Broad Learning based Social Community Detection

Jiawei Zhang and Philip S. Yu

Abstract In the real-world online social networks, users tend to form different social communities. Due to its extensive applications, community detection in online social networks has been a hot research topic in recent years. In this chapter, we will focus on introducing the social community detection problem in online social networks. To be more specific, we will take the hard community detection problem as an example to introduce the existing models proposed for conventional (one single) homogeneous social network, and the recent broad learning based (multiple aligned) heterogeneous social networks respectively.

Key Word: Community Detection; Social Media; Aligned Heterogeneous Networks; Broad Learning

1 Overview

"Birds of a feather flock together." In the real-world online social networks, users also tend to form different social groups [2]. Users belonging to the same groups usually have more frequent interactions with each other, while those in different groups will have less interactions on the other hand [62]. Formally, such social groups form by users in online social networks are called the online social communities [57]. Online social communities will partition the network into a number of connected components, where the intra-community social connections are usually far more dense compared with the inter-community social connections [57]. Meanwhile, from the mathematical representation perspective, due to these online

Jiawei Zhang
Department of Computer Science, Florida State University, FL, USA. e-mail: jzhang@cs.fsu.edu
Philip S. Yu
Department of Computer Science, University of Illinois at Chicago, IL, USA. e-mail: psyu@cs.uic.edu

social communities, the social network adjacency matrix tend to be not only sparse but also low-rank [51].

Identifying the social communities formed by users in online social networks is formally defined as the *community detection* problem [57, 55, 16]. Community detection is a very important problem for online social network studies, as it can be crucial prerequisite for numerous concrete social network services: (1) better organization of users' friends in online social networks (e.g., Facebook and Twitter), which can be achieved by applying community detection techniques to partition users' friends into different categories, e.g., schoolmates, family, celebrities, etc. [10]; (2) better recommender systems for users with common shopping preference in e-commerce social sites (e.g., Amazon and Epinions), which can be addressed by grouping users with similar purchase records into the same clusters prior to recommender system building [36]; and (3) better identification of influential users [44] for advertising campaigns in online social networks, which can be attained by selecting the most influential users in each community as the seed users in the viral marketing [35].

In this chapter, we will focus on introducing the *social community detection* problem in online social networks. Given a heterogeneous network G with node set $\mathscr{V}$, we can represent the involved user nodes in network G as set $\mathscr{U} \subset \mathscr{V}$. Based on both the social structures among users as well as the diverse attribute information from the network G, the *social community detection* problem aims at partitioning the user set $\mathscr{U}$ into several subsets $\mathscr{C} = \{\mathscr{U}_1, \mathscr{U}_2, \cdots, \mathscr{U}_k\}$, where each subset $\mathscr{U}_i, i \in \{1,2,\cdots,k\}$ is called a social community. Term k formally denotes the total number of partitioned communities, which is usually provided as a hyper-parameter in the problem.

Depending on whether the users are allowed to be partitioned into multiple communities simultaneously or not, the *social community detection* problem can actually be categorized into two different types:

- *Hard Social Community Detection*: In the *hard social community detection* problem, each user will be partitioned into one single community, and all the social communities are disjoint without any overlap. In other words, given the communities $\mathscr{C} = \{\mathscr{U}_1, \mathscr{U}_2, \cdots, \mathscr{U}_k\}$ detected from network G, we have $\mathscr{U} = \bigcup_i \mathscr{U}_i$ and $\mathscr{U}_i \cap \mathscr{U}_j = \emptyset, \forall i, j \in \{1,2,\cdots,k\} \wedge i \neq j$.
- *Soft Social Community Detection*: In the *soft social community detection* problem, users can belong to multiple social communities simultaneously. For instance, if we apply the *Mixture-of-Gaussian Soft Clustering* algorithm as the base community detection model, each user can belong to multiple communities with certain probabilities. In the *soft social community detection* result, the communities are no longer disjoint and will share some common users with other communities.

Meanwhile, depending on the network connection structures, the *community detection* problem can be categorized as *directed network community detection* [28] and *undirected network community detection* [62]. Based on the heterogeneity of the network information, the *community detection* problem can be divided into the

homogeneous network community detection [47] and *heterogeneous network community detection* [37, 42, 52, 58]. Furthermore, according to the number of networks involved, the *community detection* problem involves *single network community detection* [22] and *multiple network community detection* [57, 55, 16, 52, 58]. In this chapter, we will take the *hard community detection problem* as an example to introduce the existing models proposed for conventional (one single) *homogeneous social network*, and especially the recent broad learning based (multiple aligned) *heterogeneous social networks* [20, 53, 54, 60] respectively.

This chapter is organized as follows. At the beginning, in Section 2, we will introduce the community detection problem and the existing methods proposed for traditional one single homogeneous networks. After that, we will talk about the latest research works on social community detection across multiple aligned heterogeneous networks. The cold start community detection [55] is introduced in Section 3, in which we propose a new information transfer algorithm to propagate information from other developed source networks to the emerging target network. In Section 4, we will be focused on the concurrent mutual community detection [57] across multiple aligned heterogeneous networks simultaneously, where information from other aligned networks will be applied to refine their community detection results mutually. Finally, in Section 5, we talk about the synergistic community detection across multiple large-scale networks based on the distributed computing platform [16].

2 Traditional Homogeneous Network Community Detection

Social community detection problem has been studied for a long time, and many community detection models have been proposed based on different types of techniques. In this section, we will talk about the social community detection problem for one single homogeneous network G, whose objective is to partition the user set $\mathscr{U}$ in network G into k disjoint subsets $\mathscr{C} = \{\mathscr{U}_1, \mathscr{U}_2, \cdots, \mathscr{U}_k\}$, where $\mathscr{U} = \bigcup_i \mathscr{U}_i$ and $\mathscr{U}_i \cap \mathscr{U}_j = \emptyset, \forall i, j \in \{1, 2, \cdots, k\}$. Several different community detection methods will be introduced, which include *node proximity based community detection*, *modularity maximization based community detection*, and *spectral clustering based community detection*.

2.1 Node Proximity based Community Detection

The *node proximity based community detection* method assumes that "close nodes tend to be in the same communities, while the nodes far away from each other will belong to different communities". Therefore, the *node proximity based community detection* model partition the nodes into different clusters based on the node proximity measures [24]. Various node proximity measures can be used here, including

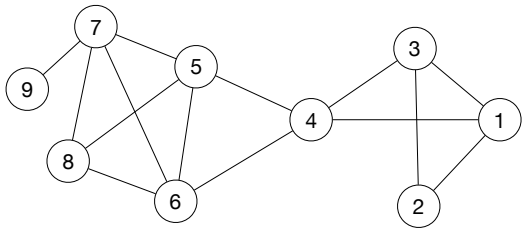


Fig. 1 Example of Homogeneous Network (Nodes 1 and 3 are Structural Equivalent).

the node *structural equivalence* to be introduced as follows, as well as various node closeness measures as introduced in [61].

In a homogeneous network G, the proximity of nodes, like u and v, can be calculated based on their positions and connections in the network structure.

Example 1. For instance, in Figure 1, we show an example of a homogeneous network G involving 9 nodes and 14 links among them. For the nodes 1 and 3, they have equivalent positions in the network structure. According to the connections around 1 and 3, we can observe the neighbors of node 1 are $\Gamma(1) = \{2,3,4\}$, while the neighbors of node 3 include $\Gamma(3) = \{1,2,4\}$. They share two common neighbors $\{1,2\}$, and also connect with each other. If we switch their positions, the network structure will still be the same as the original one and the neighbors of nodes 1 and 3 will still remain the same.

Definition 1. (Structural Equivalence): Given a network $G = (\mathscr{V}, \mathscr{E})$, two nodes $u, v \in \mathscr{V}$ are said to be *structural equivalent* iff

1. Nodes u and v are not connected and u and v share the same set of neighbors (i.e., $(u,v) \notin \mathscr{E} \wedge \Gamma(u) = \Gamma(v)$),

2. Or u and v are connected and excluding themselves, u and v share the same set of neighbors (i.e., $(u,v) \in \mathscr{E} \wedge \Gamma(u) \setminus \{v\} = \Gamma(v) \setminus \{u\}$).

As mentioned before, for the nodes which are *structural equivalent*, they are *substitutable* and switching their positions will not change the overall network structure. The *structural equivalence* concept can be applied to partition the nodes into different communities. For the nodes which are *structural equivalent*, they can be grouped into the same communities, while for the nodes which are not equivalent in their positions, they will be partitioned into different groups. However, the *structural equivalence* can be too restricted for practical use in detecting the communities in real-world social networks. Computing the *structural equivalence* relationships among all the node pairs in the network can lead to very high time cost. What's more, the *structural equivalence* relationship will partition the social network structure into lots of small-sized fragments, since the users will have different social patterns in making friends online and few user will have identical neighbors actually.

To avoid the weakness mentioned above, some other measures are proposed to measure the proximity among nodes in the networks. For instance, as introduced in [61], the node closeness measures based on the social connections can all be applied here to compute the node proximity, e.g., "common neighbor", "Jaccard's coefficient". Here, if we use "common neighbor" as the proximity measure, by applying the "common neighbor" measure to the network G, we can transform the network G into a set of instances $\mathscr{V}$ with mutual closeness scores $\{c(u,v)\}_{u,v\in\mathscr{V}}$. Some existing similarity/distance based clustering algorithms, like k-Medoids (a variant of k-Means), can be applied to partition the users into different communities.

2.2 Modularity Maximization based Community Detection

Besides the pairwise proximity of nodes in the network, the connection strength of a community is also very important in the community detection process. Different measures have been proposed to compute the strength of a community, like the *modularity* measure [29] to be introduced in this part.

The *modularity* measure takes account of the node degree distribution. For instance, given the network G, the expected number of links existing between nodes u and v with degrees $D(u)$ and $D(v)$ can be represented as $\frac{D(u)\cdot D(v)}{2|\mathscr{E}|}$. Meanwhile, in the network, the real number of links existing between u and v can be denoted as entry $A[u,v]$ in the social adjacency matrix $\mathbf{A}$. For the user pair (u,v) with a low expected connection confidence score, if they are connected in the real world, it indicates that u and v have a relatively strong relationship with each other. Meanwhile, if the community detection algorithm can partition such user pairs into the same group, it will be able to identify very strong social communities from the network.

Based on such an intuition, the strength of a community, e.g., $\mathscr{U}_i \in \mathscr{C}$ can be defined as

$$\sum_{u,v\in\mathscr{U}_i}\left(A[u,v]-\frac{D(u)\cdot D(v)}{2|\mathscr{E}|}\right). \tag{1}$$

Example 2. For instance, let's take network shown in Figure 1 as an example. We assume the network nodes are partitioned into two groups, i.e., $\mathscr{C}=\{\mathscr{U}_1,\mathscr{U}_2\}$, where $\mathscr{U}_1=\{1,2,3,4\}$ and $\mathscr{U}_2=\{5,6,7,8,9\}$. According to the network structure, we can compute the expected number of links between user pairs within community $\mathscr{U}_1$ in Table 1.

(u,v)	(1, 1)	(1, 2)	(1,3)	(1,4)	(2, 1)	(2, 2)	(2, 3)	(2, 4)	(3, 1)	(3, 2)	(3, 3)	(3, 4)	(4, 4)	⋯
D(u), D(v)	3, 3	3, 2	3, 3	3, 4	2, 3	2, 2	2, 3	2, 4	3, 3	3, 2	3, 3	3, 4	4, 3	⋯
A[u,v]	0	1	1	1	1	0	1	0	1	1	0	1	1	⋯
$\frac{D(u)\cdot D(v)}{2\lvert\mathscr{E}\rvert}$	$\frac{9}{28}$	$\frac{6}{28}$	$\frac{9}{28}$	$\frac{12}{28}$	$\frac{9}{28}$	$\frac{4}{28}$	$\frac{6}{28}$	$\frac{8}{28}$	$\frac{9}{28}$	$\frac{6}{28}$	$\frac{9}{28}$	$\frac{12}{28}$	$\frac{12}{28}$	⋯
$A[u,v]-\frac{D(u)\cdot D(v)}{2\lvert\mathscr{E}\rvert}$	$\frac{-9}{28}$	$\frac{22}{28}$	$\frac{19}{28}$	$\frac{16}{28}$	$\frac{19}{28}$	$\frac{-4}{28}$	$\frac{22}{28}$	$\frac{-8}{28}$	$\frac{19}{28}$	$\frac{22}{28}$	$\frac{-9}{28}$	$\frac{16}{28}$	$\frac{16}{28}$	⋯

Table 1 Numerical Analysis of Community $\mathscr{U}_1=\{1,2,3,4\}$.

According to the above equation, we can compute the strength of community $\mathscr{U}_1 = \{1,2,3,4\}$ as

$$\sum_{u,v\in\mathscr{U}_1}\left(A[u,v]-\frac{D(u)\cdot D(v)}{2|\mathscr{E}|}\right)=4.857. \tag{2}$$

Furthermore, the strength of the overall community detection result $\mathscr{C} = \{\mathscr{U}_1, \mathscr{U}_2, \cdots, \mathscr{U}_k\}$ can be defined as the *modularity* of the communities as follows.

Definition 2. (Modularity): Given the community detection result $\mathscr{C} = \{\mathscr{U}_1, \mathscr{U}_2, \cdots, \mathscr{U}_k\}$, the modularity of the community structure is defined as

$$Q(\mathscr{C})=\frac{1}{2|\mathscr{E}|}\sum_{\mathscr{U}_i\in\mathscr{C}}\sum_{u,v\in\mathscr{U}_i}\left(A[u,v]-\frac{D(u)\cdot D(v)}{2|\mathscr{E}|}\right). \tag{3}$$

The *modularity* concept effectively measures the strength of the detected community structure. Generally, for a community structure with a larger *modularity* score, it indicates a good community detection result.

Example 3. By following the analysis provided in Example 2, we can also compute the strength of community $\mathscr{U}_2$ as

$$\sum_{u,v\in\mathscr{U}_2}\left(A[u,v]-\frac{D(u)\cdot D(v)}{2|\mathscr{E}|}\right)=4.857. \tag{4}$$

Therefore, we can compute the *modularity* of community detection results $\mathscr{C} = \{\mathscr{U}_1, \mathscr{U}_2\}$ as

$$Q(\mathscr{C})=\frac{1}{2|\mathscr{E}|}\sum_{\mathscr{U}_i\in\mathscr{C}}\sum_{u,v\in\mathscr{U}_i}\left(A[u,v]-\frac{D(u)\cdot D(v)}{2|\mathscr{E}|}\right)=0.347. \tag{5}$$

Another way to explain the *modularity* is from the number of links within and across communities. By rewriting the above *modularity* equation, we can have

$$\begin{aligned}
Q(\mathscr{C}) &= \frac{1}{2|\mathscr{E}|}\sum_{\mathscr{U}_i\in\mathscr{C}}\sum_{u,v\in\mathscr{U}_i}\left(A[u,v]-\frac{D(u)\cdot D(v)}{2|\mathscr{E}|}\right) &(6)\\
&= \frac{1}{2|\mathscr{E}|}\left(\sum_{\mathscr{U}_i\in\mathscr{C}}\sum_{u,v\in\mathscr{U}_i}A[u,v]-\sum_{\mathscr{U}_i\in\mathscr{C}}\sum_{u,v\in\mathscr{U}_i}\frac{D(u)\cdot D(v)}{2|\mathscr{E}|}\right) &(7)\\
&= \frac{1}{2|\mathscr{E}|}\left(\sum_{\mathscr{U}_i\in\mathscr{C}}\sum_{u,v\in\mathscr{U}_i}A[u,v]-\frac{1}{2|\mathscr{E}|}\sum_{\mathscr{U}_i\in\mathscr{C}}\sum_{u\in\mathscr{U}_i}D(u)\cdot\sum_{u\in\mathscr{U}_i}D(v)\right) &(8)\\
&= \frac{1}{2|\mathscr{E}|}\left(\sum_{\mathscr{U}_i\in\mathscr{C}}\sum_{u,v\in\mathscr{U}_i}A[u,v]-\frac{1}{2|\mathscr{E}|}\sum_{\mathscr{U}_i\in\mathscr{C}}(\sum_{u\in\mathscr{U}_i}D(u))^2\right). &(9)
\end{aligned}$$

In the above equation, term $\sum_{u,v\in\mathscr{U}_i} A[u,v]$ denotes the number of links connecting users within the community $\mathscr{U}_i$ (which will be 2 times the intra-community links for undirected networks, as each link will be counted twice). Term $\sum_{u\in\mathscr{U}_i} D(u)$ denotes the sum of node degrees in community $\mathscr{U}_i$, which equals to the number of intra-community and inter-community links connected to nodes in community $\mathscr{U}_i$. If there exist lots of inter-community links, then the *modularity* measure will have a smaller value. On the other hand, if the inter-community links are very rare, the *modularity* measure will have a larger value. Therefore, maximizing the community *modularity* measure is equivalent to minimizing the inter-community link numbers.

The *modularity* measure can also be represented with linear algebra equations. Let matrix $\mathbf{A}$ denote the adjacency matrix of the network, and vector $\mathbf{d} \in \mathbb{R}^{|\mathscr{V}|\times 1}$ denote the degrees of nodes in the network. we can define the *modularity matrix* as

$$\mathbf{B} = \mathbf{A} - \frac{\mathbf{d}\mathbf{d}^\top}{2|\mathscr{E}|}. \tag{10}$$

Let matrix $\mathbf{H} \in \{0,1\}^{|\mathscr{V}|\times k}$ denotes the communities that users in $\mathscr{V}$ belong to. In real application, such a binary constraint can be relaxed to allow real value solutions for matrix $\mathbf{H}$. The optimal community detection result can be obtained by solving the following objective function

$$\max \frac{1}{2|\mathscr{E}|}\mathrm{Tr}(\mathbf{H}^\top \mathbf{B}\mathbf{H}) \tag{11}$$

$$s.t. \mathbf{H}^\top \mathbf{H} = \mathbf{I}, \tag{12}$$

where constraint $\mathbf{H}^\top\mathbf{H} = \mathbf{I}$ ensures there are not overlap in the community detection result.

