Community Detection in Node Attributed Networks: A Late-fusion Approach

by

Chang Liu

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Statistical Machine Learning

Department of Computing Science

University of Alberta

Abstract

With the burgeoning of online social media and the deluge of information in today's "big data" era, traditional community mining that relies on the connections of the nodes no longer suffices to find communities where the attributes of these nodes play an important role. Though vast research has been done to incorporate attribute information in search of network communities, few has focused on the late-fusion approach, where two partitions of network are identified with traditional community detection and clustering algorithms respectively, and are later combined to produce the final communities. We propose a new late-fusion method that assimilates two sources of information by creating an integrated graph whose edges represent the agreement of communities coming from the two partitions. We design a new technique to cope with networks with binary or categorical attributes in a way that clusters reflecting node similarities are found by a community detection algorithm on a virtual graph. We introduce a weighting parameter to allow for leveraging the strength between node connections and attributes. We experimentally demonstrate the performance of our method on various synthetic and real networks. We show that our late-fusion method comes as a flexible, accurate and efficient solution to the problem of community detection in attributed networks.

Dedicated to

My parents,

And

Hao, my love.

Acknowledgements

I would like to express my sincere thankfulness to my two supervisors, Osmar Zaïane and Christine Lageron, who have provided me with tremendous guidance and help throughout my master. Without them, this work could not have been completed. I have learned so much from the two of them both as a researcher and a human being.

I am also thankful for Shiva Zamani Gharaghooshi, who is not only an amazing colleague but also a great friend. Special thanks to two of my dear friends at University of Alberta, Fushan Li and Wenting Zhang, with whom I had the best time outside of research.

I would like to thank my parents for their unconditional love and support. Finally, I would like to thank my boyfriend, Hao Li, who accompanies me when I am going through hardships, and is always there cheering for me when I make it.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Thesis Motivation

We are living in a world filled with complex systems that can be represented as networks. For example, we have biological systems such as protein-protein interactions [18], [79], [83], where proteins act as nodes and the interactions between them are the edges in a network; In online social networking sites such as Facebook or LinkedIn, every member can be seen as a single node and users are linked by their friendships or professional connections; Another notable example is the World Wide Web (WWW) [30], [32], where each web page can be taken as a node, and the hyperlinks that redirect to other web pages are directed edges of the WWW network.

All of the complex mentioned above share properties such as power law degree distribution [1], [2], high clustering coefficient, assortative mixing [66] and so forth. In this thesis we focus on the *community structure* [37] of real-world networks. These topological properties reflect the real world systems and do not appear in simple networks such as a lattice or a random graph (see Figure 1.1). The object of our study is this kind of complex networks. Figure 1.2 gives an example of a complex network.

Among all of the features of complex networks, we are most interested in the community structure of these networks. Loosely speaking, the notion of community describes a subset of nodes in a network that have denser connec-

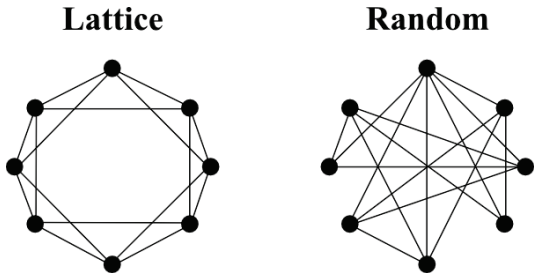


Figure 1.1: Examples of lattice and random graphs.

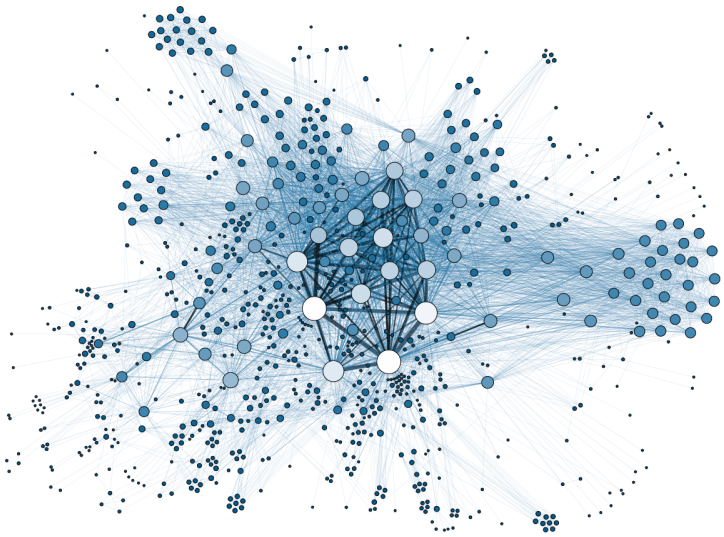
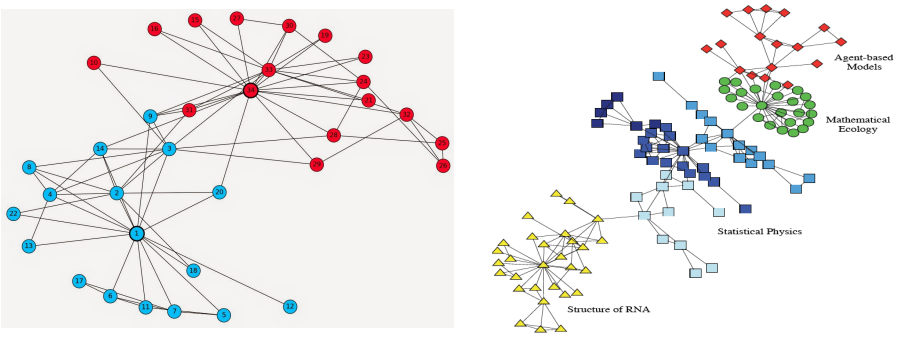


Figure 1.2: Example of a complex network.

tions with each other than nodes outside of the subset. In real networks, the communities can be determined by a wide variety of possible group organizations, such as affiliation, friendship, interest, functional modules, etc. Figure shows two examples of community structure in networks. We will further discuss the definitions of communities in a network in Section 2.1.1.

Identifying the communities of a network can help us better understand the structure of the network. Furthermore, it can lead to many substantial applications. First, community detection can be beneficial to medical research. Consider for example protein-protein interaction (PPI) networks. The communities in PPI networks often correspond to proteins that have the same or similar functions, and these proteins are expected to be involved in the same processes. Detecting communities in PPI networks is crucial to the un-



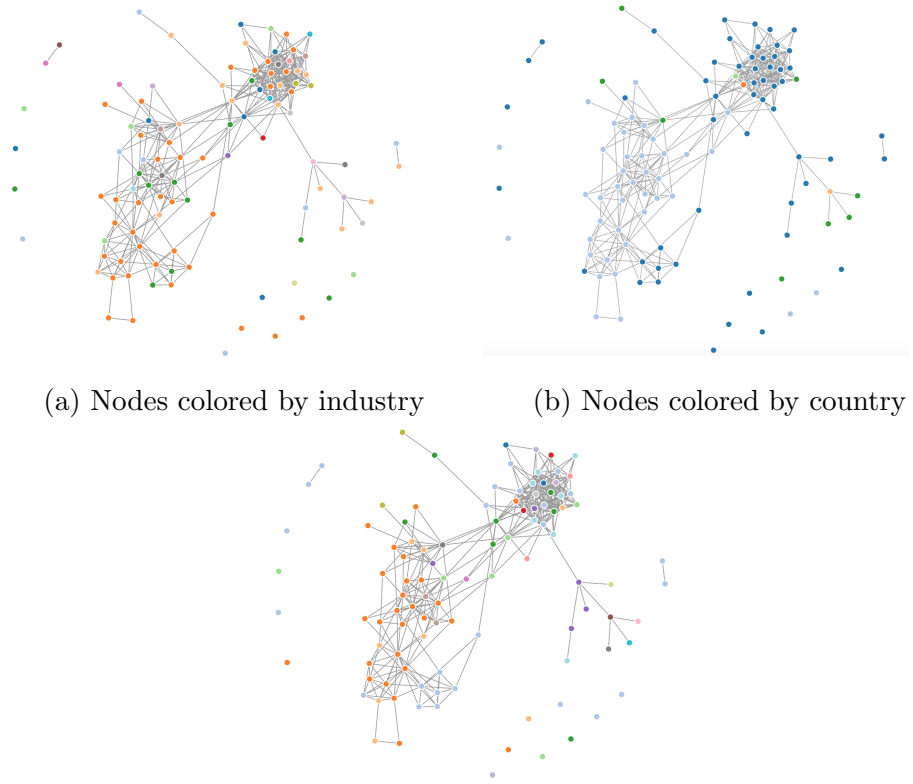
(a) (b)

Figure 1.3: Community structure in social networks. (1.3a) Zachary's karate club, a standard benchmark in community detection. (1.3b) Collaboration network between scientists working at the Santa Fe Institute.

derstanding of the processes that could be associated to cancer or metastasis. Second, identifying communities can help improve the web services and searching results. In WWW networks, web pages related to similar topics tend to cluster into communities. Identifying these artificial clusters can be a complement to the PageRank algorithm and lead to more accurate and reasonable Google ranking results [19]. Third, network communities can be utilized for resource optimization. For instance, clusters of large graphs can be used to efficiently store the graph data to handle navigational queries [3] and generate compact routing tables [84]. We can also mention other applications of community detection, such as node classification, online advertising and recommendation, link prediction, etc.

There have already been a great number of methods that find communities of a network. We provide an overview of these methods in Chapter 2. These traditional methods focus on the inhomogeneous distribution of edges and detect communities based on the linkage of nodes, i.e., the structural information of networks. However, today's networks bear with massive information more than just the connections between nodes. Most notably we draw our attention to the attributes attached to each node. The attributes of a node can take in a wide range of forms. It can be the content of a web page, the profile of an

online forum member, or the functionality of a protein. Figure 1.4 shows a visualization of my own LinkedIn connections network, where nodes are colored by different attributes. We can see that when looking at different colors, the corresponding node attributes exhibit different degrees of homogeneity, and hence indicate different perspectives of identifying the community structure. For instance, according to the connections, the connected part of the network in Figure 1.4 can be partitioned into two modules, the larger one on the lower left and the smaller one on the upper right. However by looking at the countries of these nodes (Figure 1.4b), we notice that a small subset of nodes located at the lower left of the the larger module have a different attribute from the module that it belongs by connections. Figure 1.4c consolidate that these people are not only from the same country, but also the exact same location. Hence we may consider putting these nodes in a separate community.



(a) Nodes colored by industry

(b) Nodes colored by country

(c) Nodes colored by location

Figure 1.4: Visualization of my LinkedIn connection network.

The presence of this additional attribute information has brought both opportunities and challenges to the community mining task. First of all, we believe that considering both connections and attributes of nodes in the detection of network communities can lead us to methods that identify communities more accurately. Researchers have found a reciprocal, co-dependent relation between the social influence represented by edges and the attribute homogeneity of nodes [45], [47], [48]. This relation states that nodes that are linked together tend to exhibit similar attributes, and at the same time, nodes are also more likely to have connections with similar ones than dissimilar ones. More importantly, utilizing attribute information has great potential to industrial impact. For example, consider again the social networking applications, Facebook and LinkedIn. Clustering the users considering both their social relationships and personal profiles is particularly useful for applications such as user-targeted online advertising, job positions recommendation, candidate hunting, and so on.

On the other hand, we are also faced with a few challenges when we want to take advantage of attribute information. Though the homophily in social networks has been extensively studied by social scientists [50], [62], heterophily (disassortativity) can also occur in networks, where links between nodes have dissimilar, or even opposite attributes. Example of such is friends with different political orientations or social status. Our analysis throughout this thesis will only focus on the homophily attributes of networks. Another challenge is the fact that not all attributes are useful in detecting a community structure. How to properly select and exploit from an abundant amount of attribute information is another hard problem has yet to be solved. Last but not least, the lack of network datasets with ground-truth communities makes the evaluation of community mining algorithms a very challenging task. For attributed networks, the situation is even worse. We will address several commonly used evaluation measures in Chapter 2

For convenience, from now on we refer to networks with node attributes as *attributed networks*. The problem of community detection in attributed networks has been in the spotlight of the field of data mining, as we will see

in Chapter 2 where we review the existing methods that have already been proposed to tackle this problem. Most of the attributed community detection algorithms in the literature follow an "early-fusion" manner, meaning that attributes are exploited before the final community structure is identified. Very few methods have used a "late-fusion" approach, which detects two sets of communities, one based on the structure and the other based on the attribute, independently. Later on, a merging strategy is applied to find the ultimate set of communities that integrates the two sets of communities that have been previously identified. We will elaborate on our proposed late-fusion method in Chapter 3.

With all being said, we are fully motivated to devote more efforts to the problem of community detection in attributed networks.

1.2 Thesis Statements

Our work in this thesis focuses on the methodology of community detection on attributed networks. It addresses the following statements:

- With the burgeoning of online social networks, traditional community detection methods based solely on node linkages are no longer able to accurately find online social communities with high accuracy, since node attributes play an increasingly significant factor in the formation of social communities. Methods that take the attribute information into consideration can greatly improve the quality of the detected communities.

- While it has been rarely explored before, the late-fusion approach provides a powerful addition to the toolkit of attributed community detection. It has the following advantages when compared to other approaches to the same problem:

  - Simplicity: the frame of our late fusion approach is very easy to understand and implement. It does not require arduous steps of pre-processing and/or post-processing.

