm.
- CCMOD : Co-clustering by direct modularity maximization [1].
- SpecMOD : Co-clustering by spectral approximation [41].
- Info : Information-theoretic Co-clustering [22].
- DeepCC : Deep learning-based co-clustering [72].
- BiMLPA : Multi-label Propagation in Bipartite Networks [69]

BUA algorithm can incorporate different centrality measures, and we denote BUA with attributed bipartite centrality as BUA(ab), BUA with degree sum as BUA(ds), BUA with degree multiply as BUA(dm), and BUA with attributed bipartite centrality as BUA(se).

Evaluation metrics. When there is no ground-truth information, we report EBMD to measure the quality of the attributed co-clusters. Otherwise, we report Normalized Mutual Information(NMI) [19] and Rand Index [59]. the NMI measure estimates the quality of the clustering with respect to a given underlying class labeling by measuring how closely the clustering algorithm could reconstruct the underlying label distribution in the data [67], and the Rand index is a measure of the similarity between two data clusterings, that is computed by the number of concordant pairs divided by the total number of pairs[59].

7.1 Experimental result

Real-world network without ground-truth information. We first compare our proposed algorithms and baseline algorithms. We set $\mu = 5$ $\epsilon = 0.2$ for LBSN, and $\mu = 3$ and top-6 similar events in EBSN. In this experiment, we report the results of our algorithms since other algorithms cannot guarantee the similarity constraints are satisfied. The first row of Figure 6 reports EBMD, which indicates the quality of the results. We observe that TDA achieves the largest EBMD for all the datasets, and GMA follows on most of datasets. Among BUA algorithms with different betweenness centrality measures, BUA(ab) attributed bipartite centrality and BUA(se) second-order centrality have larger EBMD. The second row of Figure 6 shows the running time. We observe that different variants of BUA are much slower than GMA and TDA, which have comparable efficiency. Note that BUA(se) does not finish within 24 hours on synthetic datasets. Thus, we do not report BUA(se) on the synthetic network dataset. This experiment shows that TDA achieves the best quality in terms of EBMD and is more efficient than other algorithms. Among BUA algorithms with different betweenness centrality measures, BUA(ab) and BUA(se) have better performance since they consider the attribute cohesiveness in each side. Since BUA(se) is less efficient, we report BUA(ab) in the rest experiments.

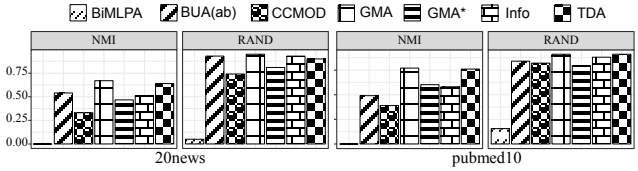


Figure 7: NMI and RI result

Real-world networks with ground-truth information. We next evaluate the effectiveness on 20news and pubmed10 that have ground-truth co-clusters. Figure 7 shows NMI [19] and Rand index [59] of the algorithms. Note that the results of SpecMOD and DeepCC are not included since they cannot finish within 24 hours. CCMOD and Info require the number of co-clusters as an input, and we give the number of ground-truth co-clusters. Since our problem is not a graph partitioning problem, some nodes (empirically less than 20%) are not included in the results. To check the accuracy, we filter out the nodes that are not included in our solution. We observe that our proposed algorithms return the best accuracy in terms of NMI and RI. This is because similarity constraint and EBMD help to identify high-quality co-clusters. We notice that BiMLPA has very low accuracy since it returns a very large co-cluster containing 99.9% nodes as a result.