The above objective function looks very similar to the objective function of *spectral clustering* to be introduced in the next section. After obtaining the optimal $\mathbf{H}$, the communities can be obtained by applying the K-Means algorithm to $\mathbf{H}$ to determine the cluster labels of each node in the network.

2.3 Spectral Clustering based Community Detection

In the community detection process, besides maximizing the proximity of nodes belonging to the same communities (as introduced in Section 2.1), minimizing the connections among nodes in different clusters is also an important factor. Different from the previous proximity based community detection algorithms, another way to address the community detection problem is from the cost perspective. Partition the nodes into different clusters will cut the links among the clusters. To ensure the nodes partitioned into different clusters have less connections with each other, the number of links to be cut in the community detection process should be as small as possible [45].

2.3.1 Cut

Formally, given the community structure $\mathscr{C} = \{\mathscr{U}_1, \mathscr{U}_2, \cdots, \mathscr{U}_k\}$ detected from network G. The number of links cut between communities $\mathscr{U}_i, \mathscr{U}_j \in \mathscr{C}$ can be represented as

$$cut(\mathscr{U}_i, \mathscr{U}_j) = \sum_{u \in \mathscr{U}_i} \sum_{v \in \mathscr{U}_j} I(u,v), \tag{13}$$

where function $I(u,v) = 1$ if $(u,v) \in \mathscr{E}$; otherwise, it will be 0.

The total number of links cut in the partition process can be represented as

$$cut(\mathscr{C}) = \sum_{\mathscr{U}_i \in \mathscr{C}} cut(\mathscr{U}_i, \bar{\mathscr{U}_i}), \tag{14}$$

where set $\bar{\mathscr{U}_i} = \mathscr{C} \setminus \mathscr{U}_i$ denotes the remaining communities except $\mathscr{U}_i$.

By minimizing the cut cost introduced in the partition process, we can obtain the optimal community detection result with the minimum number of cross-community links.

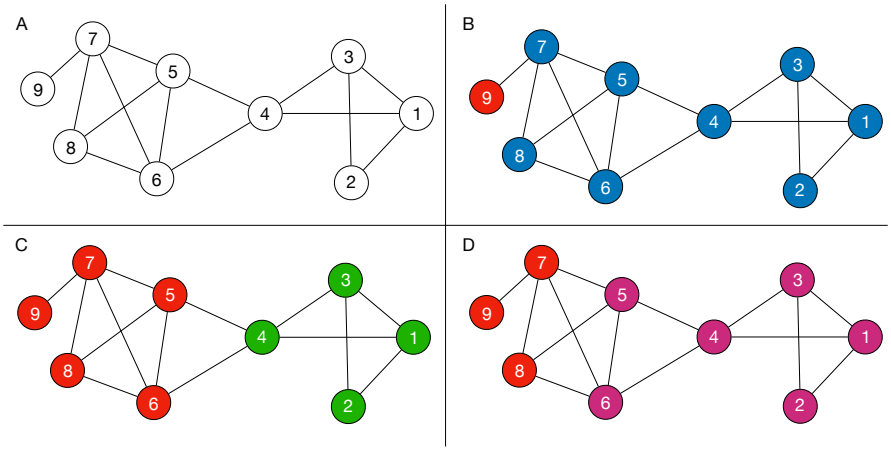


Fig. 2 Comparison of Cut, Ratio-Cut and Normalized-Cut Measures in Social Network Community Detection.

Example 4. For instance, in Figure 2, we show 3 different community detection results (i.e., plots B-D) of the input network as illustrated in plot A. For the 9 nodes in the network, plot B partition the node into 2 communities: $\mathscr{C} = \{\mathscr{U}_1, \mathscr{U}_2\}$, where $\mathscr{U}_1 = \{9\}$ and $\mathscr{U}_2 = \{1,2,3,4,5,6,7,8\}$. Link $(7,9)$ is the only link between different communities in the network. According to the above definition, the introduced cut can be represented as

$$cut(\mathscr{C}) = cut(\mathscr{U}_1, \bar{\mathscr{U}_1}) + cut(\mathscr{U}_2, \bar{\mathscr{U}_2}) = 2, \tag{15}$$

where $\bar{\mathscr{U}_1} = \mathscr{U}_2$, $\bar{\mathscr{U}_2} = \mathscr{U}_1$ and $cut(\mathscr{U}_1, \mathscr{U}_2) = |\{(7,9)\}| = 1$.

Meanwhile, for the community detection results in plot C, its introduced cut will be 2×2, since two edges $\{(4,5),(4,6)\}$ are between the two communities, i.e., $\mathscr{U}_1 = \{5,6,7,8,9\}$ and $\mathscr{U}_2 = \{1,2,3,4\}$. Community detection results in plot D introduces a cut of 8, and 4 edges $\{(5,7),(5,8),(6,7),(6,8)\}$ connects those two detected communities.

Considering that we don't allow empty communities, and plot B actually identifies the optimal community structure of the input network data, where the cut cost is minimized. However, we can also observe that the achieved community structure is also extremely imbalanced, where community $\mathscr{U}_1 = \{9\}$ contains a singleton node, while $\mathscr{U}_2 = \{1,2,3,4,5,6,7,8\}$ contains 8 nodes. Such a problem will be much more severe when it comes to the real-world social network data. In the following part of this section, we will introduce two other cost measures that can help achieve more balanced community detection results.

2.3.2 Ratio-Cut and Normalized-Cut

As shown in the example, the minimum cut cost treat all the links in the network equally, and can usually achieve very imbalanced partition results (e.g., a singleton node as a cluster) when applied in the real-world community detection problem. To overcome such a disadvantage, some models have been proposed to take the community size into consideration. The community size can be calculated by counting the number of nodes or links in each community, which will lead to two new cost measures: *ratio-cut* and *normalized-cut*.

Formally, given the community detection result $\mathscr{C} = \{\mathscr{U}_1, \mathscr{U}_2, \cdots, \mathscr{U}_k\}$ in network G, the *ratio-cut* and *normalized-cut* costs introduced in the community detection result can be defined as follows respectively.

$$ratio-cut(\mathscr{C}) = \frac{1}{k} \sum_{\mathscr{U}_i \in \mathscr{C}} \frac{cut(\mathscr{U}_i, \bar{\mathscr{U}_i})}{|\mathscr{U}_i|}, \tag{16}$$

where $|\mathscr{U}_i|$ denotes the number of nodes in community $\mathscr{U}_i$.

$$ncut(\mathscr{C}) = \frac{1}{k} \sum_{\mathscr{U}_i \in \mathscr{C}} \frac{cut(\mathscr{U}_i, \bar{\mathscr{U}_i})}{vol(\mathscr{U}_i)}, \tag{17}$$

where $vol(\mathscr{U}_i)$ denotes the degree sum of nodes in community $\mathscr{U}_i$.

Example 5. For instance, by following the example as illustrated in Example 4 and Figure 2. We have already computed the cut cost introduced by the community detection results in plots B, C, D, which are 2, 4 and 8 respectively. Here, if we also consider the node number of node degree volume of each community, we can get the *ratio-cut* and *ncut* of these community detection results as follows:

- Plot B: $\mathscr{U}_1 = \{9\}$ and $\mathscr{U}_2 = \{1,2,3,4,5,6,7,8\}$. We have $cut(\mathscr{U}_1, \bar{\mathscr{U}_1}) = cut(\mathscr{U}_2, \bar{\mathscr{U}_2}) = 1$. The sizes and volumes of these communities are $|\mathscr{U}_1| = 1$, $|\mathscr{U}_2| = 8$, and

$vol(\mathscr{U}_1) = 1$, $vol(\mathscr{U}_2) = 27$. Therefore, we have

$$ratio\text{-}cut(\mathscr{C}) = \frac{1}{2}\left(\frac{cut(\mathscr{U}_1, \bar{\mathscr{U}_1})}{|\mathscr{U}_1|} + \frac{cut(\mathscr{U}_2, \bar{\mathscr{U}_2})}{|\mathscr{U}_2|}\right) = \frac{1}{2}\left(\frac{1}{1} + \frac{1}{8}\right) = \frac{9}{16}, \quad (18)$$

$$ncut(\mathscr{C}) = \frac{1}{2}\left(\frac{cut(\mathscr{U}_1, \bar{\mathscr{U}_1})}{vol(\mathscr{U}_1)} + \frac{cut(\mathscr{U}_2, \bar{\mathscr{U}_2})}{vol(\mathscr{U}_1)}\right) = \frac{1}{2}\left(\frac{1}{1} + \frac{1}{27}\right) = \frac{14}{27}. \quad (19)$$

- Plot C: $\mathscr{U}_1 = \{5,6,7,8,9\}$ and $\mathscr{U}_2 = \{1,2,3,4\}$. We have $cut(\mathscr{U}_1, \bar{\mathscr{U}_1}) = cut(\mathscr{U}_2, \bar{\mathscr{U}_2}) = 2$. The sizes and volumes of these communities are $|\mathscr{U}_1| = 5$, $|\mathscr{U}_2| = 4$, and $vol(\mathscr{U}_1) = 16$, $vol(\mathscr{U}_2) = 12$. Therefore, we have

$$ratio\text{-}cut(\mathscr{C}) = \frac{1}{2}\left(\frac{cut(\mathscr{U}_1, \bar{\mathscr{U}_1})}{|\mathscr{U}_1|} + \frac{cut(\mathscr{U}_2, \bar{\mathscr{U}_2})}{|\mathscr{U}_2|}\right) = \frac{1}{2}\left(\frac{1}{5} + \frac{1}{4}\right) = \frac{9}{40}, \quad (20)$$

$$ncut(\mathscr{C}) = \frac{1}{2}\left(\frac{cut(\mathscr{U}_1, \bar{\mathscr{U}_1})}{vol(\mathscr{U}_1)} + \frac{cut(\mathscr{U}_2, \bar{\mathscr{U}_2})}{vol(\mathscr{U}_1)}\right) = \frac{1}{2}\left(\frac{1}{16} + \frac{1}{12}\right) = \frac{7}{96}. \quad (21)$$

- Plot D: $\mathscr{U}_1 = \{7,8,9\}$ and $\mathscr{U}_2 = \{1,2,3,4,5,6\}$. We have $cut(\mathscr{U}_1, \bar{\mathscr{U}_1}) = cut(\mathscr{U}_2, \bar{\mathscr{U}_2}) = 4$. The sizes and volumes of these communities are $|\mathscr{U}_1| = 3$, $|\mathscr{U}_2| = 6$, and $vol(\mathscr{U}_1) = 20$, $vol(\mathscr{U}_2) = 8$. Therefore, we have

$$ratio\text{-}cut(\mathscr{C}) = \frac{1}{2}\left(\frac{cut(\mathscr{U}_1, \bar{\mathscr{U}_1})}{|\mathscr{U}_1|} + \frac{cut(\mathscr{U}_2, \bar{\mathscr{U}_2})}{|\mathscr{U}_2|}\right) = \frac{1}{2}\left(\frac{1}{3} + \frac{1}{6}\right) = \frac{1}{4}, \quad (22)$$

$$ncut(\mathscr{C}) = \frac{1}{2}\left(\frac{cut(\mathscr{U}_1, \bar{\mathscr{U}_1})}{vol(\mathscr{U}_1)} + \frac{cut(\mathscr{U}_2, \bar{\mathscr{U}_2})}{vol(\mathscr{U}_1)}\right) = \frac{1}{2}\left(\frac{1}{20} + \frac{1}{8}\right) = \frac{7}{80}. \quad (23)$$

As shown in the above example, from the computed costs, we find that the community detected in plot C achieves much lower ratio-cut and ncut costs compared with those in plots B and D. Compared against the regular *cut* cost, both *ratio-cut* and *normalized-cut* prefer a balanced partition of the social network.

2.3.3 Spectral Clustering

Actually the objective function of both *ratio-cut* and *normalized-cut* can be unified as the following linear algebra equation

$$\min_{\mathbf{H} \in \{0,1\}^{|\mathscr{V}| \times k}} \mathrm{Tr}(\mathbf{H}^\top \bar{\mathbf{L}} \mathbf{H}), \quad (24)$$

where matrix $\mathbf{H} \in \{0,1\}^{|\mathscr{V}| \times k}$ denotes the communities that users in $\mathscr{V}$ belong to.

Let $\mathbf{A} \in \{0,1\}^{|\mathscr{V}| \times |\mathscr{V}|}$ denote the social adjacency matrix of the network, and we can represent the corresponding diagonal matrix of $\mathbf{A}$ as matrix $\mathbf{D}$, where $\mathbf{D}$ has value $D(i,i) = \sum_j A(i,j)$ on its diagonal. The Laplacian matrix of the network adjacency matrix $\mathbf{A}$ can be represented as $\mathbf{L} = \mathbf{D} - \mathbf{A}$. Depending on the specific measures applied, matrix $\bar{\mathbf{L}}$ can be represented as

$$\bar{\mathbf{L}} = \begin{cases} \mathbf{L}, & \text{for ratio-cut measure,} \\ \mathbf{D}^{\frac{-1}{2}}\mathbf{L}\mathbf{D}^{\frac{-1}{2}}, & \text{for normalized-cut measure.} \end{cases} \tag{25}$$

The binary constraint on the variable $\mathbf{H}$ renders the problem a non-linear integer programming problem, which is very hard to solve. One common practice to learn the variable $\mathbf{H}$ is to apply spectral relaxation to replace the binary constraint with the orthogonality constraint.

$$\min \mathrm{Tr}(\mathbf{H}^\top \bar{\mathbf{L}}\mathbf{H}), \tag{26}$$

$$s.t. \mathbf{H}^\top \mathbf{H} = \mathbf{I}. \tag{27}$$

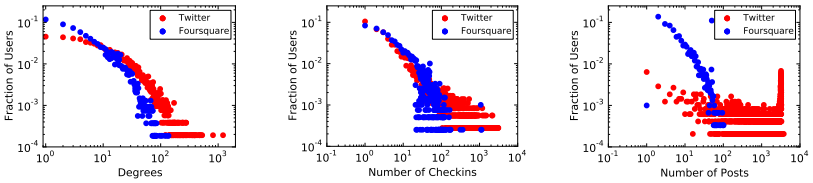
As proposed in [38], the optimal solution $\mathbf{H}^*$ to the above objective function equals to the eigen-vectors corresponding to the k smallest eigen-values of matrix $\bar{\mathbf{L}}$.

3 Emerging Network Community Detection

The community detection algorithms introduced in the previous section are mostly proposed for one single homogeneous network. However, in the real world, most of the online social networks are actually heterogeneous containing very complex information. In recent years, lots of new online social networks have emerged and start to provide services, the information available for the users in these emerging networks is usually very limited. Meanwhile, many of the users are also involved in multiple online social networks simultaneously. For users who are using these emerging networks, they may also be involved in other developed social networks for a long time [56, 50]. The abundant information available in these mature networks can actually be useful for the community detection in the emerging networks. In this section, we will introduce the cross-network community detection for emerging networks with information transferred from other mature social networks [55].

3.1 Background Knowledge

Witnessing the incredible success of popular online social networks, e.g., Facebook and Twitter, a large number of new social networks offering specific services also spring up overnight to compete for the market share. Generally, emerging networks are the networks containing very sparse information and can be (1) the social networks which are newly constructed and start to provide social services for a very short period of time; or (2) even more mature ones that start to branch into new geographic areas or social groups [54]. These emerging networks can be of a wide variety, which include (1) location-based social networks, e.g., Foursquare and Jiepang;



(a) degree distribution (b) checkin number distribution (c) post number distribution

Fig. 3 Information and anchor user distributions in Foursquare and Twitter. (a): social degree distribution, (b): number of check-ins distribution, (c): number of posts distribution.

(2) photo organizing and sharing sites, e.g., Pinterest and Instagram; (3) educational social sites, e.g., Stage 32.

Community detection in emerging networks is a new problem and conventional community detection methods for well-developed networks cannot be applied directly. Compared with well-developed networks, information in emerging networks can be too sparse to support traditional community detection methods to calculate effective closeness scores and achieve good results. According to the market report from DRM[1], by the end of 2013, the total number of registered users in Foursquare has reached 45 million but these Foursquare users have only post 40 million tips. In other words, each user has posted less than one tip in Foursquare on average. Meanwhile, the 1 billion registered Twitter users have published more than 300 billion tweets by the end of 2013 and each Twitter user has written more than 300 tweets. We also provide a statistics investigation on the a crawled dataset, which include both Foursquare and Twitter, and the information distribution results are given in Figure 3. As shown in Figures 3(a)-3(c), users in Twitter have far more social connections, posts and location check-ins than users in Foursquare. The shortage of information encountered in community detection problems for emerging networks can be a serious obstacle for traditional community detection methods to achieve good performance and is urgent to be solved.

In this section, we will introduce the social community detection for *emerging networks* with information propagated across multiple *partially aligned social networks*, which is formally defined as the "*emerging network community detection*" problem. Especially, when the network is brand new, the problem will be the "*cold start community detection*" problem. *Cold start problem* is mostly prevalent in *recommender systems* [53], where the system cannot draw any inferences for users or items, for which it has not yet gathered sufficient information, but few works have been done on studying the *cold start problem* in clustering/community detection problems. The "*emerging network community detection*" problem and "*cold start community detection*" problem studied in this section are both novel problems and very different from other existing works on community detection.