  - Flexibility: since late fusion is a realization of combining community detection based on link structure with clustering based on attribute similarity, it can take advantage of the existing methods on both sides. Moreover, we also introduce a weighting parameter to leverage the influence of structural information against attribute information in our framework. Hence depending on the nature of the network under study, people can accordingly choose suitable algorithms and/or the best weight.
  - Accuracy: despite being simple and flexible, the late-fusion method that we propose in this thesis exhibits great performance on a wide range of real and synthetic networks when evaluated against available ground-truth communities.
  - Scalability: unlike many other attributed community detection algorithms that have a major drawback of being time-consuming, our approach successfully circumvent this issue by utilizing matrix operation which is scalable and easily parallelizeable by MapReduce [29].

- Our late-fusion method identifies network communities with the following two properties:
  - Structure: nodes belonging to the same community have more dense connections than nodes across different communities.
  - Attribute: nodes belonging to the same community have more homogeneous attributes than nodes across different communities.

1.3 Thesis Contributions

We propose a novel community detection method applied to attributed networks that follows a late-fusion approach. A great many algorithms for attributed community detection have been devised over the years (see Section 2.3), however they inevitably require careful and often complex design to incor-

porate node attributes into the process of community detection. Our method, on the other hand, chooses to take advantage of the abundant existing community detection algorithms based on node connections and also the clustering algorithms based on node attributes.

Our method starts with identifying two sets of communities (i.e., two partitionings over a network), denoted by $\mathcal{P}_s$ and $\mathcal{P}_a$ where the subscript a stands for "attribute" and the subscript s stands for "structure". $\mathcal{P}_s$ and $\mathcal{P}_a$ are obtained by employing a community detection algorithm and a clustering algorithm to utilize the structure and the attribute respectively. Based on the community assignment of the two partitionings, we can compute two corresponding matrices D_s and D_a with binary entries, where $d_{ij} = 1$ if nodes i and j are assigned to the same community, and otherwise $d_{ij} = 0$. The "fusion" part happens in the next step where we sum up D_s and D_a to form an integrated adjacency matrix D. By careful design, we can take D as an adjacency matrix, based on which an integrated graph is built whose edges represent the agreement between the two partitions. Now we can employ community detection algorithms such as Louvain [10] or SIWO [36] to find the set of communities on the integrated graph, which will be the final detected communities that have the information of both structure and attribute combined together.

Networks exhibit various types of attributes. The attributes can be numerical (such as people's age, salary, etc.), categorical (for instance, people's occupation and education) or binary (whether or not a person has been on a sport team, etc). Noticing that we can easily translate a categorical attribute to a binary one by flatting a multi-label variable to a one-hot vector, we put binary and categorical attributes in the same group and devise different approaches to treat numerical and binary/categorical attributes separately. As for numerical attributes, we can easily apply clustering algorithms such as k-means [57] or DBSCAN [31] to get the communities (i.e., clusters) based on the node similarity. When it comes to binary/categorical attributes, we build another virtual graph. On the virtual graph, the existence of an edge between a given pair of nodes shows that these two nodes are similar enough in terms of attribute.Then the communities that reflect attribute similarity

can be retrieved by applying classic community detection algorithms on this virtual graph.

In the fusion stage, the integrated adjacency matrix is computed by the equation $D = \alpha D_s + (1 - \alpha)D_a$, where α is a real value range from 0 to 1 to leverage the influence between the attribute and the structure. This hyper-parameter gives users the freedom to put different weights on the two sources of information according to their preferences or any prior knowledge about the network.

1.4 Thesis Organizations

The contents of this thesis is organized as follows: In Chapter 2, we review related work and divide our reviews into three parts. First, we revisit traditional community detection algorithms that only consider node connections (Section 2.1). Second, we review several representative unsupervised clustering algorithms that consider only node attributes (Section 2.2). At last, we introduce a typology of community detection methods that take both structure and attributes into account (Section 2.3). We propose our late-fusion method in Chapter 3, where we introduce two late-fusion algorithms that are devoted to numeric and binary attributes respectively. Then in Chapter 4, we provide a thorough empirical study on a series of real and synthetic network datasets. By comparing the performance of our method against the others, we show the strength and weakness of each method, and prove our earlier statements. At last, we summarize our work and point out several future directions in Chapter 5.

Chapter 2

Background and Related Work

To figure out how to combine structure and attribute together, we need to first understand how they have been used in separation. Therefore, in this section, we review two aspects of work in the literature: classic community detection methods based on linkage of nodes (Section 2.1), and classic unsupervised clustering based on feature vectors of data points (Section 2.2). Then we review previous methods that find communities using both structure and attribute information, in Section 2.3. At the end of this chapter (Section 2.4), we discuss different evaluation metrics that measure the quality of communities.

2.1 Community Detection Based on Node Connections

2.1.1 Notations

The first important question that needs to be answered is the definition of community and community detection. As a matter of fact, the definition of community still evolves depending on the specific problem one tries to solve or application one has in mind. To better formulate the problem, let us introduce several useful notations that will be used throughout the manuscript. In this thesis we use capital letter G to denote a network, or interchangeably, a graph. Following the convention, a network G is written as $G = (V, E)$ where V refers

to the set of nodes belonging to G, and E stands for the set of edges that link the nodes. $|V|$ and $|E|$ are used to denote the number of nodes and edges. We use calligraphy letter $\mathcal{P}$ to represent a partitioning over the node set V, and it can be expressed as $\mathcal{P} = \{C_1, ..., C_k\}$ where each $C_i \in \mathcal{P}$ forms a community of G and k denotes the number of communities.

2.1.2 Definitions of Network Communities

The definitions of communities have been evolving over the time. Recently, Fortunato and Hric [35] summarized a set of variables that people often use to define network communities. These variables are categorized into three classes, where the first class comprises measures based on internal connectedness, the second class includes measures based on external connectedness, and the third class contains measures that combine internal and external connectedness. As for internal connectedness, *interal degree* k_C^{int} is defined as the sum of internal degrees of nodes belonging to community C. Then the *internal edge density* δ_C^{int} is given by the ratio between the number of internal edges of C and the number of all possible internal edges, which is expressed as $\delta_C^{int} = \frac{k_C^{int}}{n_C(n_C-1)}$, where n_C is the number of nodes in C. As for external measures, we have the counterpart of k_C^{int}, k_C^{ext}, which is the sum of external degrees of nodes belonging to C. The *external edge density* δ_C^{ext} is the ratio between the number of external edges of C and the number of all possible external edges: $\delta_C^{ext} = \frac{k_C^{ext}}{n_C(|V|-n_C)}$. Finally, *conductance* is the measure combining internal and external measures by the ratio $\frac{k_C^{ext}}{k_C^{int}+k_C^{ext}}$.

From a classic point of view, definitions of communities seek to either maximize the internal connectedness or minimize the external connectedness. For example, *clique* is a popular concept that requires every node is connected to all other nodes in the same community [60]. Despite a clique has the largest possible internal connectedness, communities usually are not complete graphs, hence cliques cannot be considered a good candidate for a community definition. To relax the strict requirement of cliques, we have alternative definitions based on reachability and diameter, which states that there should relatively

be short paths between two members from the same community. *n-clique*[59], *n-clans* and *c-clubs*[64] are three variants falling into this class. A proper definition of network communities should consider both internal and external connections of nodes, i.e., the number of internal edges should be larger than the number of external edges. Inspired by this idea, lots of evaluation measures have been proposed. Here we briefly introduce a measure called *modularity*, proposed by Newman and Girvan [69], and is deemed as one of the most effective metric to define and evaluate community structure. Modularity compares the node degree distribution of each community with the expected degree distribution. This measure is based on a null model proposed by Newman and Girvan, which is a randomized version of the original graph where edges are rewired at random, but the expected degree of each node is kept the same. For example, given a network $G = (V, E)$, the expected number of edges between two nodes i, j with degree d_i, d_j is $\frac{d_i d_j}{2|E|}$. The concepts of modularity and its applications will be further discussed in later sections.

There are also other types of definitions for network communities. We can mention the structural equivalence [58] which states that two nodes belong to the same community if they have identical neighbors, and the automorphic equivalence [13] which assigns the same community to nodes that occupy indistinguishable structural locations in a network. The problem with equivalence is that most empirically observed networks hardly have any nodes that are structurally or automorphically equivalent. In this thesis we formulate our community detection problem as follows: Given a network $G = (V, E)$, we aim at finding a partitioning $\mathcal{P} = \{C_1, ..., C_k\}$ of V into k subgroups such that for each group C, (1) nodes belonging to the same community are densely connected, and (2) nodes from distinct communities are loosely connected. By definition, the partitioning $\mathcal{P}$ must also satisfy the following three conditions:

1. $\cup_{i \in \{1,2,...,k\}} C_i = V$
2. $C_i \cap C_j = \emptyset, \forall i, j \in \{1, 2, ..., k\}, i \neq j$
3. $C_i \neq \emptyset, \forall i \in \{1, 2, ..., k\}$.

There are many methods that are able to find such partitionings of a network. We will devote the following several sections to these algorithms. It should be noted that in real-world networks, communities can have overlaps, since a node may belong to more than one group. This is most commonly seen in social networks. For example, an individual can belong to a community defined by the university goes to, and a different community defined her high school [51]. We will not emphasize on approaches that find overlapping communities, but interested readers can refer to [94] to learn more about overlapping community detection in networks.

2.1.3 Graph Partitioning

Algorithms that fall into this category aim at dividing the original graph G into k groups, so that the number of edges (which is called *cut size*, denoted by R) running between communities is minimized. We introduce two algorithms in this section: the Kernighan-Lin algorithm [44] and the spectral bisection method [7].

The Kernighan-Lin algorithm indirectly minimizes the cut size by maximizing a value Q, which represents the difference between the number of edges within the groups and the number of edges between the groups. It starts by partitioning the graph into two groups S_1 and S_2 with predefined sizes. Then the subsets of S_1 and S_2 that have the same number of nodes are swaped between S_1 and S_2. By conducting different swaps repeatedly, we can find a partitioning $\mathcal{P}$ of V that gives the largest value of Q. This approach can be extended to find a partitioning of V that contains more than two groups [87].

The spectral bisection method uses the technique of eigen-decomposition of the Laplacian matrix to represent and minimize cut size R. Given a partitioning $\mathcal{P}$ of graph nodes V that splits all the nodes of graph G into two groups, S_1 and S_2. The spectral bisection method constructs a vector $\mathbf{s}$ of length $|V|$, whose value $v = s_i = 1$ if node i belongs to group S_1, and $s_i = -1$ when node i belongs to group S_2. Therefore the cut size R can be expressed

by vector $\mathbf{s}$ and the Laplacian matrix $\mathbf{L}$, written as:

$$R = \frac{1}{4}\mathbf{s}^T\mathbf{L}\mathbf{s} \tag{2.1}$$

Note $\mathbf{s}$ can be written as a linear combinations of the eigenvectors of $\mathbf{L}$, i.e., $\mathbf{s} = \sum_i a_i\mathbf{v}_i$, where $\mathbf{v}_i, i = 1, ..., n$ are the eigenvectors. Hence

$$R = \sum_i a_i^2\lambda_i \tag{2.2}$$

where λ_i is the eigenvalue corresponding to eigenvector $\mathbf{v}_i$. It turns out that the minimum of R can be approximated by λ_2 if λ_2 is close enough to zero. The group assignment of nodes in graph G can be set according to the sign of $\mathbf{v}_2$.

2.1.4 Agglomerative Hierarchical Methods

In most cases of community detection, we do not know the exact number of communities in the graph. Hierarchical methods provide an alternative class of solutions to the problem. The implementations of hierarchical algorithms can be further classified in two categories: agglomerative algorithms and divisive algorithms. We devote this section to agglomerative methods. We will come to divisive methods in the next section.

Agglomerative methods follow a bottom-up process in which nodes are initially treated as singleton community. Then communities that have sufficient similarity are iteratively merged until an objective function is optimized, or a preset number of communities have been found, or there is no more merging that can be done. We further address three kinds of methods that follow this approach: methods based on modularity, methods based on dynamics and the SIWO approach.