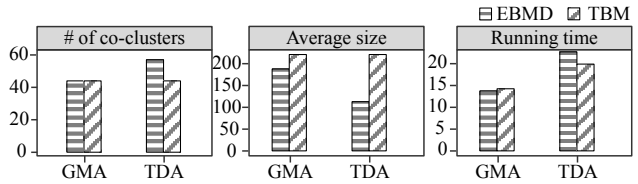


Figure 8: Comparing EBMD and existing bipartite modularity

Comparing EBMD with traditional bipartite modularity. This experiment is to evaluate the performance when we optimize EBMD by comparing with the traditional bipartite modularity [6] named TBM as the objective function in our proposed algorithms. We use two algorithms: GMA and TDA since (1) they perform better than BUA and (2) BUA cannot be extended to use the existing modularity (since it uses a heuristic to maximize EBMD of the result co-clusters). We report the number of attributed co-clusters and the average size of the attributed co-clusters. Figure 8 shows the comparison results when we use EBMD (EBMD-based) and the existing bipartite modularity (TBM-based) on Yelp dataset. The results on other datasets are qualitatively similar. GMA returns the

same number of co-clusters (44 co-clusters) for both modularity measures since the number of co-clusters depends on the attributes. However, the average size of co-clusters of TBM is 18% larger than the EBMD. For TDA, TBM returns few co-clusters compared with EBMD and the average size of co-clusters of TBM is 96% larger than that of EBMD. The results clearly show that EBMD is able to alleviate the resolution limit problem and TBM returns few and larger co-clusters due to the resolution limit problem. For the scalability test, in TDA, optimizing TBM is relatively faster than optimizing EBMD since TBM prefers large-sized co-clusters as a result.

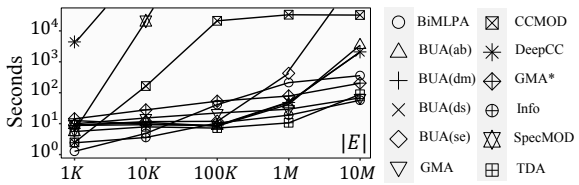


Figure 9: Scalability with number of bipartite edges

Scalability test. We next evaluate the scalability of the algorithms. We fix the U and V sides and randomly vary the number of check-in edges from $1K$ to $10M$ on Yelp. Figure 9 shows that the running time increases linearly with the number of edges. We set $nPart = 10$ for GMA and TDA. When the number of edges is 10^5, it did not finish within 24 hours since it is required to compute common neighbors many times. For baseline algorithms, we use default parameters and set the number of co-cluster $nClus$ as 10. This is because by setting $nClus = nPart$, it will generate the similar number of co-clusters and it is a fair comparison. If we set larger $nClus$ or $nPart$, the baseline algorithms cannot finish in a reasonable time. We observe that the running times of GMA and TDA linearly increase with the size of the graph. SpecMOD and DeepCC cannot finish within 24 hours when there are more than 10^5 edges. CCMOD is 370 order of magnitude slower than TDA when $|E| = 10^7$.

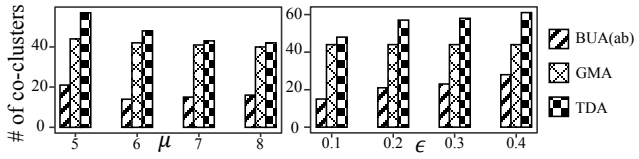
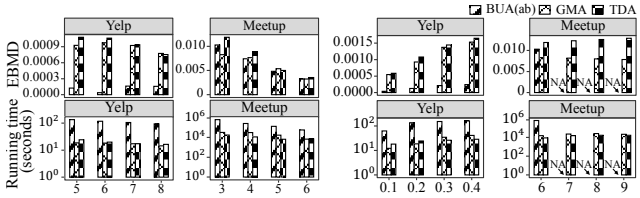


Figure 10: Number of co-clusters (Yelp)

Number of connected components. Analyzing the number of returned co-clusters helps to understand the characteristics of an algorithm. Hence, we report the number of co-clusters of our algorithms by varying parameters μ and ϵ in Yelp dataset in Figure 10. Note that BUA(ab) returns fewer co-clusters since many real-world graph has a small-diameter [60]. Let us recall BUA algorithm. It expands a seed node of a seed edge to satisfy the similarity constraint by iteratively adding a set of neighbor nodes in U (or V) side. When U or V side has a small diameter, many nodes can belong to a single co-cluster, and thus it returns fewer co-clusters. TDA and GMA return more co-clusters comparing with BUA(ab) since employing a graph partitioning method can help to find more small-sized co-clusters. In GMA and TDA, to avoid finding a large giant co-cluster, it utilizes the graph partitioning technique. Therefore, we can control the number of co-clusters by varying $nPart$.