[1] http://expandedramblings.com

3.2 Problem Formulation

Networks studied in this section can be formulated as two partially aligned attribute augmented heterogeneous networks: $\mathscr{G} = ((G^t, G^s), (A^{t,s}, A^{s,t}))$, where G^t and G^s are the emerging target network and well-developed source network respectively and $A^{t,s}, A^{s,t}$ are the sets of anchor links between G^t and G^s. Both G^t and G^s can be formulated as the attribute augmented heterogeneous social network, e.g., $G^t = (\mathscr{V}^t, \mathscr{E}^t, \mathscr{A}^t)$ (where sets $\mathscr{V}^t$, $\mathscr{E}^t$ and $\mathscr{A}^t$ denote the user nodes, social links and diverse attributes in the network). With information propagated across $\mathscr{G}$, we can calculate the *intimacy matrix*, $\mathbf{H}$, among users in $\mathscr{V}^t$. *emerging network community detection* problem aims at partitioning user set $\mathscr{V}^t$ of the emerging network G^t into K disjoint clusters, $\mathscr{C} = \{C_1, C_2, \cdots, C_K\}$, based on the *intimacy matrix*, $\mathbf{H}$, where $\bigcup_i^K C_i = \mathscr{V}^t$ and $C_i \cap C_j = \emptyset, \forall i, j \in \{1, 2, \cdots, K\}, i \neq j$. When the target network G^t is brand new, i.e., $\mathscr{E}^t = \emptyset$ and $\mathscr{A}^t = \emptyset$, the problem will be the *cold start community detection* problem. The "*emerging network community detection*" studied in this section is also very challenging to solve due to the following reasons:

- *network heterogeneity problem*: Proper definition of closeness measure among users with link and attribute information in the heterogeneous social networks is very important for community detection problems.
- *shortage of information*: Community detection for emerging networks can suffer from the shortage of information problem, i.e., the "*cold start problem*" [53, 54].
- *network difference problem*: Different networks can have different properties. Some information propagated from other well-developed networks can be useful for solving the *emerging network community detection* problem but some can be misleading on the other hand.
- *high memory space cost*: Community detection across multiple aligned networks can involve too many nodes and connections, which will lead to high space cost.

To solve all the above challenges, a novel community detection method, CAD, will be introduced in great detail in this section: (1) CAD introduces a new concept, *intimacy*, to measure the closeness relationships among users with both link and attribute information in online social networks; (2) CAD can propagate useful information from aligned well-developed networks to the emerging network to solve the shortage of information problem; (3) CAD addresses the network heterogeneity and difference problems with both *micro-level* and *macro-level* control of the link and attribute information proportions, whose parameters can be adjusted by CAD automatically; (4) effective and efficient cross-network information propagation models are introduced in this section to solve the high space cost problem.

3.3 Intimacy Matrix of Homogeneous Network

The CAD model is built based on the closeness scores among users, which is formally called the *intimacy scores* in this section. Here, we will introduce the *intimacy*

scores and *intimacy matrix* used in CAD from a information propagation perspective.

For a given homogeneous network, e.g., $G = (\mathscr{V}, \mathscr{E})$, where $\mathscr{V}$ is the set of users and $\mathscr{E}$ is the set of social links among users in $\mathscr{V}$, we can define the adjacency matrix of G to be $\mathbf{A} \in \mathbb{R}^{|\mathscr{V}|\times|\mathscr{V}|}$, where $A(i,j) = 1$, iff $(u_i, u_j) \in \mathscr{E}$. Meanwhile, via the social links in $\mathscr{E}$, information can propagate among the users within the network, whose propagation paths can reflect the closeness among users [33]. Formally, term

$$p_{ji} = \frac{A(j,i)}{\sqrt{\sum_m A(j,m)\sum_n A(n,i)}} \tag{28}$$

is called the information *transition probability* from u_j to u_i, which equals to the proportion of information propagated from u_j to u_i in one step.

We can use an example to illustrate how information propagates within the network more clearly. Let's assume that user $u_i \in \mathscr{V}$ injects a stimulation into network G initially and the information will be propagated to other users in G via the social interactions afterwards. During the propagation process, users receive stimulation from their neighbors and the amount is proportional to the difference of the amount of information reaching the user and his neighbors. Let vector $f^{(\tau)} \in \mathbb{R}^{|\mathscr{V}|}$ denote the states of all users in $\mathscr{V}$ at time τ, i.e., the proportion of stimulation at users in $\mathscr{V}$ at τ. The change of stimulation at u_i at time $\tau + \Delta t$ is defined as follows:

$$\frac{f^{(\tau+\Delta t)}(i) - f^{(\tau)}(i)}{\Delta t} = \alpha \sum_{u_j \in \mathscr{V}} p_{ji}(f^{(\tau)}(j) - f^{(\tau)}(i)), \tag{29}$$

where coefficient α can be set as 1 as proposed in [64]. The *transition probabilities* $p_{ij}, i, j \in \{1, 2, \cdots, |\mathscr{V}|\}$ can be represented with the *transition matrix*

$$\mathbf{X} = (\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}) \tag{30}$$

of network G, where $\mathbf{X} \in \mathbb{R}^{|\mathscr{V}|\times|\mathscr{V}|}$, $X(i,j) = p_{ij}$ and diagonal matrix $\mathbf{D} \in \mathbb{R}^{|\mathscr{V}|\times|\mathscr{V}|}$ has value $D(i,i) = \sum_{j=1}^{|\mathscr{V}|} A(i,j)$ on its diagonal.

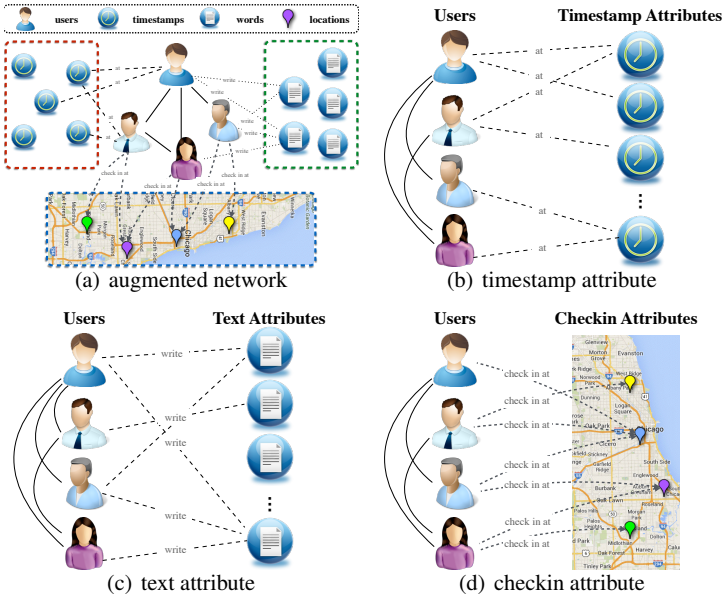
Definition 3. (Social Transition Probability Matrix): The *social transition probability matrix* of network G can be represented as $\mathbf{Q} = \mathbf{X} - \mathbf{D}_{\mathbf{X}}$, where $\mathbf{X}$ is the *transition matrix* defined above and diagonal matrix $D_{\mathbf{X}}$ has value $D_{\mathbf{X}}(i,i) = \sum_{j=1}^{|\mathscr{V}|} \mathbf{X}(i,j)$ on its diagonal.

Furthermore, by setting $\Delta t = 1$, denoting that stimulation propagates step by step in a discrete time through network, we can rewrite the propagation updating equation as:

$$f^{(\tau)} = f^{(\tau-1)} + \alpha(\mathbf{X} - \mathbf{D}_{\mathbf{X}})f^{(\tau-1)} = (\mathbf{I} + \alpha\mathbf{Q})f^{(\tau-1)} \tag{31}$$

$$= (\mathbf{I} + \alpha\mathbf{Q})^{\tau} f^{(0)}. \tag{32}$$

Such a propagation process will stop when $f^{(\tau)} = f^{(\tau-1)}$, i.e.,



(a) augmented network (b) timestamp attribute (c) text attribute (d) checkin attribute

Fig. 4 An example of attribute augmented heterogeneous network. (a): attribute augmented heterogeneous network, (b): timestamp attribute, (c): text attribute, (d): location checkin attribute.

$$(\mathbf{I}+\alpha\mathbf{Q})^{(\tau)} = (\mathbf{I}+\alpha\mathbf{Q})^{(\tau-1)}. \tag{33}$$

The smallest τ that can stop the propagation is defined as the *stop step*. To obtain the *stop step* τ, CAD need to keep checking the powers of $(\mathbf{I}+\alpha\mathbf{Q})$ until it doesn't change as τ increases, i.e., the *stop criteria*.

Definition 4. (Intimacy Matrix): Matrix

$$\mathbf{H} = (\mathbf{I}+\alpha\mathbf{Q})^{\tau} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|} \tag{34}$$

is defined as the *intimacy matrix* of users in $\mathcal{V}$, where τ is the *stop step* and $H(i,j)$ denotes the *intimacy score* between u_i and $u_j \in \mathcal{V}$ in the network.

3.4 Intimacy Matrix of Attributed Heterogeneous Network

Real-world social networks can usually contain various kinds of information, e.g., links and attributes, and can be formulated as $G = (\mathcal{V}, \mathcal{E}, \mathcal{A})$ as introduced in Section 3.2. Attribute set $\mathcal{A} = \{a_1, a_2, \cdots, a_m\}$, $a_i = \{a_{i1}, a_{i2}, \cdots, a_{in_i}\}$, can have n_i different values for $i \in \{1, 2, \cdots, m\}$. An example of attribute augmented heterogeneous network is given in Figure 4, where Figure 4(a) is the input *attribute augmented heterogeneous network*. Figures 4(b)-4(d) show the attribute information

in the network, which include timestamps, text and location checkins. Including the attributes as a special type of nodes in the graph definition provides a conceptual framework to handle social links and node attributes in a unified framework. The effect on increasing the dimensionality of the network will be handled as in Lemma 3.4 in lower dimensional space.

Definition 5. (Attribute Transition Probability Matrix): The connections between users and attributes, e.g., a_i, can be represented as the *attribute adjacency matrix* $\mathbf{A}_{a_i} \in \mathbb{R}^{|\mathscr{V}|\times n_i}$. Based on $\mathbf{A}_{a_i}$, CAD formally defines the *attribute transition probability matrix* from users to attribute a_i to be $\mathbf{R}_i \in \mathbb{R}^{|\mathscr{V}|\times n_i}$, where

$$\mathbf{R}_i(i,j) = \frac{1}{\sqrt{(\sum_{m=1}^{n_i}\mathbf{A}_{a_i}(i,m))(\sum_{n=1}^{|\mathscr{V}|}\mathbf{A}_{a_i}(n,j))}}\mathbf{A}_{a_i}(i,j). \tag{35}$$

Similarly, CAD defines the *attribute transition probability matrix* from attribute a_i to users in $\mathscr{V}$ as $\mathbf{S}_i = \mathbf{R}_i^T$.

The importance of different information types in calculating the closeness measure among users can be different. To handle the *network heterogeneity problem*, the CAD model proposes to apply the *micro-level* control by giving different information sources distinct weights to denote their differences: $\omega = [\omega_0, \omega_1, \cdots, \omega_m]^\top$, where $\sum_{i=0}^{m}\omega_i = 1.0$, ω_0 is the weight of link information and ω_i is the weight of attribute a_i, for $i \in \{1,2,\cdots,m\}$.

Definition 6. (Weighted Attribute Transition Probability Matrix): With weights ω, CAD can define matrices

$$\tilde{\mathbf{R}} = [\omega_1\mathbf{R}_1, \cdots, \omega_n\mathbf{R}_n], \tag{36}$$

$$\tilde{\mathbf{S}} = [\omega_1\mathbf{S}_1, \cdots, \omega_n\mathbf{S}_n]^T \tag{37}$$

to be the *weighted attribute transition probability matrices* between users and all attributes, where $\tilde{\mathbf{R}} \in \mathbb{R}^{|\mathscr{V}|\times(n_{aug}-|\mathscr{V}|)}$, $\tilde{\mathbf{S}} \in \mathbb{R}^{(n_{aug}-|\mathscr{V}|)\times|\mathscr{V}|}$, $n_{aug} = (|\mathscr{V}| + \sum_{i=1}^{m} n_i)$ is the number of all user and attribute nodes in the augmented network.

Definition 7. (Network Transition Probability Matrix): Furthermore, the *transition probability matrix* of the whole attribute augmented heterogeneous network G is defined as

$$\tilde{\mathbf{Q}}_{aug} = \begin{bmatrix}\tilde{\mathbf{Q}} & \tilde{\mathbf{R}}\\ \tilde{\mathbf{S}} & \mathbf{0}\end{bmatrix}, \tag{38}$$

where $\tilde{\mathbf{Q}}_{aug} \in \mathbb{R}^{n_{aug}\times n_{aug}}$ and block matrix $\tilde{\mathbf{Q}} = \omega_0\mathbf{Q}$ is the *weighted social transition probability matrix* of social links in $\mathscr{E}$.

In the real world, heterogeneous social networks can contain large amounts of attributes, i.e., n_{aug} can be extremely large. The *weighted transition probability matrix*, i.e., $\tilde{\mathbf{Q}}_{aug}$, can be of extremely high dimensions and can hardly fit in the memory. As a result, it will be impossible to update the matrix until the *stop criteria*

meets to obtain the *stop step* and the *intimacy matrix*. To solve such problem, CAD proposes to obtain the *stop step* and the *intimacy matrix* by applying partitioned block matrix operations with the following Lemma 3.4.

Lemma 1. $(\tilde{\mathbf{Q}}_{aug})^k = \begin{bmatrix} \tilde{\mathbf{Q}}_k & \tilde{\mathbf{Q}}_{k-1}\tilde{\mathbf{R}} \\ \tilde{\mathbf{S}}\tilde{\mathbf{Q}}_{k-1} & \tilde{\mathbf{S}}\tilde{\mathbf{Q}}_{k-2}\tilde{\mathbf{R}} \end{bmatrix}$, $k \geq 2$, *where*

$$\tilde{\mathbf{Q}}_k = \begin{cases} \mathbf{I}, & \text{if } k = 0, \\ \tilde{\mathbf{Q}}, & \text{if } k = 1, \\ \tilde{\mathbf{Q}}\tilde{\mathbf{Q}}_{k-1} + \tilde{\mathbf{R}}\tilde{\mathbf{S}}\tilde{\mathbf{Q}}_{k-2}, & \text{if } k \geq 2 \end{cases} \tag{39}$$

and the intimacy matrix among users in $\mathcal{V}$ can be represented as

$$\tilde{\mathbf{H}}_{aug} = \left(\mathbf{I} + \alpha\tilde{\mathbf{Q}}_{aug}\right)^{\tau}(1:|\mathcal{V}|, 1:|\mathcal{V}|) \tag{40}$$

$$= \left(\sum_{t=0}^{\tau}\binom{\tau}{t}\alpha^t(\tilde{\mathbf{Q}}_{aug})^t\right)(1:|\mathcal{V}|, 1:|\mathcal{V}|) \tag{41}$$

$$= \left(\sum_{t=0}^{\tau}\binom{\tau}{t}\alpha^t\left((\tilde{\mathbf{Q}}_{aug})^t(1:|\mathcal{V}|, 1:|\mathcal{V}|)\right)\right) \tag{42}$$

$$= \left(\sum_{t=0}^{\tau}\binom{\tau}{t}\alpha^t\tilde{\mathbf{Q}}_t\right), \tag{43}$$

where $\mathbf{X}(1:|\mathcal{V}|, 1:|\mathcal{V}|)$ is a sub-matrix of $\mathbf{X}$ with indexes in range $[1, |\mathcal{V}|]$, τ is the stop step, achieved when $\tilde{\mathbf{Q}}_\tau = \tilde{\mathbf{Q}}_{\tau-1}$, i.e., the stop criteria, $\tilde{\mathbf{Q}}_\tau$ is called the stationary matrix of the attributed augmented heterogeneous network.

Proof. The lemma can be proved by induction on k [63], which will be left as an exercise for the readers. Considering that $(\tilde{\mathbf{R}}\tilde{\mathbf{S}}) \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ can be precomputed in advance, the space cost of Lemma 3.4 is $O(|\mathcal{V}|^2)$, where $|\mathcal{V}| \ll n_{aug}$.

Since we are only interested in the *intimacy* and *transition matrices* among user nodes instead of those between the augmented items and users for the community detection task, CAD creates a reduced dimensional representation only involving users for $\tilde{\mathbf{Q}}_k$ and $\tilde{\mathbf{H}}$ such that CAD can capture the effect of "user-attribute" and "attribute-user" transition on "user-user" transition. $\tilde{\mathbf{Q}}_k$ is a reduced dimension representation of $\tilde{\mathbf{Q}}^k_{aug}$, while eliminating the augmented items, it can still capture the "user-user" transitions effectively.

3.5 Intimacy Matrix across Aligned Heterogeneous Networks

When G^t is new, the *intimacy matrix* $\tilde{\mathbf{H}}$ among users calculated based on the information in G^t can be very sparse. To solve this problem, CAD proposes to propagate

useful information from other well developed aligned networks to the emerging network. Information propagated from other aligned well-developed networks can help solve the shortage of information problem in the emerging network [53, 54]. However, as proposed in [32], different networks can have different properties and information propagated from other well-developed aligned networks can be very different from that of the emerging network as well.