Methods Based on Modularity

Modularity is a quality function firstly proposed by Newman and Girvan [69]. It measures the goodness of community partitioning based on the comparison of degree distribution in each community with the expected distribution in

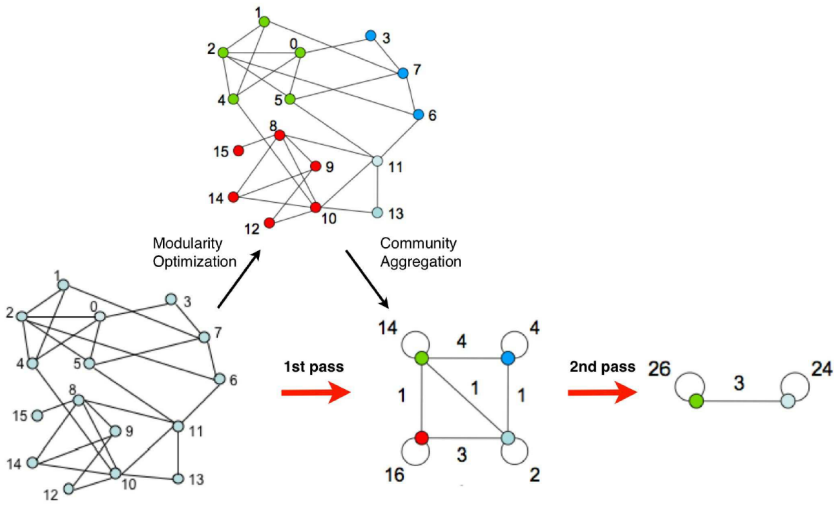


Figure 2.1: Visualization of the steps of the Louvain algorithm.

a null model. As stated earlier in Section 2.1.2, the null model is a random re-organization of the edges that follows the node degree distribution of the original network. The classic form of modularity is expressed as:

$$Q = \frac{1}{2|E|} \sum_{ij \in V} (A_{ij} - \frac{d_i d_j}{2|E|}) \delta(C_i, C_j), \tag{2.3}$$

where A_{ij} is the entry of the adjacency matrix, d_i and d_j are the degrees of nodes i and j and $\frac{d_i d_j}{2|E|}$ corresponds to the expected number of edges joining nodes i and j, δ is the Kronecker function that takes the value of 1 if $C_i = C_j$ and 0 otherwise, and C_i and C_j are the communities that i and j belong to.

Since the optimization of modularity is NP-hard [14], a wide variety of approaches have been proposed to find decent approximations of the maximum modularity, among which the Louvain method [10] is one of the most popular and successful methods. The algorithm of Louvain starts by assigning a different community to each node of the network. Then for each node i, move i to the community which its neighbor j belongs to, to achieve the maximum modularity gain. The second phase is creating a weighted super-network, in which the nodes are the communities obtained from the first phase. This two-phase process is conducted repeatedly until there is no positive gain of modularity. Figure 2.1 presents a graphical illustration of the Louvain algorithm.

The most notable advantage of Louvain method is that experimentally it has been showed grow linearly with the network size. However, it fails to solve an inherent defect in modularity, the *resolution limit*. Fortunato and Barthelemy [34] state that modularity optimization may fail to identify modules smaller than a scale which depends on the total size of the network and on the degree of interconnectedness of the modules. Nowadays, community detection algorithms that are improvements upon the Louvain method are still being actively studied by researchers.

Methods Based on Dynamics

Dynamic process on a graph is also useful for detecting communities. Let us consider a random walk on a graph G as a dynamic process. At each time step, a walker is at a node and chooses to move to the next node randomly and uniformly from its neighbors. Therefore a random walker tends to "get trapped" into densely connected parts of the network, which reveals the community structure. In this section we introduce two representative methods of this class, Walktrap [74] and Infomap [80].

Walktrap defines the similarity distance between two nodes i and j by the probability that a random walker moves from i to j in a fixed number of steps t. Having this distance well-defined, the algorithm also starts by assigning a different community for each individual node. Then the Ward hierarchical clustering algorithm [91] is applied where it repeatedly chooses two communities to merge based on their similarity distance, and updates the distances between the new communities. Walktrap has a computational complexity at $O(|V||E|H)$ with H being the height of the hierarchical dendrogram, so it does not scale well on large networks.

Infomap is an extension of the Louvain algorithm to minimize a mapping equation developed by Rosvall and Bergstrom [80], who use the random walk on a network as a proxy for information flows in real systems. They construct a mapping equation that connects the problem of finding the community structure in a network with the problem of encoding the network structure with as

little entropy loss as possible. Comparing with modularity optimization methods, Infomap runs at $O(|E|)$ and is suitable to weighted and directed graphs In particular, it naturally fits in network data where links represent patterns of movement among nodes.

The SIWO Approach

The aforementioned two approaches have major drawbacks. Fortunato and Barthelemy [34] showed that modularity suffers from the resolution limit, meaning that by optimizing modularity, communities that are smaller than a scale cannot be detected. The field-of-view limit [82] is in contrast to the resolution limit which results in overpartitioning communities with a large diameter. Schaub *et al.* [82] showed that both modularity and the map equation are affected by the field-of-view limit.

SIWO, proposed by Gharaghooshi[36], is short for **S**trong **I**n, **W**eak **O**ut. SIWO is a new objective to replace the modularity by using a novel concept of edge strength that is defined according to the number of shared neighbors between a given pair of nodes. Consider an edge $e_{ij} = (i, j)$ that connects nodes i and j. Let S_{ij} denote the number of shared neighbors between i and j. Define $S_i^{max} = \max_{j:(i,j)\in E} S_{ij}$, which is the maximum value of S_{ij} with fixed node i. In the same way we can define S_j^{max}. Now the strength w_{ij} of edge e_{ij} is given by

$$w_{ij} = \begin{cases} S_{ij}\frac{2}{S_i^{max}+1} + \frac{1}{S_j^{max}+1} - 1, & \text{if } CC(i) \geq CC(j) \\ S_{ij}\frac{2}{S_j^{max}+1} + \frac{1}{S_i^{max}+1} - 1, & \text{otherwise} \end{cases} \tag{2.4}$$

where $CC(\cdot)$ refers to the local clustering coefficient [92].

Having the edge strength well defined, the algorithm follows the same optimization manner as in Louvain with the modularity replaced by SIWO measure:

$$\sum_{i,j\in V} \frac{w_{ij}\delta(C_i, C_j)}{2} \tag{2.5}$$

with $\delta(\cdot)$ being the Kronecker function which also appears in Equation 2.3.

2.1.5 Divisive Hierarchical Methods

Contrary to agglomerative methods, divisive methods follow a top-down process. We introduce two representative algorithms, proposed by Girvan and Newman [37], and Radicchi *et al.* [77]. The shared idea of these two algorithms is that they define an edge measure to identify edges that are most likely to run across communities.

Newman and Girvan proposed edge betweeness, which is the number of shortest path that pass through a given edge e. Therefore edges with large values are more likely to be the"bridge" of communities. The algorithm iteratively removes edges with the largest value of betweeness, which will divide the original graph into smaller communities. Since the concept of edge betweeness needs to look up all possible node pairs, it is computationally expensive, running at $O(|V||E|^2)$. Different from [37], Radicchi *et al.* only considered local measure of edges and proposed edge-clustering coefficient, defined as the number of triangles to which a given edge belongs, divided by the number of triangles that might potentially include it. Following the same procedure as [37], the algorithms runs at $O(|E|^2)$.

2.1.6 Spectral Clustering

Spectral clustering [90] can be used to cluster data points based on their feature vectors. Since the application of spectral clustering requires to construct a similarity graph, spectral clustering is naturally extended to graph objects and has become a very popular algorithm to detect communities in networks.

The core idea of spectral clustering is to find the eigenvalues (also known as *spectrum*) of the Laplacian matrix, which is computed as

$$L = diag(d) - W \tag{2.6}$$

with $diag(d)$ being a diagonal matrix where the entities are the degree of nodes of G. W is the (weighted) adjacency matrix. Let $\Lambda \in \mathcal{R}^{|V|\times p}$ denote the matrix that contains the eigenvectors $\mathbf{e}_1, ..., \mathbf{e}_p$ of L as the columns. Then every node of the original graph G is now embedded to a p-dimensional feature space

and represented by $y_i \in \mathcal{R}^p (i = 1, .., |V|)$ corresponding to the i-th row of Λ. Next, k-means algorithms [61] can be applied to cluster the nodes into k communities.

Since spectral clustering depends on a clustering algorithm such as k-means, it usually requires prior knowledge like the number of communities. Moreover, in real networks that are very sparse, obtaining the eigenvectors of the Laplacian matrix becomes a hard problem [35].

2.1.7 Methods Based on Statistical Inference

Statistical inference is a powerful tool in community detection. The standard methods aim to fit the data into a model to probabilistically depict the relations between nodes. For example, Bayesian inference [93] presumes a generative statistical model P with parameter set Θ, and tries to find the Θ that maximizes the likelihood that the observed data is produced by the statistical model given Θ, $P(D|\Theta)$. This can be represented by a probabilistic graphical model, where the communities of the observed data are the latent variables to be inferred.

Another popular generative method of this kind is called Stochastic Blockmodels [39]. Given a graph $G = (V, E)$, the community assignment of the nodes is indicated by a set of labels $\{C\}$, where $C_i = 1, 2, ..., k$ is the community label of node i. The corresponding blockmodel is expressed by a $k \times k$ block matrix $\mathbf{B}$: $B_{C_i C_j} = 1$ if edges between classes C_i and C_j is allowed, otherwise it is zero. The aim is to find the set of communities $\{C\}$ as well as the matrix $\mathbf{B}$ that best fits the adjacency matrix of the graph G.

Our review of classic community detection algorithms is by no means exhaustive, since there have been numerous work devoted to this area. We recommend interested readers go to more extensive literature review of community detection such as [33] and [35].

2.2 Clustering Based on Node Attributes

Unlike traditional community mining algorithms which group network nodes based on their connections, clustering algorithms aim at grouping a set of data samples according the similarity between their features. The features of the samples are embedded into a N-dimensional Euclidean space and hence the similarity between them is often measured in terms of their distances. The goal of clustering is to group the data samples into clusters so that (1) samples within each cluster are similar to each other, and (2) samples from different clusters are dissimilar. To fit network data in the context of clustering, we regard the nodes as the samples, the attributes associated with every node as features of each sample. In this section, we introduce a few commonly used clustering algorithms that are based on the similarity of node attributes.

2.2.1 Agglomerative Hierarchical Methods

In Section 2.1.4 we presented some agglomerative hierarchical methods such as Louvain and SIWO. The gist of Louvain and SIWO is to optimize an objective function which encourages the set of densely connected nodes to form into a community. Here we introduce another class of hierarchical clustering methods which focus on the similarity between nodes rather than their connections. A representative is called *linkage methods* [43].

The linkage methods can be summarized as the following four steps:

1. Start with $|V|$ clusters where each node is treated as a singleton cluster: $C_i, i = 1, \cdots |V|$. Compute a $|V| \times |V|$ symmetric distance matrix D.

2. Search the distance matrix to identify the closest pair of clusters, C_i and C_j. Denote the distance between C_i and C_j by d_{ij}

3. Merge C_i and C_j to form a new cluster C_{ij}. Update the distance matrix according to a pre-chosen aggregation measure.

4. Repeat Step 2 and 3 until (a) a preset number of clusters k has been reached, or (b) all nodes are in a single large cluster.

Linkage methods differ in the measure of distance between clusters, i.e., the computation of distance matrix D. The three most commonly used linkage methods are: single linkage, where the minimum distance of any pair of nodes coming from two clusters is defined as the distance between that two clusters; complete linkage, which takes the maximum distance between two nodes from two clusters as the distance between that two clusters; and average linkage, which is similar in spirit of single linkage and complete linkage, but instead uses the average distance between all pairs of nodes coming from two different clusters.

2.2.2 Centroid-based Clustering

Centroid-based clustering is sometimes also referred to as partitional clustering, which bears the same idea as graph partitioning described in Section 2.1.3 that divides data samples into a predefined number of k non-overlapping clusters. The clustering process is performed by minimizing a cost function based on the distance of each sample to its clustering center, called centroid. The most popular algorithm of this approach is k-means clustering, proposed by MacQueen *et al.* [61]. K-means clustering aims to minimize the total intra-cluster distance, defined as

$$\sum_{i=1}^{k} \sum_{\mathbf{x}_j \in C_i} \|\mathbf{x}_j - \mathbf{c}_i\|^2 \tag{2.7}$$

where $\mathbf{x}_j$ is the feature vector of a sample j, C_i is the cluster to which j belongs, and $\mathbf{c}_i$ is the feature vector of the centroid of C_i.

Looking for the global optimum of 2.7 is an NP-hard problem. But we can approximate the solution with fast convergence by Lloyd algorithm [57]. The stpes are described as follows:

1. Assign k centroids for the data samples, usually at random or based on some prior knowledge.

2. Proceed through each sample to assign it to the cluster whose centroid is the nearest. Recalcualte the centroids after every sample has been

assigned to a cluster.

3. Repeat step 2 until the centroids do not change any more.

There are a couple of restrictions when using k-means. First, the k-means method requires prior knowledge of the number of real communities, which is not always available in practice. Second, to circumvent the local minimum resulted from Lloyd's algorithm, it is recommended that we perform multiple runs of the algorithm with random centroid initializations. Another caveat of using k-means is that it assumes that the clusters are convex shaped.

2.2.3 Density-based Clustering

Density-based clustering algorithms identify clusters as areas of high density separated by areas of low density. Therefore, as opposed to k-means, density-based methods are able to detect clusters with any shape. The most representative density-based clustering algorithm is DBSCAN [31].

There are two important parameters of the DBSCAN algorithm, the minimum number of samples m and the distance threshold ϵ. These two parameters altogether determine when a set of data points should be regarded as dense. First, a core sample is defined as a sampled data point such that there exists at least m other data points within a distance of ϵ. These m data points are defined as the neighbors of the core sample. Then the cluster is determined by recursively starting with a core sample, then adding all of its neighbors to the set of core samples, identifying the neighbors of the new core samples, and so on. Any node that is not a core sample is regarded as an outlier.

2.3 Community Detection Methods on Attributed Networks

The methods discussed Section 2.1 and 2.2 only consider a single source of information, either node connections or node similarities based on attributes.