When $nPart$ is fixed, TDA returns more co-clusters compared with GMA since TDA iteratively splits co-clusters to find a solution. We observe that when μ increases, the number of co-clusters decreases since a larger μ makes some nodes not to be considered due to the similarity constraint. When ϵ increases (ϵ is a distance threshold in Yelp and k in top-k events in Meetup), the number of co-clusters increases since a larger ϵ allows additional edges in each side.



(a) Varying μ **(b) Varying ϵ**

Figure 11: Varying parameters

Varying μ and ϵ. We next observe the results when we vary user parameters to understand their effect. Figure 11a presents the result when we change μ value in Yelp and Meetup dataset. The results on the other datasets are qualitatively similar and are ignored due to space limitation. We observe that when μ increases, EBMD decreases since a larger internal density in U and V sides does not directly relate to the cohesive bipartite edges. We observe that when μ increases, the running time decreases since we do not need to consider some nodes that do not satisfy the similarity constraint.

Figure 11b presents the result when changing the threshold ϵ value in Yelp and Meetup. We observe that when ϵ increases (ϵ is a distance threshold in Yelp and k in top-k events in Meetup), EBMD increases and the running time decrease. The reason is that a larger ϵ allows additional edges in each side. These additional edges will cause more nodes to be included in co-clusters, resulting in higher EBMD. As expected, when ϵ increases, the running time increases due to the increased number of edges in the similarity graph. Note that in Meetup dataset, BUA(ab) does not finish within 24 hours, and thus its result is not included.

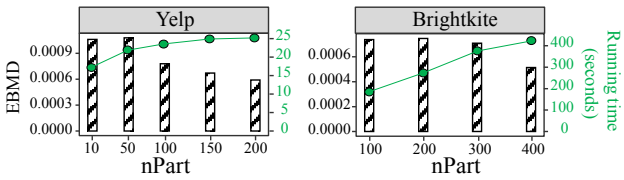
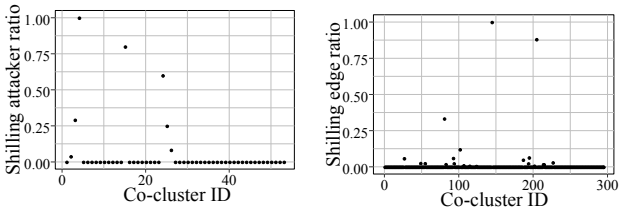


Figure 12: Varying $nPart$ on TDA (Yelp, Brightkite)

Varying $nPart$. In GMA and TDA, we utilize the graph partitioning technique METIS. This experiment is to evaluate the effect of $nPart$ in METIS. Hence, we vary $nPart$ and report the EBMD and the running time in Yelp and Brightkite datasets. Figure 12 shows EBMD of TDA when we change $nPart$. The results on the other datasets display a similar trend and are ignored. We observe that when $nPart$ increases, the running time of the algorithms increases. This is because we need to consider many small-sized partitions. We also observe that when $nPart$ becomes larger, EBMD decreases since it cannot find large-sized co-clusters, and thus it may not result in large EBMD. Based on this result, we choose $nPart = 50$ for Yelp, $nPart = 200$ for Brightkite. Similarly, we set $nPart = 1,000$ for Gowalla, and $nPart = 5,000$ for Meetup dataset.



(a) MovieLens **(b) E-commerce**

Figure 13: Case study

7.2 Case study

We utilize our algorithm for finding shilling attackers in MovieLens dataset [31] and detecting fraud edges from Alibaba e-commerce network. We use TDA algorithm with $\mu = 3$ with $nPart = 10$.