To handle this problem, CAD model proposes to apply the *macro-level control* technique by using weights, $\rho^{s,t}, \rho^{t,s} \in [0,1]$, to control the proportion of information propagated between developed network G^s and emerging network G^t. If information from G^s is helpful for improving the community detection results in G^t, CAD can set a higher $\rho^{s,t}$ to propagate more information from G^s. Otherwise, CAD can set a lower $\rho^{s,t}$ instead. The weights $\rho^{s,t}$ and $\rho^{t,s}$ can be adjusted automatically with method to be introduced in Subsection 3.7.

Definition 8. (Anchor Transition Matrix): To propagate information across networks, CAD introduces the *anchor transition matrices* between G^t and G^s to be $\mathbf{T}^{t,s} \in \mathbb{R}^{|\mathcal{V}^t|\times|\mathcal{V}^s|}$ and $\mathbf{T}^{s,t} \in \mathbb{R}^{|\mathcal{V}^s|\times|\mathcal{V}^t|}$, where $\mathbf{T}^{t,s}(i,j) = \mathbf{T}^{s,t}(j,i) = 1$, iff $(u_i^t, u_j^s) \in A^{t,s}, u_i^t \in \mathcal{V}^t, u_j^s \in \mathcal{V}^s$.

Meanwhile, with weights $\rho^{s,t}$ and $\rho^{t,s}$, the *weighted network transition probability matrix* of G^t and G^s are represented as

$$\bar{\mathbf{Q}}^t_{aug} = (1-\rho^{t,s}) \begin{bmatrix} \tilde{\mathbf{Q}}^t & \tilde{\mathbf{R}}^t \\ \tilde{\mathbf{S}}^t & \mathbf{0} \end{bmatrix} \tag{44}$$

and

$$\bar{\mathbf{Q}}^s_{aug} = (1-\rho^{s,t}) \begin{bmatrix} \tilde{\mathbf{Q}}^s & \tilde{\mathbf{R}}^s \\ \tilde{\mathbf{S}}^s & \mathbf{0} \end{bmatrix}, \tag{45}$$

where $\bar{\mathbf{Q}}^t_{aug} \in \mathbb{R}^{n^t_{aug}\times n^t_{aug}}$ and $\bar{\mathbf{Q}}^s_{aug} \in \mathbb{R}^{n^s_{aug}\times n^s_{aug}}$, n^t_{aug} and n^s_{aug} are the numbers of all nodes in G^t and G^s respectively.

Furthermore, to accommodate the dimensions, CAD introduces the *weighted anchor transition matrices* between G^s and G^t to be

$$\bar{\mathbf{T}}^{t,s} = (\rho^{t,s}) \begin{bmatrix} \mathbf{T}^{t,s} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \tag{46}$$

$$\bar{\mathbf{T}}^{s,t} = (\rho^{s,t}) \begin{bmatrix} \mathbf{T}^{s,t} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \tag{47}$$

where $\bar{\mathbf{T}}^{t,s} \in \mathbb{R}^{n^t_{aug}\times n^s_{aug}}$ and $\bar{\mathbf{T}}^{s,t} \in \mathbb{R}^{n^s_{aug}\times n^t_{aug}}$. Nodes corresponding to entries in $\bar{\mathbf{T}}^{t,s}$ and $\bar{\mathbf{T}}^{s,t}$ are of the same order as those in $\bar{\mathbf{Q}}^t_{aug}$ and $\bar{\mathbf{Q}}^s_{aug}$ respectively.

By combining the weighted intra-network transition probability matrices together with the weighted anchor transition matrices, CAD defines the *transition probability matrix* across *aligned networks* as

$$\bar{\mathbf{Q}}_{align} = \begin{bmatrix} \bar{\mathbf{Q}}^t_{aug} & \bar{\mathbf{T}}^{t,s} \\ \bar{\mathbf{T}}^{s,t} & \bar{\mathbf{Q}}^s_{aug} \end{bmatrix} \tag{48}$$

where $\bar{\mathbf{Q}}_{align} \in \mathbb{R}^{n_{align} \times n_{align}}$, $n_{align} = n^t_{aug} + n^s_{aug}$ is the number of all nodes across the aligned networks.

Definition 9. (Aligned Network Intimacy Matrix): Based on the previous remarks, with $\bar{\mathbf{Q}}_{align}$, CAD can obtain the the *intimacy matrix*, $\bar{\mathbf{H}}_{align}$, of users in G^t to be

$$\bar{\mathbf{H}}_{align} = (\mathbf{I} + \alpha\bar{\mathbf{Q}}_{align})^{\tau}(1 : |\mathcal{V}^t|, 1 : |\mathcal{V}^t|), \tag{49}$$

where $\bar{\mathbf{H}}_{align} \in \mathbb{R}^{|\mathcal{V}^t| \times |\mathcal{V}^t|}$, τ is the *stop step*.

Meanwhile, the structure of $(\mathbf{I} + \alpha\bar{\mathbf{Q}}_{align})$ can not meet the requirements of Lemma 3.4 as it doesn't have a zero square matrix at the bottom right corner. As a result, methods introduced in Lemma 3.4 cannot be applied. To obtain the *stop step*, there is no other choice but to keep calculating powers of $(\mathbf{I} + \alpha\bar{\mathbf{Q}}_{align})$ until the *stop criteria* can meet, which can be very time consuming. In this part, we will introduce with the following Lemma 3.5 adopted by CAD model for efficient computation of the high-order powers of matrix $(\mathbf{I} + \alpha\bar{\mathbf{Q}}_{align})$.

Lemma 2. *For the given matrix $(\mathbf{I} + \alpha\bar{\mathbf{Q}}_{align})$, its k_{th} power meets*

$$(\mathbf{I} + \alpha\bar{\mathbf{Q}}_{align})^k\mathbf{P} = \mathbf{P}\Lambda^k, k \geq 1, \tag{50}$$

matrices $\mathbf{P}$ and Λ contain the eigenvector and eigenvalues of $(\mathbf{I} + \alpha\bar{\mathbf{Q}}_{align})$. The i_{th} column of matrix $\mathbf{P}$ is the eigenvector of $(\mathbf{I} + \alpha\bar{\mathbf{Q}}_{align})$ corresponding to its i_{th} eigenvalue λ_i and diagonal matrix Λ has value $\Lambda(i,i) = \lambda_i$ on its diagonal.

Proof. The Lemma can be proved by induction on k [34] as follows:
Base Case: When $k = 1$, let $\mathbf{p}_i$ and λ_i be the i_{th} eigenvector and eigenvalue of matrix $\mathbf{Q}$ respectively, where

$$\mathbf{Q}\mathbf{p}_i = \lambda_i\mathbf{p}_i. \tag{51}$$

Organizing all the eigenvectors and eigenvalues of $\mathbf{Q}$ in matrix $\mathbf{P}$ and Λ, we can have

$$\mathbf{Q}^1\mathbf{P} = \mathbf{P}\Lambda^1. \tag{52}$$

Inductive Assumption: When $k = m, m \geq 1$, let's assume the lemma holds when $k = m, m \geq 1$. In other words, the following equation holds:

$$\mathbf{Q}^m\mathbf{P} = \mathbf{P}\Lambda^m. \tag{53}$$

Induction: When $k = m+1, m \geq 1$,

$$\mathbf{Q}^{(m+1)}\mathbf{P} = \mathbf{Q}\mathbf{Q}^m\mathbf{P} = \mathbf{Q}\mathbf{P}\Lambda^m = \mathbf{P}\Lambda\Lambda^m = \mathbf{P}\Lambda^{(m+1)}. \tag{54}$$

In sum, the lemma holds for $k \geq 1$.

The time cost of calculating Λ^k is $O(n_{align})$, which is far less than that required to calculate $(\mathbf{I} + \alpha\bar{\mathbf{Q}}_{align})^k$.

Definition 10. (Eigen-decomposition based Aligned Network Intimacy Matrix): In addition, if $\mathbf{P}$ is invertible, we can have

$$(\mathbf{I}+\alpha\bar{\mathbf{Q}}_{align})^k = \mathbf{P}\Lambda^k\mathbf{P}^{-1}, \tag{55}$$

where Λ^k has $\Lambda(i,i)^k$ on its diagonal. And the intimacy calculated based on eigenvalue decomposition will be

$$\bar{\mathbf{H}}_{align} = \left(\mathbf{P}\Lambda^{\tau}\mathbf{P}^{-1}\right)(1:|\mathscr{V}^t|, 1:|\mathscr{V}^t|). \tag{56}$$

where the *stop step* τ can be obtained when $\mathbf{P}\Lambda^{\tau}\mathbf{P}^{-1} = \mathbf{P}\Lambda^{\tau-1}\mathbf{P}^{-1}$, i.e., *stop criteria*.

3.6 Approximated Intimacy to Reduce Dimension

Eigendecomposition based method proposed in Lemma 3.5 enables CAD to calculate the powers of $(\mathbf{I}+\alpha\mathbf{Q}_{align})$ very efficiently. However, when applying Lemma 3.5 to calculate the *intimacy matrix* of real-world partially aligned networks, it can suffer from many serious problems. The reason is that the dimension of $(\mathbf{I}+\alpha\mathbf{Q}_{align})$, i.e., $n_{align}\times n_{align}$, is so high that matrix $(\mathbf{I}+\alpha\mathbf{Q}_{align})$ can hardly fit in the memory. To solve that problem, CAD proposes to calculate the approximated *intimacy matrix* $\bar{\mathbf{H}}^{approx}_{align}$ with less space and time costs instead.

Let's define the transition probability matrices of G^t and G^s to be $\tilde{\mathbf{Q}}^t_{aug}$ and $\tilde{\mathbf{Q}}^s_{aug}$ respectively. By applying Lemma 3.4, we can get their *stop step* and the *stationary matrices* to be τ^t, τ^s, $\tilde{\mathbf{Q}}^t_{\tau^t}$ and $\tilde{\mathbf{Q}}^t_{\tau^s}$ respectively. *Stationary matrices* $\tilde{\mathbf{Q}}^t_{\tau^t}$, $\tilde{\mathbf{Q}}^t_{\tau^s}$ together with the *anchor transition matrix*, $\mathbf{T}^{\mathbf{t},\mathbf{s}}$ and $\mathbf{T}^{t,s}$, can be used to define a low-dimensional *reduced aligned network transition probability matrix*, which only involves users explicitly, while the effect of "attribute-user" or "user-attribute" transition is implicitly absorbed into $\tilde{\mathbf{Q}}^t_{\tau^t}$ and $\tilde{\mathbf{Q}}^s_{\tau^s}$:

$$\bar{\mathbf{Q}}^{user}_{align} = \begin{bmatrix} (1-\rho^{t,s})\tilde{\mathbf{Q}}^t_{\tau^t} & (\rho^{t,s})\mathbf{T}^{t,s} \\ (\rho^{s,t})\mathbf{T}^{s,t} & (1-\rho^{s,t})\tilde{\mathbf{Q}}^s_{\tau^s} \end{bmatrix}, \tag{57}$$

where $\bar{\mathbf{Q}}^{user}_{align} \in \mathbb{R}^{(|\mathscr{V}|^t+|\mathscr{V}^s|)^2}$ and $(|\mathscr{V}|^t+|\mathscr{V}^s|) \ll n_{align}$.

Definition 11. (Approximated Aligned Network Intimacy Matrix): Furthermore, with Lemma 3.5, we can get *intimacy matrix* of users in G^t based on $\bar{\mathbf{Q}}^{user}_{align}$ to be:

$$\bar{\mathbf{H}}^{approx}_{align} = \left(\mathbf{P}^*(\Lambda^*)^{\tau}(\mathbf{P}^*)^{-1}\right)(1:|\mathscr{V}^t|, 1:|\mathscr{V}^t|), \tag{58}$$

where $(\mathbf{I}+\alpha\bar{\mathbf{Q}}^{user}_{align}) = \mathbf{P}^*\Lambda^*(\mathbf{P}^*)^{-1}$ and τ is the *stop step*.

The approximated intimacy matrix computation method introduced above can greatly reduce the time and space costs. Let $|\mathscr{V}^t| = n^t$, the size of intimacy matrix

Algorithm 1 CAD with Parameter Self-Adjustment

Require: aligned network: $\mathscr{G} = \{\{G^t, G^s\}, \{A^{t,s}, A^{s,t}\}\}$
parameters: ω, ρ, γ, α, β and method type M
Ensure: community detection results of G^t: $\mathscr{C}$

$\omega_{old} = \omega$, $\rho_{old} = \rho, E_{old} = \infty$
for parameter $\delta \in \omega \cup \{\rho\}$ **do**
  while *True* **do**
    $\delta = (1+\gamma)\delta$ and renormalize ω if $\delta \in \omega$ to get ω_{new}, ρ_{new}
    construct transition probability matrix $\bar{\mathbf{Q}}_{align}$
    if $M = approximation$ **then**
      construct $\tilde{\mathbf{Q}}^{user}_{align}$ with $\tilde{\mathbf{Q}}^t_{\tau^t}$, $\tilde{\mathbf{Q}}^s_{\tau^s}$ calculated according to Lemma 1
      calculate $\bar{\mathbf{H}}^{approx}_{align}$ with $\bar{\mathbf{Q}}^{user}_{align}$ according to Lemma 2
      $\bar{\mathbf{H}}_{align} = \bar{\mathbf{H}}^{approx}_{align}$
    else
      calculate $\bar{\mathbf{H}}_{align}$ with $\bar{\mathbf{Q}}_{align}$ according to Lemma 2
    end if
    get lower-dimensional latent feature vectors $\mathbf{U}$
    $\mathscr{C} = Kmeans(\mathbf{U})$
    $E_{new} = -\sum_{i=1}^{K} P(i)\log P(i), P(i) = \frac{|U_i|}{\sum_{i=1}^{K}|U_i|}, U_i \in \mathscr{C}$
    if $E_{new} < E_{old}$ **then**
      $\omega_{old} = \omega, \rho_{old} = \rho, E_{old} = E_{new}$
    else
      $\omega = \omega_{old}, \rho = \rho_{old}$
      break
    end if
  end while
end for

$\bar{\mathbf{H}}_{align}$ will be $(n^t)^2$. However, to obtain $\bar{\mathbf{H}}_{align}$, we need to calculate the transition probability matrix $\bar{\mathbf{Q}}_{align}$ in advance, whose size is $(n_{align})^2$.

Space Cost: In eigendecomposition based method, we have to calculated and store matrices $\bar{\mathbf{Q}}^{eigen}_{align}$, $\mathbf{P}$, $\mathbf{P}^{-1}$, $\Lambda \in \mathbb{R}^{n_{align}\times n_{align}}$, whose space costs are $\mathrm{O}(4n^2_{align})$. However, in the approximation based method, we just need to store matrices $\tilde{\mathbf{Q}}^x \in \mathbb{R}^{n^x \times n^x}$, $\tilde{\mathbf{R}}^x \in \mathbb{R}^{n^x \times (\sum_i n^x_i)}$, $\tilde{\mathbf{S}}^x \in \mathbb{R}^{(\sum_i n^x_i)\times n^x}, x \in \{s,t\}$, as well as $\bar{\mathbf{Q}}^{approx}_{align} \in \mathbb{R}^{(n^t+n^s)\times(n^t+n^s)}$, whose space cost will be $\mathrm{O}(\max\{(n^t+n^s)^2, n^t(\sum_i n^t_i), n^s(\sum_i n^s_i)\}) < \mathrm{O}(4n^2_{align})$.

Time Cost: In eigendecomposition based method, the matrix eigendecomposition of $\bar{\mathbf{Q}}^{eigen}_{align}$, inversion $\mathbf{P}^{-1}$ and multiplication of $\mathbf{P}\Lambda^k\mathbf{P}^{-1}$ are all time-consuming operations, whose time costs are $\mathrm{O}(kn^2_{align})$ [46], $\mathrm{O}(n^2_{align}\log(n_{align}))$ [11] and $\mathrm{O}(2n^3_{align})$ respectively. As a result, the time cost of eigendecomposition based method is about $\mathrm{O}(2n^3_{align})$. However, in approximation based methods, we need to apply Lemma 2 to get $\bar{\mathbf{H}}^t$ and $\bar{\mathbf{H}}^t$, whose time cost is

$$\mathrm{O}(\max\{\tau((n^t)^3 + (n^t)^2(\sum_i a^t_i)), \tau((n^s)^3 + (n^s)^2(\sum_i a^s_i))\}), \tag{59}$$

which is much smaller than that of eigendecomposition based methods.

3.7 Clustering and Weight Self-Adjustment

Intimacy matrix $\bar{\mathbf{H}}_{align}$ (or $\bar{\mathbf{H}}_{align}^{approx}$) stores the intimacy scores among users in $\mathscr{V}^t$ and can be used to detect the communities in the network. CAD will use the low-rank matrix factorization method used proposed in [43] to get the latent feature vectors, $\mathbf{U}$, for each user. To avoid overfitting, CAD introduces two regularization terms to the object function as follows:

$$\min_{\mathbf{U},\mathbf{V}} \left\| \bar{\mathbf{H}}_{align} - \mathbf{U}\mathbf{V}\mathbf{U}^T \right\|_F^2 + \theta \left\| \mathbf{U} \right\|_F^2 + \beta \left\| \mathbf{V} \right\|_F^2, \tag{60}$$

$$s.t. \mathbf{U} \geq \mathbf{0}, \mathbf{V} \geq \mathbf{0}, \tag{61}$$

where $\mathbf{U}$ is the latent feature vectors, $\mathbf{V}$ stores the correlation among rows of $\mathbf{V}$, θ and β are the weights of $\|\mathbf{U}\|_F^2$, $\|\mathbf{V}\|_F^2$ respectively.