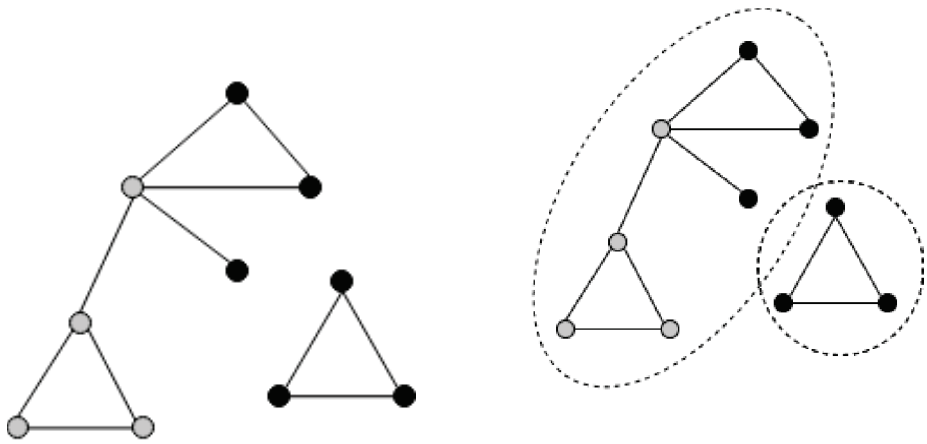
We would like to expand our review of community detection literature to the approaches that take both sources of information into account.

The goal of community detection in attributed network can be formalized as: Given an attributed graph $G = (V, E, A)$, where V is the set of nodes, E denotes the set of edges, and A represents the set of attribute vectors, the task consists of building a partitioning $\mathcal{P} = \{C_1, C_2, ..., C_k\}$ of V in k communities such that:

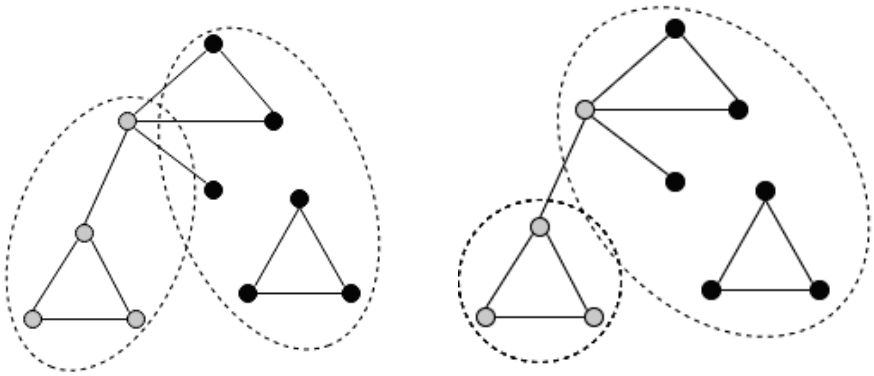
- nodes in the same community are *densely connected* and *similar in terms of attributes*;
- nodes from different communities are *loosely connected* and *dissimilar in attributes*.

Let us use a simple example to better illustrate the idea of community detection in attributed graph (Figure 2.2). Figure 2.2a presents a graph whose nodes display two types of attributes, which we use grey and black colors to represent. If we employ traditional community detection methods based on the linkage, we would get a partitioning as shown in Figure 2.2b. On the other hand, if we only consider the attributes, it will lead us to results shown in Figure 2.2c. If we consider both the attributes and the connections, probably the result would look like Figure 2.2d. This example tells us that by looking at different information and metrics, we may get different partitionings even on the same network.

There is abundant work focusing on community detection in attributed networks, with different ideologies and applications. According to the approach where node connections and attributes are exploited, we categorize these methods into the following seven classes: methods based on optimization (Section 2.3.1), methods based on unifying edge weights (Section 2.3.2), methods graph augmentation (Section 2.3.3), methods based on core expansion (Section 2.3.4), methods based on statistical inference (Section 2.3.5), methods based on embedding (Section 2.3.6), and methods based on late fusion (Section 2.3.7).



(a) Original graph (b) Structure-based



(c) Attribute-based (d) Structure and attribute

Figure 2.2: A network example with categorical attribute

2.3.1 Methods Based on Optimization

Methods that fall into this group follow the approach where node connections and similarities are fused into an objective function. By optimizing this objective function, one can find a network partitioning that integrates both sources of information. For example, PICS [4] is a parameter-free algorithm that encodes a connectivity matrix A and a binary feature matrix F into a cost function. Many other methods introduce the concept of node similarity into the original function of modularity. For instance, Cruz *et al.* [24] include an entropy-based data clustering algorithm in the optimization of modualrity.

They aim at finding the partitioning with low entropy and high modularity. Dang and Viennet [26] follow the similar idea where a term representing the node similarity: $S(i, j)$ is added to the modularity function. Then a weighting parameter $\alpha \in [0, 1]$ is used to decide the contribution of structural and attribute information. Asim *et al.* [5] combine the gain in the modularity with multiple common users' attributes to detect communities in the network. In particular, we would like to detail an algorithm following this approach called I-Louvain, proposed by Combe *et al.* [22].

Let us first introduce the *inertia based modularity*: Given a partitioning $\mathcal{P} = \{C_1, ..., C_k\}$ of the node set V into k disjoint communities, the quality measure $Q_{inertia}(\mathcal{P})$ is defined by

$$Q_{inertia}(\mathcal{P}) = \sum_{(v,v') \in V \cdot V} \left[\left(\frac{I(V, v) \cdot I(V, v')}{(2|V| \cdot I(V))^2} - \frac{\|v - v'\|^2}{2|V| \cdot I(V)} \right) \cdot \delta(C_v, C_{v'}) \right] \quad (2.8)$$

In Equation 2.8, the inertia $I(V, v)$ of V through v is equal to the sum of the squared Euclidean distances between v and other nodes in V: $I(V, v) = \sum_{v' \in V} \|v - v'\|^2$. The inertia $I(V)$ of V through its gravity center g, which is also called the second central moment in statistics, is a homogeneity measure defined by $I(V) = \sum_{v' \in V} \|v - g\|^2$. C_v denotes the community of nodes v and δ is the Kronecker function which equals 1 when C_v and $C_{v'}$ are the same, and equals 0 otherwise. Based on the inertia modularity, the optimization objective of the I-Louvain method is given by:

$$QQ^+(\mathcal{P}) = Q_{NG}(\mathcal{P}) + Q_{inertia}(\mathcal{P}) \quad (2.9)$$

with Q_{NG} being the Newman modularity [67].

The optimization of Equation 2.9 is similar to those in the Louvain method [10] which includes a two-phase procedure. The first phase is to repeatedly and sequentially move each node to the community of its neighbor to achieve the maximum increment of QQ^+ and the corresponding community partitioning $\mathcal{P}$, and the second phase is to merge all the all the within-community nodes as a super node in preparation for community adjustment in the next iteration. It should be noted that in the second phase, the attribute distance D also needs

to be merged. In the supernetwork G' created by the second phase of Louvain, the distance $D'(x, y)$ between two nodes x, y in G', which corresponds to two communities of G obtained from the first phase, is given by

$$D'(x, y) = \sum_{(i,j)\in V\times V} D(i, j) \cdot \delta(\tau(i), x) \cdot \delta(\tau(j), y) \tag{2.10}$$

where the function τ gives the mapping of a node $v \in V$ to the node $v' \in V'$ which corresponds to the community of v in $\mathcal{P}$.

It has been shown that I-Louvain outperforms k-means and Louvain methods in terms of NMI on real and synthetic networks. Since I-Louvain is an inheritance of Louvain, the method is also very efficient as Louvain, in that in practice, the time complexity grows linearly with the network size.

2.3.2 Methods Based on Unifying Edge Weights

Methods of this kind follow the idea of integrating node attributes into edge weights. They design a similarity measure of the node attributes and later this measure is used to (re-)weigh the edges in the network. Once the edges are (re-)weighted, existing community detection algorithms that can be applied to weighted networks are employed to find the communities.

Neville *et al.* [65] proposed a *matching coefficient* similarity measure that counts the number of attributes that two connected nodes i and j have in common. Matching coefficient is only applicable to networks with categorical node attributes. Later on, Steinhaeuser and Chawla [85] extended matching coefficient measure by defining the weight between two nodes i, j as $w_{ij} = 1 - \alpha_t|f_t(i) = f_t(j)|$ for continuous attribute a_t, with α_t being a normalizing parameter corresponding to attribute a_t.

Cruz *et al.* [25] put forward another way to modify the weights of edges according to the attribute similarity between nodes. To begin with, they use self-organizing map (SOM, [46]) to discover groups of nodes that have similar attributes. If a pair of nodes i, j belong to the same group, the weight of the edge that connects them is changed to a value proportional to a constant $\alpha > 1$. In the last step, the network communities are identified using the

Louvain method.

There are other methods following the same approach. For example, Combe *et al.* [21] used Euclidean and cosine distance to obtain new edge weights. But all of these methods share a major drawback, that only the pairs of nodes that are initially connected will be affected. Nodes that are disconnected will be ignored and will probably be assigned to different communities regardless of how close they are in terms of attributes.

2.3.3 Methods Based on Graph Augmentation

Graph augmentation refers to methods that deal with attribute by adding virtual edges or nodes to the original graph, which results in an *augmented graph*. Example of graph augmentation methods can be found in [42], [81], [104], [105]. Next we detail some representatives of them.

The COmmunity Discovery Inferred from Content Information and Link-structure (CODICIL) is a method proposed by Ruan *et al.* [81]. Given an undirected graph G with each node being characterized by a term vector $\boldsymbol{t}$, the algorithm firstly utilizes attribute information by generating content edges based on the cosine similarity between the TF-IDF vectors of the content vectors, using the top-k criteria. Secondly, a new graph is formed by sampling from the union of content edges and topological edges with a bias towards locally relevant ones, in order to only keep the edges that are relevant in local neighborhoods. In the sampling procedure, the authors look at the neighbors $ngbr(v) = \{u_i\}$ of each node v, compute the Jaccard coefficient of $ngbr(v)$ and each $ngbr(u_i)$ as a topological similarity between v and its neighbors u_i, and also compute their content similarity using TF-IDF vectors. Then the topological and content similarities are combined with a weighting parameter α. For each node v in the graph, only $\sqrt{|ngbr(v)|}$ number of edges are retained in terms of their aggregated similarity values. At last, standard community discovery algorithms are applied to the final version of graph.

The time complexity of the CODICIL algorithm is $O(|V|^2 \log|V|)$. In the paper, the CODICIL method is applied to unweighted graphs with textual at-

tributes. But it is claimed by the authors that the algorithm can be extended to weighted graphs by changing the aggregated similarity to the product of the aggregated similarity and the original edge weight. It can also be extended to categorical attribute since attribute assignment of a node can be represented by an indicator vector, meaning whether a term exists or not.

kNN-enhance [42] shares the same idea as CODICIL as it also looks at the k objects that are most similar in terms of attributes to a specific node v. However in the next step, sampling is replaced by adding directed edges from v to one of its k-nearest neighbors if there does not exit a structural link between them. To measure the attribute similarity between the nodes, the cosine similarity $\mathbf{x}_i \cdot \mathbf{x}'_j$ ($\mathbf{x}_i, \mathbf{x}_j$ are the normalized attribute vectors of nodes i, j) is used for nodes with binary attributes, and $1 - \|\mathbf{x}_i - \mathbf{x}_j\|$ is used for numerical attributes, where $\|\mathbf{x}_i - \mathbf{x}_j\|$ is the normalized Euclidean distance between $\mathbf{x}_i$ and $\mathbf{x}_j$. After the kNN-enhanced network is established, the authors used a K-rank-D method [54] which is based on centrality and dispersion of nodes, to select the community centers. Later on, two attributed community detection methods, kNN-Kmeans and kNN-nearest, are used to perform the community detection tasks on real and synthetic datasets. kNN-Kmeans uses the strategy of the K-means method, which iteratively updates the community centers. kNN-nearest assigns each remaining node to the same cluster as its nearest neighbor of higher Pagerank centrality.

Zhou *et al.* [104] developed an attributed communiyt detection algorithm based on both **S**tructure and **A**ttribute similarities, called SA-Clustering. They utilize attribute information to add virtual nodes and edges to the original graph, hence create an attribute-augmented graph. They also take into account the fact that structural edges as well as different kinds of attribute edges usually have distinct levels of importance, i.e., different types of edges should have different weights. So they propose a weight self-adjustment method to learn the degree of contributions of different attributes.

Firstly, to integrate the structural and attribute similarities into a unified

framework, the method starts by inserting a set of attribute nodes to the original graph G. An attribute node v_{jk} represents an attribute-value pair (a_j, a_{jk}), which means that the attribute a_j takes the value of a_{jk}. The attribute node v_{jk} is connected to a structural node v_i when v_i has the value a_{jk} on its attribute a_j, and are isolated otherwise. In this way the SA-Clustering creates an augmented graph G_a. Next, the authors define a probability transitioning matrix over G_a in terms of the initial weights assigned to the edges, and based upon which the node closeness is measured by a random walk distance. The optimization objective function aims to maximize the between-community random walk distance. A k-medoids clustering approach is used to partition the graph G_a and the edge weights are iteratively updated via a majority vote mechanism till convergence.