Finding Shilling Attackers. For Movielens dataset, we adopt simulated bot attacker [16] to inject shilling attackers. Totally, 94 shilling attackers are injected. For U and V sides, we consider that any users (or movies) are connected if they watch (or are watched by) at least 5 same movies (or users). Figure 13a shows the ratio of the shilling attackers among all the resulted co-clusters. Among 53 co-clusters, shilling attackers are located in 7 co-clusters. Especially, in the co-cluster #4, all the identified users are the shilling attackers.

Finding Fraud Edges. Alibaba e-commerce dataset consists of $1.5M$ users, $0.5M$ items and $2M$ purchasing links from users to the items. It has ground-truth fraud edges. For both U and V sides, we keep 5 nearest neighbors based on the euclidean distance between their attribute vectors. Figure 13b shows the ratio of the fraud edges among all resulted co-clusters. Among 295 co-clusters, only a few co-clusters contain many fraud edges. We notice that the co-cluster #148 contains all the fraud edges in Figure 13b.

From both case studies, we notice that the shilling attackers and fraud edges are located concentratedly, and thus we show that our algorithm has the ability to identify them.

8 RELATED WORK

Co-clustering problem [21, 22] is to perform simultaneous clustering of the rows and columns of a matrix and has been extensively studied on various types of data. Many co-clustering algorithms have been developed [21, 22, 24, 71, 72]. These approaches only focus on finding co-clusters without considering attributes of nodes. However, our ABC considers the attributes of the nodes and similarity constraints for finding co-clusters. As discussed in Introduction, it is difficult to adapt these algorithms for solving ABC. The most close work to ours is the modularity-based co-clustering. Barber [6] proposes a bipartite modularity, and a spectral partitioning algorithm using the Eigenvectors of the Laplacian matrix. This algorithm aims to minimize the number of crossing edges between clusters. Our EBMD is an extended version of Barber's bipartite modularity. Aliem [1] propose iterative alternating optimization procedure to maximize the bipartite modularity [6] for a predefined number of clusters. Guimera et al. [30] propose a special bipartite modularity to find one-side clusters. Murata [56] proposes a bipartite modularity by allowing one-to-many bipartite clusters, meaning that the one side of a cluster such as c_V can belong to different co-clusters c'. In contrast, in ABC we propose the EBMD to address the resolution limit problem of the bipartite modularity. Additionally, our ABC problem needs to meet the similarity constraints among entities in each side, which is not considered in these previous studies. Clearly, consideration of node attributes in the ABC problem introduces new challenges.

The attributed network clustering problem [8, 70] is related to the ABC problem. Given an attributed network, where each node is associated with a set of attributes, the attributed network clustering problem aims to find a set of clusters of which each is structurally cohesive while sharing similar attributes. However, they cannot be used to solve the problem of finding attributed bipartite co-clusters.

Our work is related to the heterogeneous information network (HIN) clustering problem [55, 66]. Recently, several HIN clustering approaches are proposed. Zhe et al. [75] propose to apply modularity optimization in attributed networks to find attributed communities. Huang et al. [33] propose an algorithm by joint nonnegative matrix factorization and graph optimization for attributed community discovery. Sun et al. [68] develop the Network Embedding for attributed clustering. Zhang et al. [73] propose a heterogeneous graph neural network model. Most of these approaches do not consider the strict cohesiveness of the identified clusters, that is, these approaches cannot be utilized to solve the ABC problem since they cannot satisfy the attribute cohesiveness. Recently, Jian et al. [36] study the relational community detection problem (RCD) in HINs by considering the strict cohesiveness. RCD problem is to find a set of maximal subgraphs by satisfying a set of constraints. However, the RCD problem cannot be utilized to find a solution to ABC since (1) it aims to find a set of maximal subgraphs; and (2) although it could consider the attributed cohesiveness in both U and V sides, respectively, it cannot consider internal dense connections and external sparse connections simultaneously by using the cohesiveness constraint, i.e., it cannot model EBMD by utilizing the specified constraints.