This object function is hard to solve and obtaining the global optimal result for both $\mathbf{U}$ and $\mathbf{V}$ simultaneously can be very challenging. CAD proposes to solve the objective function by fixing one variable, e.g., $\mathbf{U}$, and update another variable, e.g.,$\mathbf{V}$, alternatively. The Lagrangian function of the object equation can be represented as:

$$\mathscr{F} = Tr(\bar{\mathbf{H}}_{align}\bar{\mathbf{H}}_{align}^T) - Tr(\bar{\mathbf{H}}_{align}\mathbf{U}\mathbf{V}^T\mathbf{U}^T) \tag{62}$$

$$- Tr(\mathbf{U}\mathbf{V}\mathbf{U}^T\bar{\mathbf{H}}_{align}^T) + Tr(\mathbf{U}\mathbf{V}\mathbf{U}^T\mathbf{U}\mathbf{V}^T\mathbf{U}^T) \tag{63}$$

$$+ \theta Tr(\mathbf{U}\mathbf{U}^T) + \beta Tr(\mathbf{V}\mathbf{V}^T) - Tr(\Theta\mathbf{U}) - Tr(\Omega\mathbf{V}) \tag{64}$$

where Θ and Ω are the multiplier for the constraint of $\mathbf{U}$ and $\mathbf{V}$ respectively. By taking derivatives of $\mathscr{F}$ with regarding to $\mathbf{U}$ and $\mathbf{V}$, we can get

$$\frac{\partial \mathscr{F}}{\partial \mathbf{U}} = -2(\bar{\mathbf{H}}_{align}^T\mathbf{U}\mathbf{V} + \bar{\mathbf{H}}_{align}\mathbf{U}\mathbf{V}^T - \mathbf{U}\mathbf{V}^T\mathbf{U}^T\mathbf{U}\mathbf{V}^T - \mathbf{U}\mathbf{V}\mathbf{U}^T\mathbf{U}\mathbf{V}^T - \theta\mathbf{U}) - \Theta^T \tag{65}$$

$$\frac{\partial \mathscr{F}}{\partial \mathbf{V}} = -2(\mathbf{U}^T\bar{\mathbf{H}}_{align}\mathbf{U} - \mathbf{U}^T\mathbf{U}\mathbf{V}\mathbf{U}^T\mathbf{U} - \beta\mathbf{V}) - \Omega^T \tag{66}$$

Let $\frac{\partial \mathscr{F}}{\partial \mathbf{U}} = \mathbf{0}$ and $\frac{\partial \mathscr{F}}{\partial \mathbf{V}} = \mathbf{0}$ and use the KKT complementary condition, we can get

$$\mathbf{U}(i,j) \leftarrow \mathbf{U}(i,j)\sqrt{\frac{\left(\bar{\mathbf{H}}_{align}^T\mathbf{U}\mathbf{V} + \bar{\mathbf{H}}_{align}\mathbf{U}\mathbf{V}^T\right)(i,j)}{\left(\mathbf{U}\mathbf{V}^T\mathbf{U}^T\mathbf{U}\mathbf{V} + \mathbf{U}\mathbf{V}\mathbf{U}^T\mathbf{U}\mathbf{V}^T + \theta\mathbf{U}\right)(i,j)}}, \tag{67}$$

$$\mathbf{V}(i,j) \leftarrow \mathbf{V}(i,j)\sqrt{\frac{\left(\mathbf{U}^T\bar{\mathbf{H}}_{align}\mathbf{U}\right)(i,j)}{\left(\mathbf{U}^T\mathbf{U}\mathbf{V}\mathbf{U}^T\mathbf{U} + \beta\mathbf{V}\right)(i,j)}}. \tag{68}$$

The low-rank matrix $\mathbf{U}$ captures the information of each users from the intimacy matrix and can be used as latent numerical feature vectors to cluster users in G^t with traditional clustering methods, e.g., Kmeans [15].

Meanwhile, to handle the *information heterogeneity problem* in each network and the *network difference problem* across networks, CAD uses weights, ω^t, ω^s, $\rho^{t,s}$ and $\rho^{s,t}$, to denote the importance of information in G^t, G^s and that propagated from G^t and G^s respectively. For simplicity, CAD sets $\omega^t = \omega^s = \omega = [\omega_0, \omega_1, \cdots, \omega_m]$ and $\rho^{t,s} = \rho^{s,t} = \rho$ in CAD. Let $\mathscr{C}$ be the community detection result achieved by CAD in G^t. The optimal optimal choices of parameters ω and ρ, evaluated by some metrics, e.g., *entropy* [64], can be achieved with the following equation:

$$\omega, \rho = \min_{\omega,\rho} E(\mathscr{C}). \tag{69}$$

The optimization problem is very difficult to solve. CAD proposes a method to adjust ω and ρ automatically to enable CAD to achieve better results.

The weight adjustment method used to deal with ω can work as follows: for example, in network G^t, we have relational information and attribute information $\mathscr{E}$ and $\mathscr{A} = \{A_1, A_2, \cdots, A_m\}$, whose weights are initialized to be $\omega = \{\omega_0, \omega_1, \cdots, \omega_m\}$. For $\omega_i \in \omega, i \in \{0, 1, \cdots, m\}$, CAD keeps checking if increasing ω_i by a ratio of γ, i.e., $(1+\gamma)\omega_i$, can improve the performance or not. If so, $(1+\gamma)\omega_i$ after re-normalization is used as the new value of ω_i; otherwise, CAD restores the old ω_i before increase and study ω_{i+1}. In the experiment, γ is set as 0.05. Similarly, for the weight of different networks, i.e, ρ, CAD can adjust them with the same methods to find the optimal ρ. The pseudo code of CAD is available in Algorithm 1.

4 Mutual Community Detection

Besides the knowledge transfer from developed networks to the emerging networks to overcome the cold start problem, information in developed networks can also be transferred mutually to help refine the detected community structure detected from each of them. In this section, we will introduce the mutual community detection problem across multiple aligned heterogeneous networks and introduce a new cross-network mutual community detection model MCD. To refine the community structures, a new concept named *discrepancy* is introduced to help preserve the consensus of the community detection result of the shared anchor users [57].

4.1 Background Knowledge

In this section, we will focus on the simultaneous community detection of each network across multiple *partially aligned social networks* simultaneously, which is formally defined as the *Mutual Community Detection* problem. The goal is to distill

relevant information from other aligned social network to compliment knowledge directly derivable from each network to improve the clustering or community detection, while preserving the distinct characteristics of each individual network. The *Mutual Community Detection* problem is very important for online social networks and can be the prerequisite for many concrete social network applications: (1) *network partition*: detected communities can usually represent small-sized subgraphs of the network, and (2) *comprehensive understanding of user social behaviors*: community structures of the shared users in multiple aligned networks can provide a complementary understanding of their social interactions in online social world.

Besides its importance, the *Mutual Community Detection* problem is a novel problem and different from existing clustering problems, including: (1) *consensus clustering* [12, 23, 31, 27, 26], which aims at achieving a consensus result of several input clustering results about the same data; (2) *multi-view clustering* [4, 6], whose target is to partition objects into clusters based on their different representations, e.g., clustering webpages with text information and hyperlinks; (3) *multi-relational clustering* [49, 3], which focuses on clustering objects in one relation (called target relation) using information in multiple inter-linked relations; and (4) *co-regularized multi-domain graph clustering* [7], which relaxes the *one-to-one* constraints on node correspondence relationships between different views in multi-view clustering to "*uncertain*" mappings. Unlike these existing clustering problems, the *Mutual Community Detection* problem aims at detecting the communities for multiple networks involving both anchor and non-anchor users simultaneously and each network contains heterogeneous information about users' social activities.

4.2 Problem Formulation

For the given multiple aligned heterogeneous networks $\mathscr{G}$, the *Mutual Community Detection* problem aims to obtain the optimal communities $\{\mathscr{C}^{(1)},\mathscr{C}^{(2)},\cdots,\mathscr{C}^{(n)}\}$ for $\{G^{(1)},G^{(2)},\cdots,G^{(n)}\}$ simultaneously, where $\mathscr{C}^{(i)}=\{U_1^{(i)},U_2^{(i)},\ldots,U_{k^{(i)}}^{(i)}\}$ is a partition of the users set $\mathscr{U}^{(i)}$ in $G^{(i)}$, $k^{(i)}=\left|\mathscr{C}^{(i)}\right|$, $U_l^{(i)}\cap U_m^{(i)}=\emptyset$, $\forall\ l,m\in\{1,2,\ldots,k^{(i)}\}$ and $\bigcup_{j=1}^{k^{(i)}}U_j^{(i)}=\mathscr{U}^{(i)}$. Users in each detected social community are more densely connected with each other than with users in other communities. In this section, we focus on studying the hard (i.e., non-overlapping) community detection of users in online social networks.

The *Mutual Community Detection* problem studied in this section is very challenging to solve due to:

- *Closeness Measure*: Users in heterogeneous social networks can be connected with each other by various direct and indirect connections. A general closeness measure among users with such connection information is the prerequisite for addressing the *Mutual Community Detection* problem.

Table 2 Summary of HNMPs.

ID	Notation	Heterogeneous Network Meta Path	Semantics
1	U $\rightarrow$ U	User $\xrightarrow{follow}$ User	Follow
2	U $\rightarrow$ U $\rightarrow$ U	User $\xrightarrow{follow}$ User $\xrightarrow{follow}$ User	Follower of Follower
3	U $\rightarrow$ U $\leftarrow$ U	User $\xrightarrow{follow}$ User $\xrightarrow{follow^{-1}}$ User	Common Out Neighbor
4	U $\leftarrow$ U $\rightarrow$ U	User $\xrightarrow{follow^{-1}}$ User $\xrightarrow{follow}$ User	Common In Neighbor
5	U $\rightarrow$ P $\rightarrow$ W $\leftarrow$ P $\leftarrow$ U	User $\xrightarrow{write}$ Post $\xrightarrow{contain}$ Word $\xrightarrow{contain^{-1}}$ Post $\xrightarrow{write^{-1}}$ User	Posts Containing Common Words
6	U $\rightarrow$ P $\rightarrow$ T $\leftarrow$ P $\leftarrow$ U	User $\xrightarrow{write}$ Post $\xrightarrow{contain}$ Time $\xrightarrow{contain^{-1}}$ Post $\xrightarrow{write^{-1}}$ User	Posts Containing Common Timestamps
7	U $\rightarrow$ P $\rightarrow$ L $\leftarrow$ P $\leftarrow$ U	User $\xrightarrow{write}$ Post $\xrightarrow{attach}$ Location $\xrightarrow{attach^{-1}}$ Post $\xrightarrow{write^{-1}}$ User	Posts Attaching Common Location Check-ins

- *Network Characteristics*: Social networks usually have their own characteristics, which can be reflected in the community structures formed by users. Preservation of each network's characteristics (i.e., some unique structures in each network's detected communities) is very important in the *Mutual Community Detection* problem.
- *Mutual Community Detection*: Information in different networks can provide us with a more comprehensive understanding about the anchor users' social structures. For anchor users whose community structures are not clear based on in formation in one network, utilizing the heterogeneous information in aligned networks to refine and disambiguate the community structures about the anchor users. However, how to achieve such a goal is still an open problem.

To solve all these challenges, a novel cross-network community detection method, MCD (Mutual Community Detector), is proposed in this section. MCD maps the complex relationships in the social network into a heterogeneous information network [41] and introduces a novel meta-path based closeness measure, *HNMP-Sim*, to utilize both direct and indirect connections among users in closeness scores calculation. With full considerations of the network characteristics, MCD exploits the information in aligned networks to refine and disambiguate the community structures of the multiple networks concurrently. More detailed information about the MCD model will be introduced as follows.

4.3 Meta Path based Social Proximity Measure

Many existing similarity measures, e.g., "Common Neighbor" [13], "Jaccard's Coefficient" [13], defined for homogeneous networks cannot capture all the connections among users in heterogeneous networks. To use both direct and indirect connections among users in calculating the similarity score among users in the heterogeneous information network, MCD introduces meta path based similarity measure HNMP-Sim, whose information will be introduced as follows.

In heterogeneous networks, pairs of nodes can be connected by different paths, which are sequences of links in the network. Meta paths [41, 42] in heterogeneous networks, i.e., *heterogeneous network meta paths* (HNMPs), can capture both direct and indirect connections among nodes in a network. The length of a meta path is defined as the number of links that constitute it. Meta paths in networks can start and end with various node types. However, in this section, we are mainly concerned about those starting and ending with users, which are formally defined as the *social HNMPs*. A formal definition of *social HNMPs* is available in [60, 57, 59]. The notation, definition and semantics of 7 different *social HNMPs* used in MCD are listed in Table 2. To extract the social meta paths, prior domain knowledge about the network structure is required.

These 7 different social HNMPs in Table 2 can cover lots of connections among users in networks. Some meta path based similarity measures have been proposed so far, e.g., the *PathSim* proposed in [41], which is defined for undirected networks and considers different meta paths to be of the same importance. To measure the social closeness among users in directed heterogeneous information networks, we extend *PathSim* to propose a new closeness measure as follows.

Definition 12. (HNMP-Sim): Let $\mathscr{P}_i(x \rightsquigarrow y)$ and $\mathscr{P}_i(x \rightsquigarrow \cdot)$ be the sets of path instances of HNMP # i going from x to y and those going from x to other nodes in the network. The HNMP-Sim (HNMP based Similarity) of node pair (x,y) is defined as

$$\text{HNMP-Sim}(x,y) = \sum_i \omega_i \left(\frac{|\mathscr{P}_i(x \rightsquigarrow y)| + |\mathscr{P}_i(y \rightsquigarrow x)|}{|\mathscr{P}_i(x \rightsquigarrow \cdot)| + |\mathscr{P}_i(y \rightsquigarrow \cdot)|} \right), \tag{70}$$

where ω_i is the weight of the i_{th} HNMP and $\sum_i \omega_i = 1$. In MCD, the weights of different HNMPs can be automatically adjusted by applying a similar greedy search technique as introduced in Section 3.7.

Let $\mathbf{A}_i$ be the *adjacency matrix* corresponding to the i_{th} HNMP among users in the network and $\mathbf{A}_i(m,n) = k$ iff there exist k different path instances of the i_{th} HNMP from user m to n in the network. Furthermore, the similarity score matrix among users of HNMP # i can be represented as $\mathbf{S}_i = \mathbf{B}_i \circ \left(\mathbf{A}_i + \mathbf{A}_i^T\right)$, where $\mathbf{A}_i^T$ denotes the transpose of $\mathbf{A}_i$ and $\mathbf{B}_i$ represents the sum of the out-degree of user x and y has values $\mathbf{B}_i(x,y) = \left(\sum_m \mathbf{A}_i(x,m) + \sum_m \mathbf{A}_i(y,m)\right)^{-1}$. The $\circ$ symbol represents the Hadamard product of two matrices. The HNMP-Sim matrix of the network which can capture all possible connections among users is represented as follows:

$$\mathbf{S} = \sum_i \omega_i \mathbf{S}_i = \sum_i \omega_i \left(\mathbf{B}_i^{-1} \circ \left(\mathbf{A}_i + \mathbf{A}_i^T\right)\right). \tag{71}$$

4.4 Network Characteristic Preservation Clustering

Clustering each network independently can preserve each networks characteristics effectively as no information from external networks will interfere with the clustering results. Partitioning users of a certain network into several clusters will cut connections in the network and lead to some costs inevitably. Optimal clustering results can be achieved by minimizing the clustering costs.

For a given network G, let $\mathscr{C} = \{U_1, U_2, \ldots, U_k\}$ be the community structures detected from G. Term $\overline{U_i} = \mathscr{U} - U_i$ is defined to be the complement of set U_i in G. Various cost measure of partition $\mathscr{C}$ can be used, e.g., *cut* and *normalized cut* as introduced in Section 4:

$$cut(\mathscr{C}) = \frac{1}{k}\sum_{i=1}^{k} S(U_i, \overline{U_i}) = \frac{1}{k}\sum_{i=1}^{k}\sum_{u\in U_i, v\in\overline{U_i}} S(u,v), \tag{72}$$

$$ncut(\mathscr{C}) = \frac{1}{k}\sum_{i=1}^{k} \frac{S(U_i, \overline{U_i})}{S(U_i,\cdot)} = \frac{1}{k}\sum_{i=1}^{k} \frac{cut(U_i, \overline{U}_i)}{S(U_i,\cdot)}, \tag{73}$$

where $S(u,v)$ denotes the HNMP-Sim between u, v and $S(U_i,\cdot) = S(U_i, \mathscr{U}) = S(U_i, U_i) + S(U_i, \overline{U}_i)$.

For all users in $\mathscr{U}$, their clustering result can be represented in the *result confidence matrix* $\mathbf{H}$, where $\mathbf{H} = [\mathbf{h_1}, \mathbf{h_2}, \ldots, \mathbf{h_n}]^{\mathbf{T}}$, $n = |\mathscr{U}|$, $\mathbf{h}_i = (h_{i,1}, h_{i,2}, \ldots, h_{i,k})$ and $h_{i,j}$ denotes the confidence that $u_i \in \mathscr{U}$ is in cluster $U_j \in \mathscr{C}$. The optimal $\mathbf{H}$ that can minimize the normalized-cut cost can be obtained by solving the following objective function [45]:

$$\min_{\mathbf{H}} \ \mathrm{Tr}(\mathbf{H}^T\mathbf{L}\mathbf{H}), \tag{74}$$

$$s.t. \ \ \mathbf{H}^T\mathbf{D}\mathbf{H} = \mathbf{I}. \tag{75}$$

where $\mathbf{L} = \mathbf{D} - \mathbf{S}$, diagonal matrix $\mathbf{D}$ has $D(i,i) = \sum_j S(i,j)$ on its diagonal, and $\mathbf{I}$ is an identity matrix.