Though SA-cluster is proven to be able to find communities where nodes within the community exhibit high density and homogeneity, efficiency becomes a major issue of this method. The random walk distance calculation involves matrix multiplication, which has a time complexity of $O(|V|^3)$, and is repeatedly calculated to update the edge weights. To overcome the computational bottleneck in SA-Cluster, Zhou *et al.* [105] proposed Inc-Cluster, which significantly increases the computational efficiency of the SA-Cluster.

Inc-Cluster seeks to avoid the repeated calculation of random walk distance in the clustering process in SA-Cluster algorithm. It is observed that the weights only change for attribute edges, but not for the structural edges. The authors develop an algorithm that divides the transition probability matrix into submatrices and incrementally update each one such that only the non-zero elements in the increment matrix of random walk distance $\triangle R$ are calculated. The upper bound of the calculation of the non-zero elements is $O(|V|^2)$ and the lower bound is $O(|V|)$. Experiment results show that Inc-Cluster greatly enhances the efficiency while maintaining the clustering quality of SA-Cluster.

2.3.4 Methods Based on Core Expansion

Methods discussed in this section follow a sequential approach to find communities in attributed networks. They first detect a set of core nodes for each community, based on either the linkage or the attribute of nodes. Later on they employ a technique to expand the sets of core nodes to form the final communities.

A representative method of this class is developed by Li *et al.* [53]. They propose a four-step method that is applied to textual data. The method starts by a core probing process, in which they identify representative nodes as community cores based on the relations. The cores are characterized by documents that are frequently referenced. In the core probing step, a parameter t is used as a filtering threshold for the core members. Next, a core merging process is adopted to alleviate the sensitivity to the initial set of t. This merging process is based on the topical similarity between documents, in which the authors firstly find candidate cores that meet the criterion of having at least one overlapping entity. Then they apply Latent Dirichlet Allocation (LDA) on all candidates to study the topical similarities and decide which candidate cores should be merged. The initial communities are formed in this step. Afterwards, all the remaining nodes (i.e. documents) are assigned to a community through a repeated manner: For each document d_i in an initial community C, all non-core documents that are linked to d_i are assigned to the same community C. A maximum number of iterations is set to avoid infinite loops. Finally, a supervised classification method is applied to prune the nodes that are weakly related to a community because of topical ambiguity. The authors use Support Vector Machine (SVM) to train a binary classifier, and in each community all the negatively classified documents are removed.

The FocusCO algorithm proposed by Perozzi *et al.* [73] is another example following the approach of core expansion. Different from [53] that exploits the structural information first, FocusCO starts by inferring the attribute relevance and creating a weighted similarity for each pair of nodes. Next the

algorithm extracts the focused clusters by identifying good candidate sets in which nodes have high weighted similarity to their neighbors. Then each core set is expanded by continually choosing new nodes to include to the cluster until there exist no more nodes that increase the quality of the cluster. The authors choose the conductance (see Equation 2.24 in Section 2.4.2) as the measure of cluster quality.

The peculiarity of the FocusCO algorithm is that FocusCO infers the correlation matrix of all feature vectors of nodes by optimizing an objective function formulated by the Mahalanobis distance. Therefore it is able to put different weights for different node attributes. The weights learned for the attributes are a reflection of user preference in real social network scenario. Furthermore, under the guidance of user preference, the algorithm can also be used to extract outliers from the network under study.

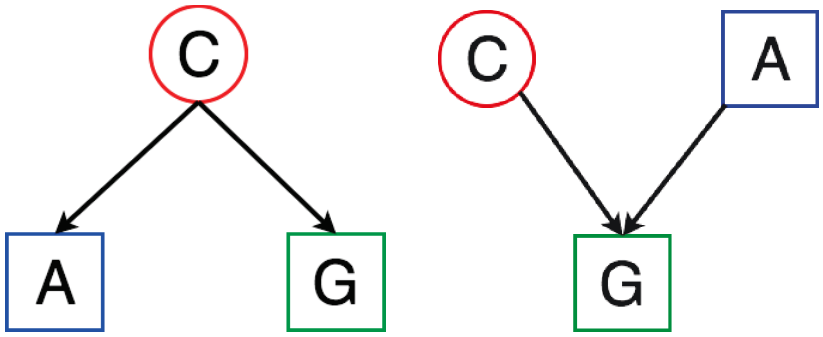
2.3.5 Methods Based on Statistical Inference

In this section we introduce a class of methods that build a statistical model for node attributes as well as their connections. Therefore the detection of network communities is converted to inferring the community assignment of the nodes.

Probabilistic inference offers an abundant amount of methods. To begin with, we can mention the topic models that are based on LDA, including Relational Topic Models (RTM) [17], Topic-Link LDA [56], Block-LDA [6], Relation Strength-Aware Clustering[88], etc. Moreover, we can cite the methods that are based on the discriminative content (DC) model, such as PCL-DC [101], PPL-DC [100] and PPSB-DC [16]. Next we have methods like cohsMix [103], BAGC [95] and GBAGC [96] that incorporate node attributes into the MixNet model [27]; Last but not least, there are other methods like BNPA [20], which is an extension of Newman's mixture model [70], and Metacode [68] which represents attribute as metadata and embeds the metadata in a stochastic block model. In this section, we choose an examplar model, CESNA [99], to elaborate the idea of converting community detection to a statistical inference

problem.

Normally there are two ways of modeling the statistical relationship between a graph G, attributes A, and communities C. The first one assumes that communities and attributes are marginally independent, as shown in Figure 2.3b. However, to allow for dependence between the network and the attributes, [99] assume that communities "generate" both the attributes and the network, as displayed by Figure 2.3a.



(a) (b)

Figure 2.3: Figure in [99], two generative probabilistic models of statistical relationship between graph G, attributes A, and communities C. Squares represent observed variables and circles represent latent variables that need to be inferred.

Next is to build the generative models of the graph structure, i.e., unknown links between known nodes, and of the attributes. The authors assume that each node v has a non-negative affiliation weight $F_{vc} \in [0, +\infty)$. Then they employ the probabilistic generative process of the BigCLAM [97], in which the probability of two nodes u, v belonging to a community c is given by $P_{uv}(c) = 1 - exp(-F_{uc} \cdot F_{vc})$. In this way, each entry of the adjacency matrix a_{uv} is generated from a Bernoulli distribution: $a_{uv} \sim \text{Bernoulli}(P_{uv})$. To predict the attribute values of the nodes based on the community membership, the authors only look at those with binary attributes, and thus a separate logistic regression is considered to model the probability of the attribute k of node v taking the value of 1, denoted by Q_{vk}. Later on the attributes values of the nodes are sampled from another Bernoulli distribution, $\text{Bernoulli}(Q_{vk})$. Now having the

probabilistic model well built, the community detection task is accomplished by maximizing the likelihood function of the community membership (and the weight parameters in the model) based on observed network G and attribute A.

2.3.6 Methods Based on Embedding

The embedding approach is an algorithmic framework for learning continuous feature representations for nodes in networks, initially proposed as *node2vec* by Grover and Leskovec [38]. Node2vec learns a mapping of nodes to a d-dimensional space of features by maximizing the likelihood of preserving network neighborhoods of nodes. Grover and Leskovec design a biased random walk to explore diverse neighborhoods. As a result, the learned node representations reflect the structural equivalence or homophily between nodes. Another famous embedding method is called *struc2vec* [78], which learns node representations from structural identity.

The feature representation of nodes can be applied to varied tasks, such as multi-label classification, link prediction and community detection. Inspired by the node embedding approach, Li *et al.* [55] design a novel community structure embedding framework to utilize both structural and attribute information to detect communities, called CDE (Community Detection Embedding approach) model.

First, to exploit structural information, especially the inherent community structures, the authors employ the skip-gram model with negative-sampling [63] to obtain a new structure embedding matrix $M_{|V|\times|V|} \in \mathcal{R}^{|V|\times|V|}$. For an arbitrary pair of nodes i and j in graph G, a community membership matrix $U_{|V|\times k}$ is defined with k being the number of communities. $U_{i:}$ is the ith row of U that represents the community assignment for node i. The sigmoid function $\sigma(U_{i:}U_{j:}^T)$ measures the similarity of i and j's community membership. Therefore the goal is to learn a matrix U such that $\sigma(U_{i:}U_{j:}^T)$ is maximized for connected i and j, meanwhile minimized for randomly selected i and j. The

negative sampling objective function for nodes i and j is formulated as follows:

$$l(i,j) = w_{ij}(\log \sigma(U_{i:}U_{j:}^T)) + s\frac{d_i d_j}{D} \log \sigma(-U_{i:}U_{j:}^T) \tag{2.11}$$

where w_{ij} is the initial weight of the network under study, s is the size of negative sampling, d_i, d_j are degree of nodes i, j respectively, and $D = \sum_{v \in V} d_v$ is the total degree for graph G.

The optimal $U_{i:}U_{j:}^T$ is learned by finding the partial derivative of Equation 2.11 with respect to $U_{i:}U_{j:}^T$. Then the new structure embedding matrix M can be expressed in terms of U, as given by Equation 2.12

$$M_{ij} = \max\{U_{i:}U_{j:}^T, 0\} = \max\{\log \frac{w_{ij}D}{d_i d_j} - \log s, 0\} \tag{2.12}$$

The maximization in Equation 2.12 is to avoid negative values for entities of M. Hence the problem for detection without considering attributes can be formulated as

$$\min_{U \geq 0} L(U) = \|M - UU^T\|^2 \tag{2.13}$$

Next, to incorporate attribute information, the authors define a community-attribute matrix $T \in \mathcal{R}^{k \times p}$, where T_{ir} represents the preference of i-th community for r-th dimension of attribute. The discovery of community membership based on attribute information is formulated by a nonnegative matrix factorization problem below:

$$\min_{U \geq 0, T \geq 0} L(T) = \|A - UT\|^2 + \alpha \sum_r \|T_{:r}\|^2 \tag{2.14}$$

where A is a $|V| \times p$ node-attribute matrix, α is the learning rate to control the sparsity of T.

Combining Equation 2.13 and 2.14 together, a unified objective function for CDE model is given by:

$$\min_{U \geq 0, T \geq 0} L(U,T) = \|A - UT\|^2 + \alpha \sum_r \|T_{:r}\|^2 + \beta \|M - UU^T\|^2 \tag{2.15}$$

where weighting parameter β is used to leverage the strength of structure information against attribute information.

Applying a non-convex optimization technique [41] over Equation 2.15, one can achieve such communities where nodes are densely connected and have homogeneous attributes in the same community, well-separated and have heterogeneous attributes in different communities. The CDE model is able to find overlapping communities since the entities in U have continuous numerical values. It can also outperforms the CESNA model described in Section 2.3.5 in terms of F1-score and Jaccard Similarity on real networks with ground-truth communities.

2.3.7 Methods Based on Late Fusion

All methods in the previous six categories share one characteristic, that they first employ a technique, which can be an optimization function, a statistical model, an augmented graph, or any other possible approaches, to integrate the network structure and node attributes. Following that direction, they move on to detect the communities. We say these methods following an "early-fusion" approach, because the two sources of information are considered together before any community is identified. There is another class of methods that first detect two different sets of communities using two different partial information: the structure and the attributes, separately and independently. Then these methods try to combine these two sets of communities together to produce the final detection results. We call this approach "late fusion". Cruz and Bothorel [23] propose a method that follows this idea. After having identified two sets of communities based on two different dimensions (the base dimension of structure and a compositional dimension of attribute, according to the authors), a contingency matrix is computed by

$$\mathcal{C} = \mathbf{C}_s^T \mathbf{C}_a$$

where $\mathbf{C}_s$ and $\mathbf{C}_a$ are two node-community affiliation matrices of dimension $|V| \times k_s$ and $|V| \times k_a$. k_s and k_a are the number of communities detected by the two algorithms. The value of entry c_{ij} of matrix $\mathcal{C}$ is the count intersection between community i of $\mathbf{C}_s$ and community j of $\mathbf{C}_a$. The final outcome is

obtained by manipulating rows of $\mathcal{C}$ to form the new communities. The authors propose two ways to define the new communities: a naive approach where a new community is formed if $c_{ij} \geq 0$, and a variance-based approach where a new community is formed only when c_{ij} is greater than the avearge of row i by at least a standard deviation of values in row i.

In the paper, Cruz and Bothorel used Louvain as the community detection algorithm, and SOM (self-organized map, [46]) as the clustering algorithm. They have achieved improvements from community mining based solely on the node connections or attributes on real networks.

2.4 Evaluation

2.4.1 External Measures

The external measures evaluate detected communities against ground-truth communities. Rabbany and Zaïane [75] have examined different evaluation approaches for communities in attributed networks. They showed that there exists a correlation between individual nodes and their connections. In this thesis, we use two well-known external measures, Normalized Mutual Information (NMI) [86] and Adjusted Rand Index (ARI) [40], to evaluate our method described in Chapter 3.