9 CONCLUSION

In this paper, we formulated the Attributed Bipartite Co-clustering problem (ABC) which aims to find a set of co-clusters which are densely connected and the nodes of each side have similar attributes. We showed that classic bipartite modularity optimization suffers from the bipartite k-clique resolution limit problem. Therefore, we proposed a new bipartite modularity measure which does not suffer from the bipartite k-clique resolution limit problem. We also proved that our problem is NP-hard and not in APX. To solve the problem, we proposed three effective and efficient heuristic solutions. Finally, extensive experiments on synthetic and real-world attributed bipartite networks were conducted to demonstrate the effectiveness and efficiency of our proposed algorithms.

ACKNOWLEDGMENTS

This research is supported, in part, by Alibaba Group through Alibaba Innovative Research (AIR) Program and Alibaba-NTU Singapore Joint Research Institute (JRI) (No. AN-GC-2019-008), MOE Tier-2 grant MOE2019-T2-2-181, and MOE Tier-1 grant RG114/19 (S).

REFERENCES

[1] Melissa Ailem, François Role, and Mohamed Nadif. 2015. Co-clustering document-term matrices by direct maximization of graph modularity. In *CIKM*. ACM, 1807–1810.

[2] Melissa Ailem, François Role, and Mohamed Nadif. 2016. Graph modularity maximization as an effective method for co-clustering text data. *Knowledge-Based Systems* 109 (2016), 160–173.

[3] Omid Amini, David Peleg, Stéphane Pérennes, Ignasi Sau, and Saket Saurabh. 2009. Degree-Constrained Subgraph Problems: Hardness and Approximation Results. In *Approximation and Online Algorithms: 6th International Workshop, WAOA 2008, Karlsruhe, Germany, September 18-19, 2008. Revised Papers*. 29–42.

[4] Omid Amini, David Peleg, Stéphane Pérennes, Ignasi Sau, and Saket Saurabh. 2012. On the approximability of some degree-constrained subgraph problems. *Discrete Applied Mathematics* 160, 12 (2012), 1661–1679.

[5] Francesca Arrigo and Michele Benzi. 2016. Updating and downdating techniques for optimizing network communicability. *SISC* 38, 1 (2016), B25–B49.

[6] Michael J Barber. 2007. Modularity and community detection in bipartite networks. *PRE* 76, 6 (2007), 066102.

[7] Phillip Bonacich. 1987. Power and centrality: A family of measures. *AJS* 92, 5 (1987), 1170–1182.

[8] Cécile Bothorel, Juan David Cruz, Matteo Magnani, and Barbora Micenkova. 2015. Clustering attributed graphs: models, measures and methods. *Network Science* 3, 3 (2015), 408–444.

[9] Ulrik Brandes. 2001. A faster algorithm for betweenness centrality. *Journal of mathematical sociology* 25, 2 (2001), 163–177.

[10] Zhan Bu, Jie Cao, Hui-Jia Li, Guangliang Gao, and Haicheng Tao. 2018. GLEAM: A graph clustering framework based on potential game optimization for large-scale social networks. *Knowledge and Information Systems* 55, 3 (2018), 741–770.

[11] Mingming Chen, Konstantin Kuzmin, and Boleslaw K Szymanski. 2014. Community detection via maximization of modularity and its variants. *IEEE Transactions on Computational Social Systems* 1, 1 (2014), 46–65.

[12] Tianlong Chen, Pramesh Singh, and Kevin E Bassler. 2018. Network community detection using modularity density measures. *JSTAT* 2018, 5 (2018), 053406.

[13] Yanhua Chen, Lijun Wang, Ming Dong, and Jing Hua. 2009. Exemplar-based visualization of large document corpus (infovis2009-1115). *IEEE Transactions on Visualization and Computer Graphics* 15, 6 (2009), 1161–1168.