4.5 Discrepancy based Clustering of Multiple Networks

Besides the shared information due to common network construction purposes and similar network features [55], anchor users can also have unique information (e.g., social structures) across aligned networks, which can provide us with a more comprehensive knowledge about the community structures formed by these users. Meanwhile, by maximizing the consensus (i.e., minimizing the "*discrepancy*") of the clustering results about the anchor users in multiple partially aligned networks, model MCD will be able to refine the clustering results of the anchor users with information in other aligned networks mutually. We can represent the

clustering results achieved in $G^{(1)}$ and $G^{(2)}$ as $\mathscr{C}^{(1)} = \{U_1^{(1)}, U_2^{(1)}, \cdots, U_{k^{(1)}}^{(1)}\}$ and $\mathscr{C}^{(2)} = \{U_1^{(2)}, U_2^{(2)}, \cdots, U_{k^{(2)}}^{(2)}\}$ respectively.

Let u_i and u_j be two anchor users in the network, whose accounts in $G^{(1)}$ and $G^{(2)}$ are $u_i^{(1)}$, $u_i^{(2)}$, $u_j^{(1)}$ and $u_j^{(2)}$ respectively. If users $u_i^{(1)}$ and $u_j^{(1)}$ are partitioned into the same cluster in $G^{(1)}$ but their corresponding accounts $u_i^{(2)}$ and $u_j^{(2)}$ are partitioned into different clusters in $G^{(2)}$, then it will lead to a *discrepancy* [57, 37] between the clustering results of $u_i^{(1)}$, $u_i^{(2)}$, $u_j^{(1)}$ and $u_j^{(2)}$ in aligned networks $G^{(1)}$ and $G^{(2)}$.

Definition 13. (Discrepancy): The discrepancy between the clustering results of u_i and u_j across aligned networks $G^{(1)}$ and $G^{(2)}$ is defined as the difference of confidence scores of u_i and u_j being partitioned in the same cluster across aligned networks. Considering that in the clustering results, the confidence scores of $u_i^{(1)}$ and $u_j^{(1)}$ ($u_i^{(2)}$ and $u_j^{(2)}$) being partitioned into $k^{(1)}$ ($k^{(2)}$) clusters can be represented as vectors $\mathbf{h}_i^{(1)}$ and $\mathbf{h}_j^{(1)}$ ($\mathbf{h}_i^{(2)}$ and $\mathbf{h}_j^{(2)}$) respectively, while the confidences that u_i and u_j are in the same cluster in $G^{(1)}$ and $G^{(2)}$ can be denoted as $\mathbf{h}_i^{(1)}(\mathbf{h}_j^{(1)})^T$ and $\mathbf{h}_i^{(2)}(\mathbf{h}_j^{(2)})^T$. Formally, the discrepancy of the clustering results about u_i and u_j is defined to be $d_{ij}(\mathscr{C}^{(1)}, \mathscr{C}^{(2)}) = \left(\mathbf{h}_i^{(1)}(\mathbf{h}_j^{(1)})^T - \mathbf{h}_i^{(2)}(\mathbf{h}_j^{(2)})^T\right)^2$ if u_i, u_j are both anchor users; and $d_{ij}(\mathscr{C}^{(1)}, \mathscr{C}^{(2)}) = 0$ otherwise. Furthermore, the discrepancy of $\mathscr{C}^{(1)}$ and $\mathscr{C}^{(2)}$ will be:

$$d(\mathscr{C}^{(1)}, \mathscr{C}^{(2)}) = \sum_{i}^{n^{(1)}} \sum_{j}^{n^{(2)}} d_{ij}(\mathscr{C}^{(1)}, \mathscr{C}^{(2)}), \tag{76}$$

where $n^{(1)} = |\mathscr{U}^{(1)}|$ and $n^{(2)} = |\mathscr{U}^{(2)}|$. In the definition, non-anchor users are not involved in the discrepancy calculation.

However, considering that $d(\mathscr{C}^{(1)}, \mathscr{C}^{(2)})$ is highly dependent on the number of anchor users and anchor links between $G^{(1)}$ and $G^{(2)}$, minimizing $d(\mathscr{C}^{(1)}, \mathscr{C}^{(2)})$ can favor highly consented clustering results when the anchor users are abundant but have no significant effects when the anchor users are very rare. To solve this problem, we propose to minimize the *normalized discrepancy* instead.

Definition 14. (Normalized Discrepancy) The normalized discrepancy measure computes the differences of clustering results in two aligned networks as a fraction of the discrepancy with regard to the number of anchor users across partially aligned networks:

$$nd(\mathscr{C}^{(1)}, \mathscr{C}^{(2)}) = \frac{d(\mathscr{C}^{(1)}, \mathscr{C}^{(2)})}{\left(\left|A^{(1,2)}\right|\right)\left(\left|A^{(1,2)}\right| - 1\right)}. \tag{77}$$

Optimal consensus clustering results of $G^{(1)}$ and $G^{(2)}$ will be $\hat{\mathscr{C}^{(1)}}, \hat{\mathscr{C}^{(2)}}$:

$$\hat{\mathscr{C}^{(1)}}, \hat{\mathscr{C}^{(2)}} = \arg\min_{\mathscr{C}^{(1)}, \mathscr{C}^{(2)}} nd(\mathscr{C}^{(1)}, \mathscr{C}^{(2)}). \tag{78}$$

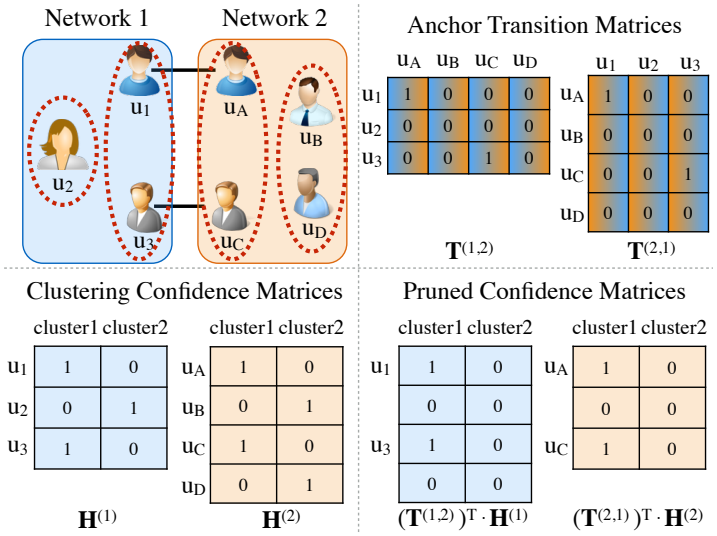


Fig. 5 An example to illustrate the clustering discrepancy.

Similarly, the normalized-discrepancy objective function can also be represented with the *clustering results confidence matrices* $\mathbf{H}^{(1)}$ and $\mathbf{H}^{(2)}$ as well. Meanwhile, considering that the networks studied in this section are partially aligned, matrices $\mathbf{H}^{(1)}$ and $\mathbf{H}^{(2)}$ contain the results of both anchor users and non-anchor users, while non-anchor users should not be involved in the discrepancy calculation according to the definition of discrepancy. We propose to prune the results of the non-anchor users with the following *anchor transition matrix* first.

Definition 15. (Anchor Transition Matrix): Binary matrix $\mathbf{T}^{(1,2)}$ (or $\mathbf{T}^{(2,1)}$) is defined as the anchor transition matrix from networks $G^{(1)}$ to $G^{(2)}$ (or from $G^{(2)}$ to $G^{(1)}$), where $\mathbf{T}^{(1,2)} = (\mathbf{T}^{(2,1)})^T$, $\mathbf{T}^{(1,2)}(i,j) = 1$ if $(u_i^{(1)}, u_j^{(2)}) \in A^{(1,2)}$ and 0 otherwise. The row indexes of $\mathbf{T}^{(1,2)}$ (or $\mathbf{T}^{(2,1)}$) are of the same order as those of $\mathbf{H}^{(1)}$ (or $\mathbf{H}^{(2)}$). Considering that the constraint on anchor links is "*one-to-one*" in this section, as a result, each row/column of $\mathbf{T}^{(1,2)}$ and $\mathbf{T}^{(2,1)}$ contains at most one entry filled with 1.

Example 6. In Figure 5, we show an example about the clustering discrepancy of two partially aligned networks $G^{(1)}$ and $G^{(2)}$, users in which are grouped into two clusters $\{\{u_1, u_3\}, \{u_2\}\}$ and $\{\{u_A, u_C\}, \{u_B, u_D\}\}$ respectively. Users u_1, u_A and u_3, u_C are identified to be anchor users, based on which we can construct the "anchor transition matrices" $\mathbf{T}^{(1,2)}$ and $\mathbf{T}^{(2,1)}$ as shown in the upper right plot. Furthermore, based on the community structure, we can construct the "*clustering confidence matrices*" as shown in the lower left plot. To obtain the clustering results of anchor users only, the *anchor transition matrix* can be applied to prune the clustering results of non-anchor users from the *clustering confidence matrices*. By multiplying

the *anchor transition matrices* $(\mathbf{T}^{(1,2)})^T$ and $(\mathbf{T}^{(2,1)})^T$ with *clustering confidence matrices* $\mathbf{H}^{(1)}$ and $\mathbf{H}^{(2)}$ respectively, we can obtain the "pruned confidence matrices" as show in the lower right plot of Figure 5. Entries corresponding anchor users u_1, u_3, u_A and u_C are preserved but those corresponding to non-anchor users are all pruned.

In this example, the clustering discrepancy of the partially aligned networks should be 0 according to the above discrepancy definition. Meanwhile, networks $G^{(1)}$ and $G^{(2)}$ are of different sizes and the pruned confidence matrices are of different dimensions, e.g., $(\mathbf{T}^{(1,2)})^T\mathbf{H}^{(1)} \in \mathbb{R}^{4\times 2}$ and $(\mathbf{T}^{(2,1)})^T\mathbf{H}^{(2)} \in \mathbb{R}^{3\times 2}$. To represent the discrepancy with the clustering confidence matrices, we need to further accommodate the dimensions of different pruned *clustering confidence matrices*. It can be achieved by multiplying one pruned *clustering confidence matrices* with the corresponding *anchor transition matrix* again, which will not prune entries but only adjust the matrix dimensions. Let $\bar{\mathbf{H}}^{(1)} = (\mathbf{T}^{(1,2)})^T\mathbf{H}^{(1)}$ and $\bar{\mathbf{H}}^{(2)} = (\mathbf{T}^{(1,2)})^T(\mathbf{T}^{(2,1)})^T\mathbf{H}^{(2)}$. In the example, we can represent the clustering discrepancy to be

$$\left\|\bar{\mathbf{H}}^{(1)}\left(\bar{\mathbf{H}}^{(1)}\right)^T - \bar{\mathbf{H}}^{(2)}\left(\bar{\mathbf{H}}^{(2)}\right)^T\right\|_F^2 = 0, \tag{79}$$

where matrix $\bar{\mathbf{H}}\bar{\mathbf{H}}^T$ indicates whether pairs of anchor users are in the same cluster or not.

Furthermore, the objective function of inferring clustering confidence matrices, which can minimize the normalized discrepancy can be represented as follows

$$\min_{\mathbf{H}^{(1)},\mathbf{H}^{(2)}} \frac{\left\|\bar{\mathbf{H}}^{(1)}\left(\bar{\mathbf{H}}^{(1)}\right)^T - \bar{\mathbf{H}}^{(2)}\left(\bar{\mathbf{H}}^{(2)}\right)^T\right\|_F^2}{\left\|\mathbf{T}^{(1,2)}\right\|_F^2\left(\left\|\mathbf{T}^{(1,2)}\right\|_F^2 - 1\right)}, \tag{80}$$

$$s.t.\ \ (\mathbf{H}^{(1)})^T\mathbf{D}^{(1)}\mathbf{H}^{(1)} = \mathbf{I}, (\mathbf{H}^{(2)})^T\mathbf{D}^{(2)}\mathbf{H}^{(2)} = \mathbf{I}. \tag{81}$$

where $\mathbf{D}^{(1)}$, $\mathbf{D}^{(2)}$ are the corresponding diagonal matrices of HNMP-Sim matrices of networks $G^{(1)}$ and $G^{(2)}$ respectively.

4.6 Joint Mutual Clustering of Multiple Networks

Normalized-Cut objective function favors clustering results that can preserve the characteristic of each network, however, normalized-discrepancy objective function favors consensus results which are mutually refined with information from other aligned networks. Taking both of these two issues into considerations, the optimal *Mutual Community Detection* results $\hat{\mathscr{C}}^{(1)}$ and $\hat{\mathscr{C}}^{(2)}$ of aligned networks $G^{(1)}$ and $G^{(2)}$ can be achieved as follows:

Algorithm 2 Curvilinear Search Method ($\mathscr{CSM}$)

Require: $\mathbf{X}_k$ C_k, Q_k and function $\mathscr{F}$
parameters $\varepsilon = \{\rho, \eta, \delta, \tau, \tau_m, \tau_M\}$
Ensure: $\mathbf{X}_{k+1}$, C_{k+1}, Q_{k+1}
1: $\mathbf{Y}(\tau) = \left(\mathbf{I} + \frac{\tau}{2}\mathbf{A}\right)^{-1}\left(\mathbf{I} - \frac{\tau}{2}\mathbf{A}\right)\mathbf{X}_k$
2: **while** $\mathscr{F}(\mathbf{Y}(\tau)) \geq \mathbf{C}_k + \rho\tau\mathscr{F}'((\mathbf{Y}(0)))$ **do**
3: $\tau = \delta\tau$
4: $\mathbf{Y}(\tau) = \left(\mathbf{I} + \frac{\tau}{2}\mathbf{A}\right)^{-1}\left(\mathbf{I} - \frac{\tau}{2}\mathbf{A}\right)\mathbf{X}_k$
5: **end while**
6: $\mathbf{X}_{k+1} = \mathbf{Y}_k(\tau)$
$Q_{k+1} = \eta Q_k + 1$
$C_{k+1} = \left(\eta Q_k C_k + \mathscr{F}(\mathbf{X}_{k+1})\right)/Q_{k+1}$
$\tau = \max\left(\min(\tau, \tau_M), \tau_m\right)$

$$\arg\min_{\mathscr{C}^{(1)},\mathscr{C}^{(2)}} \alpha \cdot ncut(\mathscr{C}^{(1)}) + \beta \cdot ncut(\mathscr{C}^{(2)}) + \theta \cdot nd(\mathscr{C}^{(1)}, \mathscr{C}^{(2)}) \tag{82}$$

where α, β and θ represents the weights of these terms and, for simplicity, α, β are both set as 1 in MCD.

By replacing $ncut(\mathscr{C}^{(1)})$, $ncut(\mathscr{C}^{(2)})$, $nd(\mathscr{C}^{(1)}, \mathscr{C}^{(2)})$ with the objective equations derived above, we can rewrite the joint objective function as follows:

$$\min_{\mathbf{H}^{(1)},\mathbf{H}^{(2)}} \alpha \cdot \mathrm{Tr}((\mathbf{H}^{(1)})^T \mathbf{L}^{(1)} \mathbf{H}^{(1)}) + \beta \cdot \mathrm{Tr}((\mathbf{H}^{(2)})^T \mathbf{L}^{(2)} \mathbf{H}^{(2)}) \tag{83}$$

$$+ \theta \cdot \frac{\left\| \bar{\mathbf{H}}^{(1)} \left(\bar{\mathbf{H}}^{(1)}\right)^T - \bar{\mathbf{H}}^{(2)} \left(\bar{\mathbf{H}}^{(2)}\right)^T \right\|_F^2}{\left\|\mathbf{T}^{(1,2)}\right\|_F^2 \left(\left\|\mathbf{T}^{(1,2)}\right\|_F^2 - 1\right)}, \tag{84}$$

$$s.t. \ \ (\mathbf{H}^{(1)})^T \mathbf{D}^{(1)} \mathbf{H}^{(1)} = \mathbf{I}, (\mathbf{H}^{(2)})^T \mathbf{D}^{(2)} \mathbf{H}^{(2)} = \mathbf{I}, \tag{85}$$

where $\mathbf{L}^{(1)} = \mathbf{D}^{(1)} - \mathbf{S}^{(1)}$, $\mathbf{L}^{(2)} = \mathbf{D}^{(2)} - \mathbf{S}^{(2)}$ and matrices $\mathbf{S}^{(1)}$, $\mathbf{S}^{(2)}$ and $\mathbf{D}^{(1)}$, $\mathbf{D}^{(2)}$ are the HNMP-Sim matrices and their corresponding diagonal matrices defined before.