NMI is the normalized version of mutual information, which measures the mutual independence between two variables. Given two partitionings $\mathcal{P}_1$, $\mathcal{P}_2$ of a network G, the NMI of $\mathcal{P}_1$ and $\mathcal{P}_2$, denoted by $NMI(\mathcal{P}_1, \mathcal{P}_2)$, is defined as

$$NMI(\mathcal{P}_1, \mathcal{P}_2) = \frac{MI(\mathcal{P}_1, \mathcal{P}_2)}{\sqrt{H(\mathcal{P}_1)H(\mathcal{P}_2)}} \tag{2.16}$$

where $H(\cdot)$ is the entropy of a partitioning, and $MI(\cdot)$, denoting the mutual information, is given by

$$MI(\mathcal{P}_1, \mathcal{P}_2) = H(\mathcal{P}_1) - H(\mathcal{P}_1|\mathcal{P}_2) \tag{2.17}$$

Therefore, when $\mathcal{P}_1$ and $\mathcal{P}_2$ have a perfect one-to-one correspondence, the NMI reaches its maximum value of 1. When the community assignment of $\mathcal{P}_1$ differs

greatly from $\mathcal{P}_2$, i.e., $\mathcal{P}_1$ is independent from $\mathcal{P}_2$, the NMI is going to be close to 0.

ARI is the adjusted-for-chance version of the Rand Index. Let $\mathcal{P}_1$ be the detected community partitioning of graph G, and $\mathcal{P}_2$ be the ground-truth partitioning. We define *true positive* as the number of node pairs that are assigned to the same community in both $\mathcal{P}_1$ and $\mathcal{P}_2$, and *true negative* as the the number of node pairs that are assigned to different communities in $\mathcal{P}_1$ and $\mathcal{P}_2$. Using a and b to denote the number of true positives and true negatives respectively, the Rand Index is calculated by:

$$RI(\mathcal{P}_1, \mathcal{P}_2) = \frac{a+b}{\binom{|V|}{2}} \tag{2.18}$$

However the Rand Index does not guarantee that random community assignments will get a value close to zero. To counter this effect, ARI is proposed as follows:

$$ARI(\mathcal{P}_1, \mathcal{P}_2) = \frac{RI - E[RI]}{\max(RI) - E[RI]} \tag{2.19}$$

Let us use $C_i^{(1)}$ and $C_j^{(2)}$ to denote the ith community from partitioning $\mathcal{P}_1$ and the jth community from partitioning $\mathcal{P}_2$. Then n_{ij} is the number of nodes in common between $C_i^{(1)}$ and $C_j^{(2)}$. Hubert and Arabie pointed out that RI can be written as:

$$RI = \sum_{ij} \binom{n_{ij}}{2} \tag{2.20}$$

Accordingly the expectation term is:

$$E[RI] = \frac{\sum_i \binom{|C_i^{(1)}|}{2} \sum_j \binom{|C_j^{(2)}|}{2}}{\binom{|V|}{2}} \tag{2.21}$$

and the maximum of RI:

$$\max(RI) = \frac{1}{2}\left[\sum_i \binom{|C_j^{(2)}|}{2} + \sum_j \binom{|C_j^{(2)}|}{2}\right] \tag{2.22}$$

Applying Equation 2.20, 2.21 and 2.22 to Equation 2.19, the ARI can be written as:

$$ARI(\mathcal{P}_1, \mathcal{P}_2) = \frac{\sum_{ij} \binom{n_{ij}}{2} - \sum_i \binom{|C_i^{(1)}|}{2} \sum_j \binom{|C_j^{(2)}|}{2} / \binom{|V|}{2}}{\frac{1}{2}[\sum_i \binom{|C_i^{(1)}|}{2} + \sum_j \binom{|C_j^{(2)}|}{2}] - \sum_i \binom{|C_i^{(1)}|}{2} \sum_j \binom{|C_j^{(2)}|}{2} / \binom{|V|}{2}} \tag{2.23}$$

The value of ARI is bounded to the range of [-1, 1]. It is negative when the RI is greater than the expected RI.

Both NMI and ARI do not require assumption on the cluster structure of the network, and their bounded values provide a good metric to compare the performance of different partitionings found by different community detection methods. Rabbany and Zaïane [76] generalize NMI and ARI and define a new metric called Clustering Agreement Index (CAI). CAI and its variants are very efficient for evluating overlapping communities. However, these external measures require knowledge of the ground-truth labels, which is usually not available in practice. In this case, we have to resort the internal measures.

2.4.2 Internal Measures

Internal measures use community scoring functions to manifest the density and separation of communities detected by an algorithm. Since these methods do not require known ground truth, therefore they can be applied to a wide range of network datasets. One of the most popular internal measures is the modularity as given in Equation 2.3. Recall that modularity measures the difference between the number of edges that lie within communities in a given partitioning $\mathcal{P}$ and the expected such number as if edges were placed at random while maintaining the degree distribution of nodes. Therefore higher modularity indicates better community quality.

Yang and Leskovec [98] studied 13 commonly used internal measures of network communities and evaluated them on a set of 230 large real-world networks. They found out that Conductance [11] and Triad-participation-ratio [98] have consistently outperformed the other measures in identifying ground-truth communities. We briefly mentioned conductance in 2.1.2 and pointed out that conductance measures the fraction of total edges that point outside a given community or cluster. Now let us give the formal definition. Given a community C which is a subset of V, we denote $E_{cb} = \{(u, v) \in E : u \in C, v \notin C\}$ as the set of edges running across the boundary of C. Then the

Conductance is formally defined as:

$$f(C) = \frac{|E_{cb}|}{\min(\sum_{u \in C} d_u, \sum_{v \notin C} d_v)} \tag{2.24}$$

where d denotes the degree of a node in network G. The Conductance in the above equation measures the quality of a single community in the given network. To evaluate k communities produced by a partitioning $\mathcal{P}$, we can simply take the minimum of the conductance of all communities:

$$F(\mathcal{P}) = \min_{C \in \mathcal{P}} f(C) \tag{2.25}$$

From Equation 2.24 and 2.25 we can see that lower conductance indicates better communities.

Triangle-participation-ratio of a community C is the fraction of nodes in C that belong to a triad. Define a set

$$S = \{u : u \in C, \{(v, w) : v, w \in C, (u, v) \in E, (u, w) \in E, (v, w) \in E\} \neq \emptyset\} \tag{2.26}$$

It can be seen that S is the set of nodes in C that also belongs to a triad. Then a the definition of Triangle-participation-ratio is given by $f(C) = \frac{|S|}{|C|}$. A high triangle-participation-ratio indicates a better quality of communities.

Chapter 3

The Late-fusion Method

We introduce another notation r, which we use to denote the number of attributes in a network. Then an $|V| \times r$ matrix A refers to the node-attribute matrix of a given network G, which is the counterpart of the feature matrix in the context of classification and clustering. Next we illustrate the methodology of our proposed late-fusion approach.

3.1 Motivations

Section 2.3 introduces a wide range of work devoted to community detection on attributed networks. However all but the last category follow an "early-fusion" manner, i.e., they consider the structure and attribute in the early stage of the community mining process, notably before any community has been identified. Contrary to early fusion, late fusion approaches aim at firstly identifying communities from the perspectives of structure and attribute, and then fusing two sets of communities in such a way that the final communities represent an integration of the structure and attribute of the network.

Thanks to the nature of late fusion, it provides us with a flexible architecture with which we can try different combinations of classic community detection and clustering algorithms in the implementation. For example, if attributes are well clustered when projected into a r-dimensional feature space, a simple clustering algorithm such as k-means can be applied. If the size of community is too small or too large, algorithms such as Louvain [10] or InfoMap [80] might fall into the resolution limit or field-of-view limit respectively. In

this case, the SIWO method described in 2.1.4 is preferred. The choice of algorithms mainly depends on our prior knowledge about the network: the real number of communities, network size, community size, types and distributions of attributes, and so forth. With early-fusion methods, we lost this flexibility to accordingly adjust our approach in community detection.

To the best of our knowledge, the only published algorithm that follows the late-fusion approach is the work done by Cruz and Bothorel [23] . As we described in 2.3.7, the authors use Louvain as the community detection algorithm, and SOM (self-organized map, [46]) as the clustering algorithm. They obtain the final partitioning by manipulating rows of the contingency table of two partitions based on node connections and attributes. We would like to explore other possible combinations of algorithms, and other variants of late-fusion techniques that can be more successfully applied to networks with binary/categorical attributes. Therefore, we devote next a few sections to describe our own late-fusion approach.

3.2 Method Overview

The idea of late fusion in attributed community detection is that one first identifies two sets of communities/clusters using two sources of information: the linkage structure of the network and the node attributes. Then the ultimate partitioning that has taken both information into consideration is achieved by combining the aforementioned two sets of communities in a carefully designed manner.

In our approach, we use the Louvain and SIWO algorithms described earlier to obtain the set of communities based on linkage. To get the clusters based on node attributes, we treat categorical or binary attributes differently from numerical attributes. As for numerical attributes, we can directly apply classic unsupervised clustering algorithms to identify the clusters. However regarding categorical or binary attributes, we create a virtual graph based on the node attributes. Concretely, if the attribute similarity between two given nodes is large enough, i.e., higher than a precomputed threshold, we add a virtual

edge between these two nodes, otherwise there's no edge between the nodes. Having this virtual graph well-built, next we can apply Louvain or SIWO on this virtual graph and it results in a partitioning represents the similarity between the attributes of the nodes.

Next in the fusion step, we combine the two partitionings by processing their node-community affiliation matrices and creating an integrated adjacency matrix. In this way, the final partition is obtained by applying Louvain or SIWO on an integrated graph induced from the integrated adjacency matrix. We detail our proposed method in the following section.

3.3 The Fusion Algorithm

We begin by presenting our way of combining two sets of communities, as displayed in Algorithm 1. Suppose we are given two partitions over a network G:

$$\mathcal{P}_s = \{C_1, C_2, ..., C_{k_1}\}$$

$$\mathcal{P}_a = \{C_1, C_2, ..., C_{k_2}\}$$

where $\mathcal{P}_s$ is the partitioning obtained by using only the structural information and $\mathcal{P}_a$ is the partitioning obtained by using only the attribute information. Two other inputs are also given: a weighting parameter α to leverage the strength between the two sources of information, and a community detection algorithm F_s to find the final partition over the integrated graph.

In line 1 of Algorithm 1, we compute the node-community affiliation matrices $\mathbf{C}_s$ ($|V| \times k_1$) and $\mathbf{C}_a$ ($|V| \times k_2$) for $\mathcal{P}_s$ and $\mathcal{P}_a$ respectively. The entries of the affiliation matrices are binary, where $c_{ij} = 1$ if and only if node i is assigned to community j. Next we multiply every affiliation matrix by its transpose, in this way we obtain matrices D_s and D_a of dimension $|V| \times |V|$ (see line 2). The entries of D_s and D_a are also binary because we only consider non-overlapping communities hence each row of an affiliation matrix $\mathbf{C}$ is a one-hot vector. A value 1 at the ith row and the jth column of D_s or D_a indicates that nodes i and j are assigned to the same community in $\mathcal{P}_s$ or $\mathcal{P}_a$. Next we combine D_s and D_a by the equation in line 3. Since we only consider non-directed

networks in this work, we need to fill the diagonal of the matrix D to be 0s, as shown by line 4. A value 1 at the ith row and the jth column ($i \neq j$) of D shows that nodes i and j are assigned to the same community in both $\mathcal{P}_s$ and $\mathcal{P}_a$; A value of D_{ij} equaling α or $1-\alpha$ indicates that nodes i and j are assigned to the same community in either $\mathcal{P}_s$ or $\mathcal{P}_a$, but not in both partitions; $D_{ij} = 0$ means the two corresponding nodes are never in the same community. Now the matrix D can be taken as a weighted adjacency matrix, based upon which we can build the integrated graph $G_{integrated}$. However in practice, it would make our algorithm run much faster without hurting the accuracy if we make D unweighted. To do so, we set a threshold $t \in (0, 1)$, below which the entries of D are reset to zero. Immediately in line 5, we build an unweighted virtual graph from the adjacency matrix D. Now the edges in $G_{integrated}$ is a representation of the agreement between structural communities and attribute clusters. The final partitioning is achieved by applying F_s on $G_{integrated}$. The partitioning $\mathcal{P}$ will be our output, containing k disjoint communities over the original network.

Algorithm 1: The fusion algorithm

Input: $\mathcal{P}_s, \mathcal{P}_a, \alpha, F_s$
Output: $\mathcal{P} = \{C_1, C_2, ..., C_k\}$
1 $\mathbf{C}_s$ = get_affiliation_matrix($\mathcal{P}_s$), $\mathbf{C}_a$ = get_affiliation_matrix($\mathcal{P}_a$)
2 $D_s = \mathbf{C}_s\mathbf{C}_s^T$, $D_a = \mathbf{C}_a\mathbf{C}_a^T$
3 $D = \alpha D_s + (1-\alpha)D_a$
4 D = fill_diagonal_zero(D)
5 $G_{integrated}$ = from_adjacency_matrix (D)
6 $\mathcal{P} = F_s(G_{integrated})$
7 **return** $\mathcal{P}$

With the late-fusion algorithm well-defined, we can apply it to networks with node attributes. However for different types of attributes, the way to identify $\mathcal{P}_a$ differs, which we illustrate in the following sections.