[14] Yizong Cheng and George M. Church. 2000. Biclustering of Expression Data. In *ISMB*. 93–103.

[15] Zhiyuan Cheng, James Caverlee, Himanshu Barthwal, and Vandana Bachani. 2014. Who is the barbecue king of texas? A geo-spatial approach to finding local experts on twitter. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval* (Gold Coast, Queensland, Australia). 335–344.

[16] Paul-Alexandru Chirita, Wolfgang Nejdl, and Cristian Zamfir. 2005. Preventing shilling attacks in online recommender systems. In *WIDM*. 67–74.

[17] Eunjoon Cho, Seth A Myers, and Jure Leskovec. 2011. Friendship and mobility: user movement in location-based social networks. In *KDD*. 1082–1090.

[18] Pierluigi Crescenzi. 1997. A short guide to approximation preserving reductions. In *Proceedings of Computational Complexity. Twelfth Annual IEEE Conference*. IEEE, 262–273.

[19] Leon Danon, Albert Diaz-Guilera, Jordi Duch, and Alex Arenas. 2005. Comparing community structure identification. *JSTAT* 2005, 09 (2005), P09008.

[20] Pasquale De Meo, Emilio Ferrara, Giacomo Fiumara, and Angela Ricciardello. 2012. A novel measure of edge centrality in social networks. *Knowledge-based systems* 30 (2012), 136–150.

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

[22] Inderjit S Dhillon, Subramanyam Mallela, and Dharmendra S Modha. 2003. Information-theoretic co-clustering. In *KDD*. ACM, 89–98.

[23] Chris Ding, Tao Li, Wei Peng, and Haesun Park. 2006. Orthogonal nonnegative matrix t-factorizations for clustering. In *KDD*. 126–135.

[24] Liang Du and Yi-Dong Shen. 2013. Towards robust co-clustering. In *IJCAI*. 1317–1322.

[25] Alessandro Epasto, Silvio Lattanzi, and Renato Paes Leme. 2017. Ego-splitting framework: From non-overlapping to overlapping clusters. In *KDD*. 145–154.

[26] Kaiyu Feng, Gao Cong, Sourav S Bhowmick, and Shuai Ma. 2014. In search of influential event organizers in online social networks. In *SIGMOD*. 63–74.

[27] Santo Fortunato and Marc Barthelemy. 2007. Resolution limit in community detection. *PNAS* 104, 1 (2007), 36–41.

[28] Michelle Girvan and Mark EJ Newman. 2002. Community structure in social and biological networks. *PNAS* 99, 12 (2002), 7821–7826.

[29] Quanquan Gu and Jie Zhou. 2009. Co-clustering on manifolds. In *KDD*. ACM, 359–368.

[30] Roger Guimerà, Marta Sales-Pardo, and Luís A Nunes Amaral. 2007. Module identification in bipartite and directed networks. *PRE* 76, 3 (2007), 036102.

[31] F Maxwell Harper and Joseph A Konstan. 2015. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)* 5, 4 (2015), 1–19.

[32] Petter Holme, Fredrik Liljeros, Christofer R Edling, and Beom Jun Kim. 2003. Network bipartivity. *PRE* 68, 5 (2003), 056107.

[33] Zhihao Huang, Xiaoxiong Zhong, Qiang Wang, Maoguo Gong, and Xiaoke Ma. 2020. Detecting community in attributed networks by dynamically exploring node attributes and topological structure. *Knowledge-Based Systems* 196 (2020), 105760.

[34] Syed Fawad Hussain and Muhammad Haris. 2019. A k-means based co-clustering (kCC) algorithm for sparse, high dimensional data. *Expert Systems with Applications* 118 (2019), 20–34.

[35] Hawoong Jeong, Bálint Tombor, Réka Albert, Zoltan N Oltvai, and A-L Barabási. 2000. The large-scale organization of metabolic networks. *Nature* 407, 6804 (2000), 651.