The objective function is a complex optimization problem with orthogonality constraints, which can be very difficult to solve because the constraints are not only non-convex but also numerically expensive to preserve during iterations. Meanwhile, by substituting $\left(\mathbf{D}^{(1)}\right)^{\frac{1}{2}} \mathbf{H}^{(1)}$ and $\left(\mathbf{D}^{(2)}\right)^{\frac{1}{2}} \mathbf{H}^{(2)}$ with $\mathbf{X}^{(1)}$, $\mathbf{X}^{(2)}$, we can transform the objective function into a standard form of problems solvable with method proposed in [48]:

Algorithm 3 Mutual Community Detector (MCD)

Require: aligned network: $\mathscr{G} = \{\{G^{(1)}, G^{(2)}\}, \{A^{(1,2)}, A^{(2,1)}\}\}$;
number of clusters in $G^{(1)}$ and $G^{(2)}$: $k^{(1)}$ and $k^{(2)}$;
HNMP Sim matrices weight: ω;
parameters: $\varepsilon = \{\rho, \eta, \delta, \tau, \tau_m, \tau_M\}$;
function $\mathscr{F}$ and consensus term weight θ
Ensure: $\mathbf{H}^{(1)}, \mathbf{H}^{(2)}$

Calculate HNMP Sim matrices, $\mathbf{S}_i^{(1)}$ and $\mathbf{S}_i^{(2)}$
$\mathbf{S}^{(1)} = \sum_i \omega_i S_i^{(1)}$, $\mathbf{S}^{(2)} = \sum_i \omega_i S_i^{(2)}$
Initialize $\mathbf{X}^{(1)}$ and $\mathbf{X}^{(2)}$ with Kmeans clustering results on $\mathbf{S}^{(1)}$ and $\mathbf{S}^{(2)}$
Initialize $C_0^{(1)} = 0, Q_0^{(1)} = 1$ and $C_0^{(2)} = 0, Q_0^{(2)} = 1$
converge $= False$
while *converge* $= False$ **do**
  /* update $\mathbf{X}^{(1)}$ and $\mathbf{X}^{(2)}$ with $\mathscr{CSM}$ */
  $\mathbf{X}_{k+1}^{(1)}, C_{k+1}^{(1)}, Q_{k+1}^{(1)} = \mathscr{CSM}(\mathbf{X}_k^{(1)}, C_k^{(1)}, Q_k^{(1)}, \mathscr{F}, \varepsilon)$
  $\mathbf{X}_{k+1}^{(2)}, C_{k+1}^{(2)}, Q_{k+1}^{(2)} = \mathscr{CSM}(\mathbf{X}_k^{(2)}, C_k^{(2)}, Q_k^{(2)}, \mathscr{F}, \varepsilon)$
  if $\mathbf{X}_{k+1}^{(1)}$ and $\mathbf{X}_{k+1}^{(2)}$ both converge **then**
    converge $= True$
  end if
end while
$\mathbf{H}^{(1)} = \left((\mathbf{D}^{(1)})^{-\frac{1}{2}}\right)^T \mathbf{X}^{(1)}$, $\mathbf{H}^{(2)} = \left((\mathbf{D}^{(2)})^{-\frac{1}{2}}\right)^T \mathbf{X}^{(2)}$

$$\min_{\mathbf{X}^{(1)},\mathbf{X}^{(2)}} \quad \alpha \cdot (\mathrm{Tr}((\mathbf{X}^{(1)})^T \tilde{\mathbf{L}}^{(1)} \mathbf{X}^{(1)}) + \beta \cdot \mathrm{Tr}((\mathbf{X}^{(2)})^T \tilde{\mathbf{L}}^{(2)} \mathbf{X}^{(2)}) \tag{86}$$

$$+ \theta \cdot \frac{\left\| \tilde{\mathbf{T}}^{(1)} \mathbf{X}^{(1)} \left(\tilde{\mathbf{T}}^{(1)} \mathbf{X}^{(1)}\right)^T - \tilde{\mathbf{T}}^{(2)} \mathbf{X}^{(2)} \left(\tilde{\mathbf{T}}^{(2)} \mathbf{X}^{(2)}\right)^T \right\|_F^2}{\left\|\mathbf{T}^{(1,2)}\right\|_F^2 \left(\left\|\mathbf{T}^{(1,2)}\right\|_F^2 - 1\right)}), \tag{87}$$

$$s.t. \ \ (\mathbf{X}^{(1)})^T \mathbf{X}^{(1)} = \mathbf{I}, (\mathbf{X}^{(2)})^T \mathbf{X}^{(2)} = \mathbf{I}. \tag{88}$$

where $\tilde{\mathbf{L}}^{(1)} = ((\mathbf{D}^{(1)})^{-\frac{1}{2}})^T \mathbf{L}^{(1)} ((\mathbf{D}^{(1)})^{-\frac{1}{2}})$, $\tilde{\mathbf{L}}^{(2)} = ((\mathbf{D}^{(2)})^{-\frac{1}{2}})^T \mathbf{L}^{(2)} ((\mathbf{D}^{(2)})^{-\frac{1}{2}})$ and $\tilde{\mathbf{T}}^{(1)} = (\mathbf{T}^{(1,2)})^T (\mathbf{D}^{(1)})^{-\frac{1}{2}}$, $\tilde{\mathbf{T}}^{(2)} = (\mathbf{T}^{(1,2)})^T (\mathbf{T}^{(2,1)})^T (\mathbf{D}^{(2)})^{-\frac{1}{2}}$.

Wen et al. [48] propose a feasible method to solve the above optimization problems with a constraint-preserving update scheme. They propose to update one variable, e.g., $\mathbf{X}^{(1)}$, while fixing the other variable, e.g., $\mathbf{X}^{(2)}$, alternatively with the curvilinear search with Barzilai-Borwein step method until convergence. For example, when $\mathbf{X}^{(2)}$ is fixed, we can simplify the objective function into

$$\min_{\mathbf{X}} \mathscr{F}(\mathbf{X}), s.t. (\mathbf{X})^T \mathbf{X} = \mathbf{I}, \tag{89}$$

where $\mathbf{X} = \mathbf{X}^{(1)}$ and $\mathscr{F}(\mathbf{X})$ is the objective function, which can be solved with the curvilinear search with Barzilai-Borwein step method proposed in [48] to update $\mathbf{X}$ until convergence and the variable $\mathbf{X}$ after the $(k+1)_{th}$ iteration will be

$$\mathbf{X}_{k+1} = \mathbf{Y}(\tau_k), \mathbf{Y}(\tau_k) = \left(\mathbf{I} + \frac{\tau_k}{2}\mathbf{A}\right)^{-1}\left(\mathbf{I} - \frac{\tau_k}{2}\mathbf{A}\right)\mathbf{X}_k, \tag{90}$$

$$\mathbf{A} = \frac{\partial \mathscr{F}(\mathbf{X}_k)}{\partial \mathbf{X}}\mathbf{X}_k^T - \mathbf{X}_k(\frac{\partial \mathscr{F}(\mathbf{X}_k)}{\partial \mathbf{X}})^T, \tag{91}$$

where let $\hat{\tau} = \left(\frac{\mathrm{Tr}\left((\mathbf{X}_k - \mathbf{X}_{k-1})^T(\mathbf{X}_k - \mathbf{X}_{k-1})\right)}{\left|\mathrm{Tr}\left((\mathbf{X}_k - \mathbf{X}_{k-1})^T(\nabla\mathscr{F}(\mathbf{X}_k) - \nabla\mathscr{F}(\mathbf{X}_{k-1}))\right)\right|}\right)$, $\tau_k = \hat{\tau}\delta^h$, δ is the Barzilai-Borwein step size and h is the smallest integer to make τ_k satisfy

$$\mathscr{F}\left(\mathbf{Y}(\tau_k)\right) \leq C_k + \rho\tau_k\mathscr{F}'_\tau(\mathbf{Y}(0)). \tag{92}$$

Terms C, Q are defined as $C_{k+1} = \left(\eta Q_k C_k + \mathscr{F}(\mathbf{X}_{k+1})\right)/Q_{k+1}$ and $Q_{k+1} = \eta Q_k + 1, Q_0 = 1$. More detailed derivatives of the curvilinear search method (i.e., Algorithm 2) with Barzilai-Borwein step is available in [48]. Meanwhile, the pseudo-code of method MCD is available in Algorithm 3. Based on the achieved solutions $\mathbf{X}^{(1)}$ and $\mathbf{X}^{(2)}$, we can get $\mathbf{H}^{(1)} = \left(\mathbf{D}^{(1)}\right)^{-\frac{1}{2}}\mathbf{X}^{(1)}$ and $\mathbf{H}^{(2)} = \left(\mathbf{D}^{(2)}\right)^{-\frac{1}{2}}\mathbf{X}^{(2)}$.

5 Large-Scale Network Synergistic Community Detection

The community detection algorithm proposed in the previous section involves very complicated matrix operations, and works well for small-sized network data. However, when being applied to handle real-world online social networks involving millions even billions of users, they will suffer from the time complexity problem a lot. In this section, we will introduce a synergistic community detection algorithm SPMN for multiple large-scale aligned online social networks [16].

5.1 Problem Formulation

The problem to be introduced here follows the same formulation as the one introduced in Section 4, but the involved networks are of far larger sizes in terms of both node number and the social connection number. Synergistic partitioning across multiple large-scale social networks is very difficult for the following challenges:

- *Social Network*: Distinct from generic data, usually contains intricate interactions, and multiple heterogeneous networks mean that the relationships across multiple networks should be taken into consideration.
- *Network Scale*: Network size implies it is difficult for stand-alone programs to apply traditional partitioning methods and it is a difficult task to parallelize the existing stand-alone network partitioning algorithms.
- *Distributed Framework*: For distributed algorithms, load balance should be taken into consideration and how to generate balanced partitions is another challenge.

Algorithm 4 Edge Weight based Matching ($\mathscr{EWM}$)

Require: Network G_h
Maximum weight of a node $maxVW = n/k$
Ensure: A coarser network G_{h+1}
map() Function:
for node i in current data bolck **do**
  if $match[i] == -1$ **then**
    $maxIdx = -1$
    $sortByEdgeWeight(NN(i))$
    for $v_j \in NN(i)$ **do**
      if $match[j] == -1$ and $VW(i) + VW(j) < maxVW$ **then**
        maxIdx = j
      end if
      $match[i] = maxIdx$
      $match[maxIdx] = i$
    end for
  end if
end for
reduce() Function:
new $newNodeID[n+1]$
new $newVW[n+1]$
set $idx = 1$
for $i \in \{1, 2, \cdots, n\}$ **do**
  if $i < match[i]$ **then**
    set $newNodeID[match[i]] = idx$
    set $newNodeID[i] = idx$
    set $newVW[i] = newVW[match[i]] = VW(i) + VW(match[i[)$
    $idx++$
  end if
end for

To address the challenges, in this section, we will introduce a network structure based distributed network partitioning framework, namely SPMN. The SPMN model identifies the anchor nodes among the multiple networks, and selects a network as the datum network, then divides it into k balanced partitions and generate ⟨anchor node ID, partition ID⟩ pairs as the main objective. Based on the objective, SPMN coarsens the other networks (called as synergistic networks) into smaller ones, which will further divides the smallest networks into k balanced initial partitions, and tries to assign same kinds of anchor nodes into the same initial partition as many as possible. Here, anchor nodes of same kind means that they are divided into same partition in the datum network. Finally, SPMN projects the initial partitions back to the original networks.

5.2 Distributed Multilevel k-way Partitioning

In this section, we describe the heuristic framework for synergistic partitioning among multiple large scale social networks, and we call the framework SPMN. For large-sized networks, data processing in SPMN can be roughly divided into two stages: datum generation stage and network alignment stage.

When got the anchor node set $\mathscr{A}^{(1,2)}$ between networks $G^{(1)}$ and $G^{(2)}$, the SPMN framework will apply a distributed multilevel k-way partitioning method onto the

datum network to generate k balanced partitions. During this process, the anchor nodes are ignored and all the nodes are treated identically. We call this process datum generation stage. When finished, partition result of anchor nodes will be generated, SPMN stores them in a set-$Map\langle anidx, pidx\rangle$, where $anidx$ is anchor node ID and $pidx$ represents the partition ID the anchor node belongs to. After the datum generation stage, synergistic networks will be partitioned into k partitions according to the $Map\langle anidx, pidx\rangle$ to make the synergistic networks to align to the datum network, and during this process *discrepancy* and *cut* are the objectives to be minimized. We call this process network alignment stage.

Algorithms guaranteed to find out near-optimal partitions in a single network have been studied for a long period. But most of the methods are stand-alone, and performance is limited by the server's capacity. Inspired by the multilevel $k-way$ partitioning (MKP) method proposed by Karypis and Kumar [19, 18] and based on our previous work [1], SPMN uses MapReduce [9] to speedup the MKP method. As the same with other multilevel methods, MapReduce based MKP also includes three phases: coarsening, initial partitioning and un-coarsening.

Coarsening phase is a multilevel process and a sequence of smaller approximate networks $G_i = (\mathscr{V}_i, \mathscr{E}_i)$ are constructed from the original network $G_0 = (\mathscr{V}, \mathscr{E})$ and so forth, where $|\mathscr{V}_i| < |\mathscr{V}_{i-1}|, i \in \{1, 2, \cdots, n\}$. To construct coarser networks, node combination and edge collapsing should be performed. The task can be formally defined in terms of matching inside the networks [5]. A intra-network matching can be represented as a set of node pairs $\mathscr{M} = \{(v_i, v_j)\}, i \neq j$ and $(v_i, v_j) \in \mathscr{E}$, in which each node can only appear for no more than once. For a network G_i with a matching $\mathscr{M}_i$, if $(v_j, v_k) \in \mathscr{M}_i$ then v_j and v_k will form a new node $v_q \in \mathscr{V}_{i+1}$ in network G_{i+1} coarsen from G_i. The weight of v_q equals to the sum of weight v_j and v_k, besides, all the links connected to v_j or v_k in G_i will be connected to v_q in G_{i+1}. The total weight of nodes will remain unchanged during the coarsening phase but the total weight of edges and number of nodes will be greatly reduced. Let's define $W(\cdot)$ to be the sum of edge weight in the input set and $N(\cdot)$ to be the number of nodes/components in the input set. In the coarsening process, we have

$$W(\mathscr{E}_{i+1}) = W(\mathscr{E}_i) - W(\mathscr{M}_i), \tag{93}$$

$$N(\mathscr{V}_{i+1}) = N(\mathscr{V}_i) - N(\mathscr{M}_i). \tag{94}$$

Analysis in [17] shows that for the same coarser network, smaller edge-weight corresponds to smaller edge-cut. With the help of MapReduce framework, SPMN uses a local search method to implement an edge-weight based matching (EWM) scheme to collect larger edge weight during the coarsening phase. For the convenience of MapReduce, SPMN designs an emerging network representation format: each line contains essential information about a node and all its neighbors (NN), such as node ID, vertex weight (VW), edge weight (W), et al. The whole network data are distributed in distributed file system, such as HDFS [39], and each data block only contains a part of node set and corresponding connection information. Function $map()$ takes a data block as input and searches locally to find node pairs to match according to the edge weight. Function $reduce()$ is in charge of node com-

Algorithm 5 Synergistic Partitioning ($\mathscr{SP}$)

Require: Network G_h
Anchor Link Map $Map < anidx, pidx >$
Maximum weight of a node $maxVW = n/k$
Ensure: A coarser network G_{h+1}
1: Call Synergistic Partitioning-Map Function
2: Call Synergistic Partitioning-Reduce Function

bination, renaming and sorting. With the new node IDs and matching, a simple MapReduce job will be able to update the edge information and write the coarser network back onto HDFS. The complexity of EWM is $O(|\mathscr{E}|)$ in each iteration and pseudo code about EWM is shown in Algorithm 4.

After several iterations, a coarsest weighted network G_s consisting of only hundreds of nodes will be generated. For the network size of G_s, stand-alone algorithms with high computing complexity will be acceptable for initial partitioning. Meanwhile, the weights of nodes and edges of coarser networks are set to reflect the weights of the finer network during the coarsening phase, so G_s contains sufficient information to intelligently satisfy the balanced partition and the minimum edge-cut requirements. Plenty of traditional bisection methods are quite qualified for the task. In SPMN, it adopts the KL method with an $O(|\mathscr{E}|^3)$ computing complexity to divide G_s into two partitions and then take recursive invocations of KL method on the partitions to generate balanced k partitions.

Un-coarsening phase is inverse processing of coarsening phase. With the initial partitions and the matching of the coarsening phase, it is easy to run the un-coarsening process on the MapReduce cluster.

5.3 Distributed Synergistic Partitioning Process

In this section we will talk about the synergistic partitioning process in SPMN based on the synergistic networks with the knowledge of partition results of anchor nodes from datum network. The synergistic partitioning is also a MKP process but quite different from general MKP methods.

In the coarsening phase, anchor nodes are endowed with higher priority than non-anchor nodes. When choosing nodes to pair, SPMN assumes that anchor nodes and non-anchor nodes have different tendencies. Let G^d be the datum network. For an anchor node v_i in another aligned networks, at the top of its preference list, it would like to matched with another anchor node v_i, which has the same partition ID in the datum network, i.e., $pidx(G^d, v_i) = pidx(G^d, v_j)$ (here $pidx(G^d, v_i)$ denotes the community label that v_j belongs to in G^d). Second, if there is no appropriate anchor node, it would try to find a non-anchor node to pair. When planing to find a non-anchor node to pair, the anchor node, assuming to be v_i, would like to find a correct direction, and it would prefer to match with the non-anchor node v_j, which has lots of anchor nodes as neighbors with the same $pidx$ with v_i. When being matched

Algorithm 6 Synergistic Partitioning-Map

Require: Network G_h
Anchor Link Map $Map < anidx, pidx >$
Maximum weight of a node $maxVW = n/k$
Ensure: A coarser network G_{h+1}

map() Function:
for node i in current data bolck **do**
  if $match[i] == -1$ **then**
    set $flag = false$
    $sortByEdgeWeight(NN(i))$
    if $v_i \in Map < anidx, pidx >$ **then**
      for $v_j \in NN(i)$ & $match[j] == -1$ **do**
        if $v_j \in Map < anidx, pidx >$ & $Map.get(v_i) == Map.get(v_j)$ & $VW(i) + VW(j) < maxVW$ **then**
          $match[i] = j, match[j] = i$
          $flag = true$, break
        end if
      end for
      if $flag == false$, no suitable anchor node **then**
        for $v_j \in NN(i)$ & $match[j] == -1$ & $VW(v_i) + VW(v_j) < maxVW$ **do**
          $indirectNeighbor = NN(v_j)$
          $sortByEdgeWeight(NN(i))$
          for $v_k \in indirectNeighbor$ **do**
            if $v_k \in Map < anidx, pidx >$ & $Map.get(v_i) == Map.get(v_k)$ **then**
              $match[i] = j, match[j] = i$
              $flag = true$, break
            end if
          end for
          if $flag == true$ **then**
            break
          end if
        end for
      end if
    else
      $sortByEdgeWeight(NN(i))$
      for $v_j \in NN(v_i)$ & $v_j \notin Map < anidx, pidx >$ & $VW(i) + VW(j) < maxVW$ & $match[j] == -1$ **do**
        $match[i] = j$, $match[j] = i$, break
      end for
    end if
  end if
end for

together, the new node will be given the same *pidx* as the anchor node. To improve the accuracy of synergistic partitioning among multiple social networks, an anchor node will never try to combine with another anchor node with different *pidx*.