3.3.1 Late Fusion on Networks with Numeric Attributes

The most commonly-seen realization where node attributes are numeric are document networks. For example, the polblog network [102] where a node is

an article of political comments, the edges are references to other articles in a political blog, and the attributes are usually represented by a feature vector obtained with techniques like LDA [9] or doc2vec [38] for each document. Numerical attributes can also be associated to online forum members, in which by analyzing all the posts of a user using similar techniques as in the political blog network, we can attach a feature vector to each node (a member of a forum) to demonstrate their opinions, interests, etc. Then the edges can be defined as the interactions between the forum members in a natural way. An example of such case is the Sina Weibo network developed by Jia *et al.* [42].

The extension of the fusion algorithm to networks with numeric attributes is very simple, which we describe in Algorithm 2. The inputs are: the original

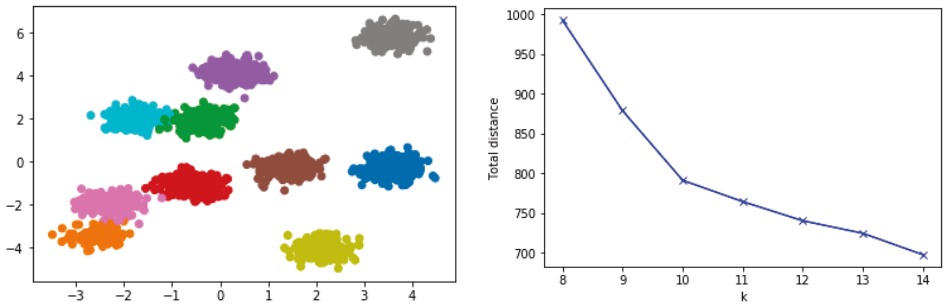
Algorithm 2: Late-fusion on networks with numeric attributes

Input: G_s, A, F_s, F_a, α
Output: $\mathcal{P} = \{C_1, C_2, ..., C_k\}$
1 $\mathcal{P}_s = F_s(G_s)$, $\mathcal{P}_a = F_a(A)$
2 $\mathcal{P}$ = Algorithm 1$(\mathcal{P}_s, \mathcal{P}_a, \alpha, F_s)$
3 **return** $\mathcal{P}$

graph $G_s = (V, E)$ which is comprised by a set of nodes V and a set of edges E; a $|V| \times r$ matrix containing the feature vectors of the nodes; two predetermined algorithms $F_s and F_a$: F_s for detecting communities based on the node connections, and F_a for clustering according to the node attributes; a weighting parameter α that leverages the strength of two sources of information. We apply F_s and F_a to G_s and A respectively, to obtain $\mathcal{P}_s$ and $\mathcal{P}_a$. Then we can directly feed $\mathcal{P}_s$, $\mathcal{P}_a, \alpha$ and F_s into the the fusion algorithm (Algorithm 1) and we get the output partitioning $\mathcal{P}$ according to Algorithm 1.

Some of the clustering algorithms require the number of communities k as an input (k-means and spectral clustering, for example), we can use the elbow method to infer the value of k. The elbow method is used to visualize the percentage of variance explained by the current clusters, i.e., the ratio of the between-cluster variance to the total variance against k. We use a simple example to illustrate the idea, as shown in Figure 3.1. Figure 3.1a is a visualization of randomly generated 10 clusters. To simplify the process, we

just compute the sum of distance between data points to its clustering centers against different choices of k, and plot the results in Figure 3.1b. We can see that the "elbow point" is at when $k = 10$. Hence we choose 10 as the optimal number of clusters/communities, which is the exact real number of clusters that we have generated.



(a) Random generated 10 clusters on 2d plane

(b) The elbow method showing the optimal k

Figure 3.1: Illustration of the elbow method using a simple example

3.3.2 Late Fusion on Networks with Binary Attributes

For online social networks, the user profile can be naturally taken as the attribute of a node. A profile contains the information of an user such as education, occupation, hobbies, residence, etc. These attributes are not easily quantifiable. On the other hand, however, we can convert a profile to a binary feature vector indicating, for example, in an anonymous way that a person has gone to university A for college, worked at a technology company B, and is interested in activity C, etc. When the attributes of a network is no longer numeric and do not cluster in the feature space, classic clustering algorithms based on the physical distance between data points may not work well on this setting anymore. To cope with this difficulty, we devise a new technique to obtain the partitioning $\mathcal{P}_a$. This new technique, presented in Algorithm 3, is designed for networks with binary node attributes. However it is applicable to networks with categorical attributes as well. We can do this by flattening an integer label which represents categorical class to a one-hot vector. Therefore for the rest of this thesis, we will not specifically distinguish categorical

attributes from binary attributes, and will focus on illustration of binary attributes only.

Algorithm 3: Late-fusion on networks with binary attributes

Input: G_s, A, F_s, α
Output: $\mathcal{P} = \{C_1, C_2, ..., C_k\}$
1 $\mathcal{P}_s = F_s(G_s)$
2 $S = AA^T$
3 $D_a = \text{convert\_to\_adjacency}(S)$
4 $G_a = \text{from\_adjacency\_matrix}\ (D_a)$
5 $\mathcal{P}_a = F_s(G_a)$
6 $\mathcal{P} = \text{Algorithm 1}(\mathcal{P}_s, \mathcal{P}_a, \alpha, F_s)$
7 **return** $\mathcal{P}$

Compared to Algorithm 2, we no longer need the clustering algorithm F_a. Instead, to get clusters $\mathcal{P}_a$, we take advantage of the fact that the entries of A is binary. As shown by line 2, we multiply attribute matrix A with its transpose. The outcome matrix S is a similarity matrix whose entry S_{ij} is the inner product of the ith row vector and the transpose of the jth row vector, which is essentially the count of shared attributes between nodes i and j. Therefore, the higher the value S_{ij} is, the more similar nodes i and j are. The next operation is to convert similarity matrix S to a virtual adjacency matrix D_a (line 3), from which a virtual graph G_a is built. To do so, we again need to find a proper threshold, which we will address shortly. Now the edges in G_a indicate that the similarities between the corresponding pairs of nodes are higher than the threshold that we've chosen. Having G_a well-established, we can apply F_s to G_a and get $\mathcal{P}_a$. The rest is to just apply our fusion algorithm (Algorithm 1) with inputs $\mathcal{P}_s, \mathcal{P}_a$, α and F_s (line 4 - 6).

Line 3 of Algorithm 3 requires to be carefully taken care of because there are many variants that we can choose from. The idea is to find a threshold t such that we set all entries of S below t to 0. Moreover, we can also reset values greater than t in S to 1 to get an unweighted virtual graph, or we can leave it as it is which results in a weighted graph. In practice, we notice that the difference between weighted and unweighted graphs does not influence much of the community detection result. However it is the way to define the threshold

t that matters. Next, we provide two ways to define the threshold:

1. **Median thresholding**: we choose the median of all the similarities as the threshold. Specifically, we take all the off-diagonal, upper triangular (or lower triangular) entries, find the median of these numbers and set it as the threshold. In this way, for a given pair of nodes i and j ($i \neq j$) to have a virtual edge in G_a, i and j must have more shared attributes than at least half of all node pairs.

2. **Equal-edge thresholding**: in this way we use the number of edges $|E|$ in the original graph G_s as the proxy for the number of edges that we should have in the virtual graph G_a. To do so, we compute the density of the original graph by the formula: $d(G_s) = \frac{2|E|}{|V|\cdot(|V|-1)}$. Then $q = 1 - d(G_s)$ is the quantile of the similarity distribution at which we should make the cut, i.e., we set entries in S whose values are below q quantile of all the similarities to 0.

The choice of thresholds is a hyper-parameter of the late-fusion method which can be tuned. We suggest the default option to be the equal-edge thresholding, as we believe that the real number of edges in the original graph reflects the appropriate density for the network. We will also see in the next chapter, that equal-edge thresholding gives better detection results than median thresholding. Another advantage of creating a virtual graph to get $\mathcal{P}_a$ is that in most cases it does not require the prior knowledge about the real number of communities. Throughout the community detection process, we are only dependent on F_s, for which we mostly land on Louvain or SIWO due to their excellent performance. Both algorithms are a hierarchical optimization procedure, which will automatically terminate when it has reached an (local) optima.

Chapter 4

Experiments

We use this chapter to discuss our experiments. Before we come to the experimental results, we first introduce our experimental settings, including the network datasets, the comparative methods, and the evaluation metrics. Then we demonstrate our experiments and discuss the results. Next we provide an empirical study on the effect of the weighting parameter α that is introduced in line 3 of Algorithm 1. After that, we analyze the time complexity of our late fusion approach and show its scalability. At the end of this chapter, we demonstrate our work on the late fusion approach using census clustering. In particular, we point out that consensus clustering is not a good choice for the late fusion technique.

4.1 Experimental Settings

For our experiments, we choose 15 networks of various types and sizes. We classify these networks into the following three groups:

- Synthetic networks with numerical attributes: We use an attributed graph generator developed by Largeron *et al.* [49] to create four attributed graphs with communities. The attributes of these networks are generated by a k-dimensional multivariate normal distribution with zero mean and a standard deviation specified by users. The communities are created according to the homophily effect between the structure and the attribute, which we have introduced earlier in Section 1.1. We treat the

generated communities as the ground-truth since it is in accordance with our objectives put forward in Section 2.3.

- Real network with numerical attributes: The Sina Weibo [1] is the largest online Chinese micro-blog social networking website. Jia *et al.* have collected users from 10 major forums to build a network with 3490 nodes and 30282 edges. The authors performed LDA topic modeling on the micro-blogs of users and created 10 dimensional numerical attributes to describe users' interests. The ground-truth communities of this network is provided by the authors based on the 10 forums where they collected the users and their mutual relationships. This dataset is available online [2].

- Real networks with binary attributes: The Facebook dataset is developed by Leskovec and Mcauley [52]. It contains 10 egocentric networks with binary attributes. The attributes of a node contain the anonymous information of the user about name, work and education. The ground-truth communities were gathered by a survey of 10 users who were asked to manually identify all their social circles (i.e., communities) to which their friends belonged. The circles could overlap, and there are also outliers which do not belong to any circle. This dataset is also availabe online [3].

Table 4.1, 4.2 and 4.3 present several characteristics about the datasets, including the number of nodes $|V|$, the number of edges $|E|$, the number of ground-truth communities k, the number of attributes r, the moduarity Q of the ground truth communities, and the within-inertia ratio I (formula given in Equation 4.1) of the ground-truth communities (only for numeric attributed networks).

To show the competitiveness of our approach, we take advantage of the availability of the source code of four algorithms to compare with our method.

[1] `http://www.weibo.com`
[2] `https://github.com/smileyan448/Sinanet`
[3] `http://snap.stanford.edu/data/`

Table 4.1: Network and community characteristics of synthetic networks.

	$\|V\|$	$\|E\|$	k	r	Q	I
Graph 1	2000	7430	10	2	0.81	0.18
Graph 2	2000	8378	10	2	0.72	0.18
Graph 3	2000	7445	10	2	0.65	0.18
Graph 4	2000	6988	10	2	0.54	0.19

Table 4.2: Network and community characteristics of Sina Weibo network.

$\|V\|$	$\|E\|$	k	r	Q	I
3490	30282	10	10	0.05	0.04

Table 4.3: Network and community characteristics of Facebook networks.

Network ID	$\|V\|$	$\|E\|$	k	r	Q
0	347	5038	24	224	0.179
107	1045	53498	9	576	0.218
348	227	6384	14	161	0.210
414	159	3386	7	105	0.468
686	170	3312	14	63	0.101
698	66	540	13	48	0.239
1684	792	28048	17	319	0.509
1912	755	60050	46	480	0.339
3437	547	9626	32	262	0.026
3980	59	292	17	42	0.242

We first choose two very successful community detection algorithms: Louvain[10] and SIWO[36]. We have introduced both algorithms in Section2.1.4. To assess our late-fusion approach under numerical node attributes, we select two algorithms as our contenders. Due to the availability of the source code implemented by the original authors, we have the I-Louvain [22] algorithm for networks with numerical attributes and the CESNA[99] algorithm for networks with binary attributes. Both I-Louvain and CESNA have been discussed in Section 2.3.

In order to evaluate the performance of different methods, we measure the results produced by each method in terms of accuracy and efficiency. Since all of our experimental datasets have ground-truth communities provided, we evaluate the accuracy using the two external metrics, NMI and ARI described in Section 2.4. For efficiency, we calculate the time elapsed in seconds from the inputted attributed networks to the outputted detected communities. Since in the experiment of the Facebook networks we do not use any prior knowledge about the real number of communities, we also compute the ratio of the number of communities detected to the real number of communities and present the results only for the facebook dataset.

4.2 Experimental Results

4.2.1 Synthetic Networks with Numerical Attributes

Users can change a set of different parameters of the attributed network generator [12] to control properties of the network such as network size, number of ground-truth communities, structural quality, attribute quality (i.e., the homogeneity of the attributes of nodes residing in the same communities) and so forth. The structural quality of the generated ground-truth communities are measured by the formula of modularity in Equation 2.3. The attribute quality is evaluated by another measure called *within inertia ratio*, whose definition is given below:

$$I(G) = \frac{\sum_{C \in \mathcal{P}} (W_C \sum_{vinC} d(v, g_C)^2)}{\sum_{v \in V} d(v, g)^2} \tag{4.1}$$

where g_C is the center of gravity of nodes in C, W_C is the weight of C and g is the center of gravity of all the nodes. So for a set of communities, the higher value of modularity indicates stronger structural strength, whereas lower value of within inertia ratio shows better attribute homogeneity.