[36] Xun Jian, Yue Wang, and Lei Chen. 2020. Effective and efficient relational community detection and search in large dynamic heterogeneous information networks. *Proceedings of the VLDB Endowment* 13, 10 (2020), 1723–1736.

[37] Andrew EG Jonas. 2012. Region and place: Regionalism in question. *Progress in Human Geography* 36, 2 (2012), 263–272.

[38] George Karypis and Vipin Kumar. 1998. A fast and high quality multilevel scheme for partitioning irregular graphs. *SISC* 20, 1 (1998), 359–392.

[39] Junghoon Kim, Tao Guo, Kaiyu Feng, Gao Cong, Arijit Khan, and Farhana M Choudhury. 2020. Densely Connected User Community and Location Cluster Search in Location-Based Social Networks. In *SIGMOD*. 2199–2209.

[40] Jungeun Kim, Jae-Gil Lee, Byung Suk Lee, and Jiajun Liu. 2020. Geosocial Co-Clustering: A Novel Framework for Geosocial Community Detection. *ACM Transactions on Intelligent Systems and Technology (TIST)* 11, 4 (2020), 1–26.

[41] Lazhar Labiod and Mohamed Nadif. 2011. Co-clustering for binary and categorical data with maximum modularity. In *ICDM*. IEEE, 1140–1145.

[42] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. 2008. Benchmark graphs for testing community detection algorithms. *Physical review E* 78, 4 (2008), 046110.

[43] Ken Lang. 1995. Newsweeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning*. 331–339.

[44] Fredrik Liljeros, Christofer R Edling, Luis A Nunes Amaral, H Eugene Stanley, and Yvonne Åberg. 2001. The web of human sexual contacts. *Nature* 411, 6840 (2001), 907.

[45] Sungsu Lim, Junghoon Kim, and Jae-Gil Lee. 2016. BlackHole: Robust community detection inspired by graph drawing. In *ICDE*. IEEE, 25–36.

[46] Renjie Lin, Shiping Wang, and Wenzhong Guo. 2019. An Overview of Co-Clustering via Matrix Factorization. *IEEE Access* 7 (2019), 33481–33493.

[47] Yiding Liu, Tuan-Anh Nguyen Pham, Gao Cong, and Quan Yuan. 2017. An experimental evaluation of point-of-interest recommendation in location-based social networks. *Proceedings of the VLDB Endowment* 10, 10 (2017), 1010–1021.

[48] Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*. 55–60.

[49] Michael McCandless, Erik Hatcher, Otis Gospodnetić, and O Gospodnetić. 2010. *Lucene in action*. Vol. 2. Manning, Greenwich, USA.

[50] Kurt Mehlhorn and Stefan Näher. 1999. *LEDA: a platform for combinatorial and geometric computing*. Cambridge university press, Cambridge.

[51] Dimitrios Michail, Joris Kinable, Barak Naveh, and John V Sichi. 2020. JGraphT—A Java Library for Graph Data Structures and Algorithms. *TOMS* 46, 2 (2020), 1–29.

[52] Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.).

[53] Mehdi Rezapoor Mirsaleh and Mohammad Reza Meybodi. 2016. A Michigan memetic algorithm for solving the community detection problem in complex network. *Neurocomputing* 214 (2016), 535–545.

[54] Atsushi Miyauchi and Noriyoshi Sukegawa. 2015. Maximizing Barber's bipartite modularity is also hard. *Optimization Letters* 9, 5 (2015), 897–913. https://doi.org/10.1007/s11590-014-0818-7

[55] Vincenzo Moscato and Giancarlo Sperli. 2021. A survey about community detection over On-line Social and Heterogeneous Information Networks. *Knowledge-Based Systems* 224 (2021), 107112.

[56] Tsuyoshi Murata. 2009. Community division of heterogeneous networks. In *International Conference on Complex Sciences*. Springer, 1011–1022.