For a non-anchor node, it would prefer to be matched with an anchor node neighbor which belongs to the dominant partition in the non-anchor node's neighbors. Here, dominant partition in a node's neighbors means the number of anchor nodes with this partition ID is the largest. Next, a non-anchor node would choose a general non-anchor node to pair with. At last, a non-anchor node would not like to combine with an anchor node being part of the partitions which are in subordinate status. After combined together, the new node will be given the same *pidx* as the anchor node. To ensure the balance among the partitions, about $\frac{1}{3}$ of the nodes in the coarsest network are unlabeled.

Example 7. In Figure 6, we show a diagrammatic sketch of synergistic partitioning process. Take network G_0 for example, nodes $v_1 - v_7$ and the corresponding links information are stored on the same server and the other nodes are stored on another

Algorithm 7 Synergistic Partitioning-Reduce

Require: Network G_h
Anchor Link Map $Map < anidx, pidx >$
Maximum weight of a node $maxVW = n/k$
Ensure: A coarser network G_{h+1}
reduce() Function:
new $newNodeID[n+1]$
new $newVW[n+1]$
set $idx = 1$
for $i \in newNodeID[]$ **do**
  if $i < match[i]$ **then**
    set $newNodeID[match[i]] = idx$
    set $newNodeID[i] = idx$
    set $newVW[i] = newVW[match[i]] = VW(i) + VW(match[i])$
    $idx++$
  end if
end for
new $newPurity[idx+1]$
new $newPidx[idx+1]$
for $i \in [1, idx]$ **do**
  $newPurity[i] = \frac{purity[i]*VW(i)+purity[j]*VW(j)}{VW(i)+VW(j)}$
  $newPidx[i] = \max\{pidx[i], pidx[match[i]]\}$
end for

server. Nodes in pairs (v_1, v_2) and (v_{10}, v_{11}) are all with the same pidx, so they should be tackled first. v_6 and v_{15} choose a correct direction to make a pair. Then, v_3 can not pair with v_1, so it chooses to combine with v_4. Finally, after searching locally, v_7 can not find a local neighbor to pair, but has to make a pair with its remote neighbor v_8.

In addition to minimizing both the discrepancy and cut discussed before, SPMN also tries to balance the size of partitions are the objectives in synergistic partitioning process. However, when put together, it is impossible to achieve them simultaneously. So, SPMN tries to make a compromise among them and develop a heuristic method to tackle the problems.

- First, according to the conclusion smaller edge-weight corresponds to smaller edge-cut and the pairing tendencies, SPMN proposes a modified EWM (MEWM) method to find a matching in the coarsening phase, of which the edge-weight is as large as possible. At the end of the coarsening phase, there is no impurity in any node, meaning that each node contains no more than one type of anchor nodes. Besides, a "*purity*" vector attribute and a *pidx* attribute are added to each node to represent the percentage of each kind of anchor nodes swallowed up by it and the *pidx* of the new node, respectively.
- Then, during the initial partitioning phase, SPMN treats the anchor nodes as labeled nodes and use a modified label propagation algorithm to deal with the non-anchor nodes in the coarsest network.
- At the end of the initial partitioning phase, SPMN will be able to generate balanced k partitions and to maximize the number of same kind of anchor nodes being divided into same partitions.
- Finally, SPMN projects the coarsest network back to the original network, which is the same as traditional MKP process.

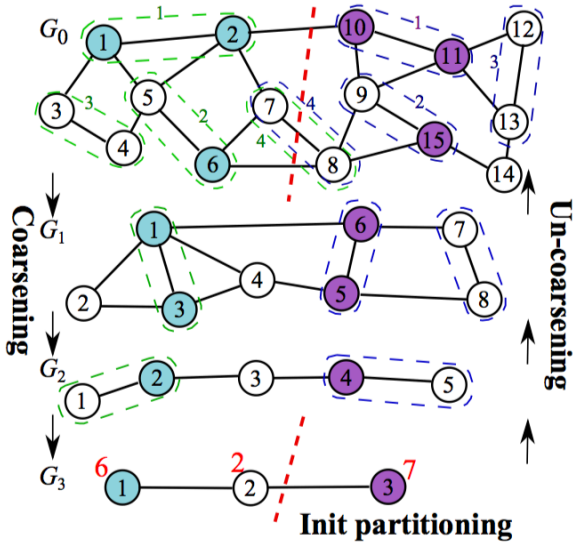


Fig. 6 An Example of Synergistic Partition Process. In coarsening phase, the networks are stored in two servers, $V_1^i = \{v^i(j)|j \leq |V^i|/2\}$ are stored on a sever and the others are on the other server. Anchor nodes are with colors, and different colors represent different partitions. Node pairs encircled by dotted chains represent the matchings. Numbers on chains mean the order of pairing.

The pseudo code of coarsening phase in synergistic partitioning process is available in Algorithm 5, which will call the $Map()$ and $Reduce()$ functions in Algorithms 6 and 7 respectively.

6 Related Works

Clustering aims at grouping similar objects in the same cluster and many different clustering methods have also been proposed. One type is the hierarchical clustering methods [14], which include agglomerative hierarchical clustering methods [8] and divisive hierarchical clustering methods [8]. Another type of clustering methods is partition-based methods, which include K-means for instances with numerical attributes [15].

In addition, clustering is also a very broad research area, which include various types of clustering problems, e.g., consensus clustering [27, 26], multi-view clustering [4, 6], multi-relational clustering [49], co-training based clustering [21], and dozens of papers have been published on these topics. Lourenco et al. [27] propose a probabilistic consensus clustering method by using evidence accumulation. Lock et al. propose a bayesian consensus clustering method in [26]. Meanwhile, Bickel et al. [4] propose to study the multi-view clustering problem, where the attributes of

objects are split into two independent subsets. Cai et al. [6] propose to apply multi-view K-Means clustering methods to big data. Yin et al. [49] propose a user-guided multi-relational clustering method, CrossClus, to performs multi-relational clustering under user's guidance. Kumar et al. propose to address the multi-view clustering problem based on a co-training setting in [21].

Clustering based community detection in online social networks is a hot research topic and many different techniques have been proposed to optimize certain measures of the results, e.g., modularity function [30], and normalized cut [38]. Malliaros et al. give a comprehensive survey of correlated techniques used to detect communities in networks in [28] and a detailed tutorial on spectral clustering has been given by Luxburg in [45]. These works are mostly studied based on homogeneous social networks.

In recent years, many community detection works have been done on heterogeneous online social networks. Zhou et al. [63] propose to do graph clustering with relational and attribute information simultaneously. Zhou et al. [64] propose a social influence based clustering method for heterogeneous information networks. Some other works have also been done on clustering with incomplete data. Sun et al. [40] propose to study the clustering problem with complete link information but incomplete attribute information. Lin et al. [25] try to detect the communities in networks with incomplete relational information but complete attribute information.

7 Summary

In this chapter, we focus on the *community detection* problem in online social networks. Community detection has been demonstrated as an important research problem especially for online social networks. Section 2 summarizes several existing community detection methods for one single homogeneous networks, which are based *node proximity*, *community modularity maximization* and *spectral clustering* respectively. These single-homogeneous network community detection methods provides the basis for addressing the problem in more complicated problem settings, e.g., *heterogeneous networks* and *multiple networks*, to be covered in Sections 3, 4 and 5. Section 3 introduces a novel problem setting based on multiple aligned heterogeneous social networks, i.e., the *emerging network community detection*, where the target network lacks sufficient information for detecting effective community structure in it. In Section 4, we talk about the *mutual community detection* of multiple aligned heterogeneous social networks, where information across these networks can be utilized for mutual community structure refinement. Finally, Section 5 introduces the *synergistic community detection* of large-scale social networks involving millions even billions of users. Three novel community detection algorithms CAD, MCD and SPMN have also been introduced in great detail in these three sections. These three proposed community detection algorithms learn the social community structures of the multiple aligned networks with the (strong/weak) supervision of anchor links, based on the assumption that anchor users tend to be

involved into relatively similar communities in different networks. Meanwhile, they also take considerations of the network properties at the same time, where each social network can maintain their characteristics as well.

References

1. C. Aggarwal, Y. Xie, and P. Yu. Gconnect: A connectivity index for massive disk-resident graphs. *VLDB Endowment*, 2009.
2. A. Arenas, L. Danon, A. Díaz-Guilera, P. M. Gleiser, and R. Guimerá. Community analysis in social networks. *The European Physical Journal B*, 2004.
3. I. Bhattacharya and L. Getoor. Relational clustering for multi-type entity resolution. In *MRDM*, 2005.
4. S. Bickel and T. Scheffer. Multi-view clustering. In *ICDM*, 2004.
5. T. Bui and C. Jones. A heuristic for reducing fill-in in sparse matrix factorization. In *PPSC*, 1993.
6. X. Cai, F. Nie, and H. Huang. Multi-view k-means clustering on big data. In *IJCAI*, 2013.
7. W. Cheng, X. Zhang, Z. Guo, Y. Wu, P. Sullivan, and W. Wang. Flexible and robust co-regularized multi-domain graph clustering. In *KDD*, 2013.
8. P. Cimiano, A. Hotho, and S. Staab. Comparing conceptual, divise and agglomerative clustering for learning taxonomies from text. In *ECAI*, 2004.
9. J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. *Communications of the ACM*, 2008.
10. M. Eslami, A. Aleyasen, R. Moghaddam, and K. Karahalios. Friend grouping algorithms for online social networks: Preference, bias, and implications. In *Social Informatics*, 2014.
11. Peter Gács and L. Lovász. Complexity of algorithms. *Lecture Notes*, 1999.
12. A. Goder and V. Filkov. Consensus Clustering Algorithms: Comparison and Refinement. In *Proceedings of the 9th Workshop on Algorithm Engineering and Experiments*, 2008.
13. M. Hasan and M. J. Zaki. A survey of link prediction in social networks. In Charu C. Aggarwal, editor, *Social Network Data Analytics*. 2011.
14. T. Hastie, R. Tibshirani, and J. Friedman. Hierarchical clustering. In *The Elements of Statistical Learning*. 2009.
15. Z. Huang. Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data Mining Knowledge Discovery*, 1998.
16. S. Jin, J. Zhang, P. Yu, S. Yang, and A. Li. Synergistic partitioning in multiple large scale social networks. In *IEEE BigData*, 2014.
17. G. Karypis and V. Kumar. Analysis of multilevel graph partitioning. In *Supercomputing*, 1995.
18. G. Karypis and V. Kumar. Parallel multilevel k-way partitioning scheme for irregular graphs. In *Supercomputing*, 1996.
19. G. Karypis and V. Kumar. Multilevel k-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed Computing*, 1998.
20. X. Kong, J. Zhang, and P. Yu. Inferring anchor links across multiple heterogeneous social networks. In *CIKM*, 2013.
21. A. Kumar and H. Daumé. A co-training approach for multi-view spectral clustering. In *ICML*, 2011.
22. J. Leskovec, K. Lang, and M. Mahoney. Empirical comparison of algorithms for network community detection. In *WWW*, 2010.
23. T. Li, C. Ding, and M. I. Jordan. Solving consensus and semi-supervised clustering problems using nonnegative matrix factorization. In *ICDM*, 2007.
24. D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *J. Am. Soc. Inf. Sci. Technol.*, 2007.

25. W. Lin, X. Kong, P. Yu, Q. Wu, Y. Jia, and C. Li. Community detection in incomplete information networks. In *WWW*, 2012.
26. E. F. Lock and D. B. Dunson. Bayesian consensus clustering. *Bioinformatics*, 2013.
27. A. Lourenço, S. R. Bulò, N. Rebagliati, A. L. N. Fred, M. A. T. Figueiredo, and M. Pelillo. Probabilistic consensus clustering using evidence accumulation. *Machine Learning*, 2013.
28. F. D. Malliaros and M. Vazirgiannis. Clustering and community detection in directed networks: A survey. *CoRR*, abs/1308.0971, abs/1308.0971, 2013.
29. M. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 2006.
30. M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 2004.
31. N. Nguyen and R. Caruana. Consensus clusterings. In *ICDM*, 2007.
32. S. Pan and Q. Yang. A survey on transfer learning. *TKDE*, 2010.
33. R. Panigrahy, M. Najork, and Y. Xie. How user behavior is related to social affinity. In *WSDM*, 2012.
34. P. Petersen. *Linear Algebra*. 2012.
35. M. Richardson and P. Domingos. Mining knowledge-sharing sites for viral marketing. In *KDD*, 2002.
36. R. Roman. Community-based recommendations to improve intranet users' productivity. Master's thesis, 2016.
37. W. Shao, J. Zhang, L. He, and P. Yu. Multi-source multi-view clustering via discrepancy penalty. In *IJCNN*, 2016.
38. J. Shi and J. Malik. Normalized cuts and image segmentation. *TPAMI*, 2000.
39. K. Shvachko, H. Kuang, S. Radia, and R. Chansler. The hadoop distributed file system. In *MSST*, 2010.
40. Y. Sun, C. Aggarwal, and J. Han. Relation strength-aware clustering of heterogeneous information networks with incomplete attributes. *VLDB*, 2012.
41. Y. Sun, J. Han, X. Yan, P. Yu, and T. Wu. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *PVLDB*, 2011.
42. Y. Sun, Y. Yu, and J. Han. Ranking-based clustering of heterogeneous information networks with star network schema. In *KDD*, 2009.
43. J. Tang, H. Gao, X. Hu, and H. Liu. Exploiting homophily effect for trust prediction. In *WSDM*, 2013.
44. M. Trusov, A. Bodapati, and R. Bucklin. Determining Influential Users in Internet Social Networks. *Journal of Marketing Research*, 2010.
45. U. von Luxburg. A tutorial on spectral clustering. *CoRR*, 2007.
46. S. Wang, Z. Zhang, and J. Li. A scalable cur matrix decomposition algorithm: Lower time complexity and tighter bound. *CoRR*, 2012.
47. X. Wang and G. Chen. Complex networks: small-world, scale-free and beyond. *IEEE Circuits and Systems Magazine*, 2003.
48. Z. Wen and W. Yin. A feasible method for optimization with orthogonality constraints. Technical report, Rice University, 2010.
49. X. Yin, J. Han, and P. Yu. Crossclus: user-guided multi-relational clustering. *Data Mining and Knowledge Discovery*, 2007.
50. Q. Zhan, J. Zhang, S. Wang, P. Yu, and J. Xie. Influence maximization across partially aligned heterogenous social networks. In *PAKDD*, 2015.
51. J. Zhang, J. Chen, S. Zhi, Y. Chang, P. Yu, and J. Han. Link prediction across aligned networks with sparse low rank matrix estimation. In *ICDE*, 2017.
52. J. Zhang, L. Cui, P. Yu, and Y. Lv. Bl-ecd: Broad learning based enterprise community detection via hierarchical structure fusion. In *CIKM*, 2017.
53. J. Zhang, X. Kong, and P. Yu. Predicting social links for new users across aligned heterogeneous social networks. In *ICDM*, 2013.
54. J. Zhang, X. Kong, and P. Yu. Transferring heterogeneous links across location-based social networks. In *WSDM*, 2014.

55. J. Zhang and P. Yu. Community detection for emerging networks. In *SDM*, 2015.
56. J. Zhang and P. Yu. Integrated anchor and social link predictions across partially aligned social networks. In *IJCAI*, 2015.
57. J. Zhang and P. Yu. Mcd: Mutual clustering across multiple social networks. In *Proceedings of IEEE BigData Congress*, 2015.
58. J. Zhang, P. Yu, and Y. Lv. Enterprise community detection. In *ICDE*, 2017.
59. J. Zhang, P. Yu, Y. Lv, and Q. Zhan. Information diffusion at workplace. In *CIKM*, 2016.
60. J. Zhang, P. Yu, and Z. Zhou. Meta-path based multi-network collective link prediction. In *KDD*, 2014.
61. Jiawei Zhang. Link prediction across heterogeneous social networks: A survey. 2014.
62. Y. Zhao, E. Levina, and J. Zhu. Community extraction for social networks. *Proceedings of the National Academy of Sciences*, 2011.
63. Y. Zhou, H. Cheng, and J. Yu. Graph clustering based on structural/attribute similarities. *VLDB*, 2009.
64. Y. Zhou and L. Liu. Social influence based clustering of heterogeneous information networks. In *KDD*, 2013.