[57] M Newman. 2010. *Networks: an introduction: OUP Oxford*. Oxford University Press, Oxford, England.

[58] Feng Pan, Xiang Zhang, and Wei Wang. 2008. CRD: fast co-clustering on large datasets utilizing sampling-based matrix decomposition. In *SIGMOD*. 173–184.

[59] William M Rand. 1971. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association* 66, 336 (1971), 846–850.

[60] Neil Robertson and Paul D Seymour. 1986. Graph minors. V. Excluding a planar graph. *Journal of Combinatorial Theory, Series B* 41, 1 (1986), 92–114.

[61] François Role, Stanislas Morbieu, and Mohamed Nadif. 2019. CoClust: A Python Package for Co-Clustering. *Journal of Statistical Software* 88, 7 (2019), 1–29. https://doi.org/10.18637/jss.v088.i07

[62] Aghiles Salah, Melissa Ailem, and Mohamed Nadif. 2018. Word co-occurrence regularized non-negative matrix tri-factorization for text data co-clustering. In *AAAI*. 3992–3999.

[63] Stephen B Seidman. 1983. Network structure and minimum degree. *Social networks* 5, 3 (1983), 269–287.

[64] Sunanda Vivek Shanbhag. 2016. *A faster version of Louvain method for community detection for efficient modeling and analytics of cyber systems.* Ph.D. Dissertation. Purdue University.

[65] Fanhua Shang, LC Jiao, and Fei Wang. 2012. Graph dual regularization non-negative matrix factorization for co-clustering. *Pattern Recognition* 45, 6 (2012), 2237–2250.

[66] Chuan Shi, Yitong Li, Jiawei Zhang, Yizhou Sun, and S Yu Philip. 2016. A survey of heterogeneous information network analysis. *IEEE Transactions on Knowledge and Data Engineering* 29, 1 (2016), 17–37.

[67] Alexander Strehl, Joydeep Ghosh, and Raymond Mooney. 2000. Impact of similarity measures on web-page clustering. In *Workshop on artificial intelligence for web search (AAAI 2000)*, Vol. 58. 64.

[68] Heli Sun, Fang He, Jianbin Huang, Yizhou Sun, Yang Li, Chenyu Wang, Liang He, Zhongbin Sun, and Xiaolin Jia. 2020. Network embedding for community detection in attributed networks. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 14, 3 (2020), 1–25.

[69] Hibiki Taguchi, Tsuyoshi Murata, and Xin Liu. 2020. Bimlpa: community detection in bipartite networks by multi-label propagation. In *International Conference on Network Science*. Springer, 17–31.

[70] Chun Wang, Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, and Chengqi Zhang. 2019. Attributed Graph Clustering: A Deep Attentional Embedding Approach. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence* (Macao, China) *(IJCAI'19)*. 3670–3676.

[71] Hua Wang, Feiping Nie, Heng Huang, and Chris Ding. 2011. Nonnegative matrix tri-factorization based high-order co-clustering and its fast implementation. In *ICDM*. IEEE, 774–783.

[72] Dongkuan Xu, Wei Cheng, Bo Zong, Jingchao Ni, Dongjin Song, Wenchao Yu, Yuncong Chen, Haifeng Chen, and Xiang Zhang. 2019. Deep Co-Clustering. In *SDM*. SIAM, 414–422.

[73] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V Chawla. 2019. Heterogeneous graph neural network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 793–803.

[74] Pei-Pei Zhang, Kan Chen, Yue He, Tao Zhou, Bei-Bei Su, Yingdi Jin, Hui Chang, Yue-Ping Zhou, Li-Cheng Sun, Bing-Hong Wang, et al. 2006. Model and empirical study on some collaboration networks. *Physica A* 360, 2 (2006), 599–616.

[75] Chen Zhe, Aixin Sun, and Xiaokui Xiao. 2019. Community Detection on Large Complex Attribute Network. In *KDD*. ACM, 2041–2049.