In this set of experiments, we generate four networks with the same number of nodes, number of ground-truth communities, number of attributes, and within inertia ratios, but with different modularities. In this way we are able to examine the influence of the structural qualities of the networks to the final partitionings without confounding factors. The properties of the four synthetic networks are presented in Table 4.1

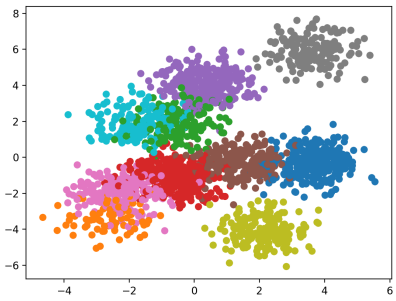
Attribute redistribution

The original node attributes created by the attributed graph generator [49] follow a multivariate normal distribution with zero mean and a standard deviation specified by users. This attribute distribution does not inherently cluster, as we can see in Figure 4.1c. Therefore, we also reassign new attribute vectors to the nodes of the experimental networks based upon their ground-truth community memberships, so that the node attributes have more clear clustering structure. Concretely, the attribute redistribution procedure follows the two steps below:

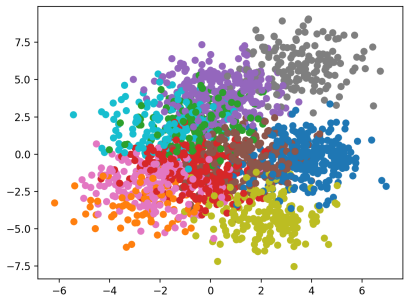
1. Given the number of communities, k, we first generate k center points for each community following a multivariate normal distribution with zero mean: $N((0,0), 5 \cdot \mathbf{I}_{2\times 2})$. Here $\mathbf{I}_{2\times 2}$ is the 2-D identity matrix. It should be noted that the coordinates of the center points are not necessarily attributes for certain nodes. They are used to determine the mean of the attributes of nodes in same communities.

2. Suppose the center of community i is denoted by $\mathbf{c}_i = (a_{i1}, a_{i2})$. Then for every node v belonging to a ground-truth community $i, (i = 0, 1, ..., k-1)$, we re-assign a new attribute vector to v generated by another multi-normal distribution with the mean being the corresponding community center, and a user-specified standard deviation: $N(\mathbf{c}_i, std \cdot \mathbf{I}_{2\times 2})$.

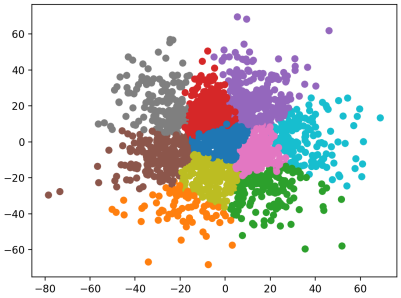
Figure 4.1 plots the node attributes of one of the experimental networks onto a two dimensional plane, using different colors to denote different ground-truth communities. We visualize the attribute distribution with the *std* in step 2 set to 0.5, 1.5 as well as the original attributes provided by the attributed graph generator.



(a) Attributes for the first group of experiments, $std = 0.5$, within inertia ratio = 0.009



(b) Attributes for the second group of experiments, $std = 1.5$, within inertia ratio = 0.022



(c) Attributes for the third group of experiments, no redistribution, within inertia ratio = 0.19

Figure 4.1: Node attributes for three groups of experiment. Each color represents a unique community

Experiment

As mentioned earlier, we have three different types of attribute distributions for a set of four graphs that are generated by the same parameters except that they have different modularities of their ground-truth communities. Hence we conduct three groups of experiments, corresponding to the three node at-

tribute distributions that are plotted in Figure 4.1a, 4.1b and 4.1c. In each group, we perform community mining using a various methods on the four graphs. We measure the NMI, ARI and the running time of each method on the four graphs to evaluate the performance of these methods. In all experiments, the α parameter in step 3 of Algorithm 1 is chosen to be 0.5, i.e., the same weights between structural and attribute information. For community detection algorithm F_s, we use Louvain and SIWO in the experiment. For clustering algorithm F_a, we choose spectral clustering (SC) and DBSCAN as the representatives. Spectral clustering requires a user to specify the number of clusters to identify, hence we pass the ground-truth number of communities to the algorithm. Meanwhile DBSCAN requires two hyper parameters: an ϵ which is the maximum distance between two samples for them to be considered as in the same neighborhood (i.e., community), and an m which is the number of samples in a neighborhood for a point to be considered as a core point. In our experiments we use the default value for ϵ in the Scikit-learn [15] package, and choose m to be the average degree of nodes.

Table 4.4: Results of experiment group 1, $std = 0.5$, time is measured in seconds.

	Graph 1			*Graph* 2			*Graph* 3			*Graph* 4		
	NMI	ARI	time	NMI	ARI	time	NMI	ARI	time	NMI	ARI	time
Louvain	.795	.797	0.41	.696	.692	0.56	.695	.686	0.49	.665	.674	0.64
SIWO	.836	**.850**	0.97	.739	**.750**	1.23	.702	**.705**	1.09	.504	.458	0.98
SC	.802	.713	1.15	.798	.711	0.93	.777	.677	0.64	**.768**	.669	0.68
DBSCAN	.469	.103	0.06	.465	.104	0.06	.434	.083	0.06	.465	.102	0.24
I-Louvain	.515	.150	39.2	.732	.720	37.5	.718	.704	30.0	.608	.503	37.6
Louvain + SC	.824	.704	7.34	.790	.650	6.68	.784	.618	5.74	.765	.597	7.14
Louvain + DBSCAN	.818	.813	8.64	.742	.720	8.59	.730	.702	8.87	.704	**.690**	10.6
SIWO + SC	**.844**	.738	10.3	**.806**	.689	9.28	**.786**	.636	7.33	.723	.508	6.46
SIWO + DBSCAN	.818	.813	11.7	.742	.720	10.5	.730	.702	10.2	.704	**.690**	11.6

Table 4.4, 4.5, and 4.6 show us the NMI, ARI and running time of the three groups of experiments, produced by two baseline algorithms, Louvain and SIWO, that only use structural information; two clustering algorithms, Spectral Clustering and DBSCAN, that only use attribute information; one

Table 4.5: Results of experiment group 2, $std = 1.5$, time is measured in seconds.

	Graph 1			*Graph* 2			*Graph* 3			*Graph* 4		
	NMI	ARI	time	NMI	ARI	time	NMI	ARI	time	NMI	ARI	time
Louvain	.795	.797	0.44	.696	.692	0.54	.695	.686	0.47	.665	**.674**	0.61
SIWO	**.836**	**.850**	0.94	**.739**	**.750**	1.26	.702	**.705**	1.18	.504	.458	1.19
SC	.529	.338	0.83	.527	.329	0.56	.522	.322	0.53	.538	.349	0.57
DBSCAN	.096	.012	0.08	.102	.014	0.10	.066	.008	0.14	.065	.011	0.09
I-Louvain	.517	.150	36.8	.726	.715	36.4	**.707**	.690	33.7	.614	.522	33.2
Louvain + SC	.734	.450	5.62	.695	.398	6.49	.696	.390	5.96	**.677**	.392	5.66
Louvain + DBSCAN	.755	.726	9.20	.663	.627	13.8	.670	.636	11.9	.641	.633	13.6
SIWO + SC	.748	.469	12.7	.712	.424	5.27	.699	.402	7.12	.625	.335	7.44
SIWO + DBSCAN	.744	.726	8.73	.663	.627	9.99	.670	.636	8.98	.641	.633	12.4

Table 4.6: Results of experiment group 3, no attribute redistribuion, time is measured in seconds.

	Graph 1			*Graph* 2			*Graph* 3			*Graph* 4		
	NMI	ARI	time	NMI	ARI	time	NMI	ARI	time	NMI	ARI	time
Louvain	.795	.797	0.67	.696	.692	0.62	.695	.686	0.40	.665	**.674**	0.73
SIWO	**.836**	**.850**	1.02	**.739**	**.750**	2.14	.702	**.705**	1.46	.504	.458	1.92
SC	.483	.270	3.31	.485	.270	2.29	.514	.307	2.32	.489	.276	2.45
DBSCAN	.000	.000	0.06	.000	.000	0.06	.000	.000	0.06	.000	.000	0.14
I-Louvain	.517	.150	35.1	.726	.715	34.3	.707	.690	34.3	.614	.522	39.5
Louvain + SC	.770	.670	11.8	.704	.589	9.27	.705	.613	10.2	**.689**	.564	9.33
Louvain + DBSCAN	.795	.797	11.2	.696	.692	10.0	.695	.685	10.4	.667	.674	12.9
SIWO + SC	.797	.703	13.2	.731	.647	16.6	**.709**	.635	12.3	.601	.467	11.0
SIWO + DBSCAN	.795	.797	11.6	.696	.692	10.6	.695	.685	11.3	.667	**.674**	12.6

contending algorithm, I-Louvain, whose source code is available online [4]; and four different combinations of late fusions. The node attributes of experiment group 1 exhibit the best clustering property. Hence we can see that our late-fusion methods outperform the other algorithms on all four networks in terms NMI, and on graph 4 in terms of ARI. On the other hand, the structural strength (i.e., modularity) decreases as we go from graph 1 to graph 4, at the same time, we can see the advantage of our late-fusion methods over the others increases in terms of both NMI and ARI. This observation indicates that the information about attribute becomes more valuable when high quality of structural communities are less available. As for running time, despite the

[4] `https://www.dropbox.com/sh/j4aqitujiaifgq4/AAAAH0L3uIPYNWKoLpcAh0TPa`

fact that classic community detection algorithms are still the fastest, which is as expected since they do not consider node attributes, our late-fusion methods outperform the attributed community mining contender I-Louvain with a remarkable margin.

Now if we move on to Table 4.5 and 4.6, we can see the same trend going on horizontally from graph 1 to graph 4. However vertically, the performance of our late-fusion methods degrad in terms of NMI and ARI compared to Table 4.4. This degradation of accuracy tells us when the clusters based on attributes are poorly detectable, it becomes harder to make use of attribute information to improve the detection results. On the other hand, if the structure of the network is strong enough, we can simply rely on the node linkages to achieve decent community mining results.

4.2.2 Numeric attributes, Sina Weibo network

To further evaluate our proposed methods, we extend the experiments to a real social network where every node represents a user and has 10 attributes in numerical values indicating their interests in the fields of finance and economics, literature and arts, fashion and vogue, current events and politics, sports, science and technology, entertainment, parenting and education, public welfare, and life style [42]. We seek to find the ground-truth forum that every user comes from in this large network.

In this experiment, we have implemented our late-fusion methods a little bit differently from what we did in the previous section. First of all, we noticed that the clustering result is very sensitive to the ϵ parameter in the DBSCAN algorithm and it is extremely difficult to infer a good value for it without resorting to further knowledge. We decided to use a simple clustering algorithm, k-means, as a supplement to spectral clustering. The second change we made is that since the ground truth communities are based on the topics of the forums that the users come from, we reckon that the formation of communities depends more on the attribute than the structure. After trying different values of α: 0.1, 0.2, 0.3, 0.4 and 0.5, we found that as long as $\alpha < 0.5$, the late-fusion methods give ideal results. In Table 4.7, $\alpha = 0.2$. Readers can

refer to Table 4.7 that shows the NMI, ARI and running time of different algorithms.

Table 4.7: Experimental results of different community detection methods on Sinanet network. Time is measured in seconds.

	NMI	ARI	time
Louvain	.232	.197	1.98
SIWO	.040	.000	3.26
SC	.612	.520	3.16
k-means	**.649**	**.579**	0.25
I-Louvain	.204	.038	261.
Louvain+SC	.611	.519	48.9
Louvain+k-means	**.649**	**.579**	42.1
SIWO+SC	.611	.519	37.9
SIWO+k-means	**.649**	**.579**	50.4

The two baseline algorithms Louvain and SIWO and the contending algorithm I-Louvain performed poorly on the Sinanet network, whereas the clustering algorithms showed high accuracy. Especially, the k-means algorithms together with our four late-fusion methods with the emphasis on attribute information produce detection results with the best NMI and ARI. These results confirm our assumption that the communities of this network are mainly determined by the attributes, i.e., the user interests of different topics. We also noticed that by changing the weight, the community detection result is dominated by the part of information that has the larger weight. As shown in the table 4.7, even though the accuracy of SIWO algorithm on this network is very low, the effect of misleading structural information is balanced out when SIWO is combined with a suitable clustering algorithm. We will further explore the effect of weighting parameter α in Section 4.2.4.

4.2.3 Binary attributes, Facebook networks

The properties of the Facebook networks are displayed in Table 4.3. For this experiment, we do not use the ground-truth number of communities, and let the algorithms themselves to infer the number of communities that should be

detected. Also we set α to its default value 0.5, because we do not know the preference a user has between attributes and connections when identifying social circles.

As mentioned earlier in Section 3.3.2, there are various implementations of late fusion on networks with binary attributes with different combinations between weighted or unweighted virtual graph, median or equal-edge thresholding, and Louvian or SIWO algorithms. Having extensively explored these different combinations, we found that on the Facebook networks, Louvain generally outperforms SIWO, equal-edge thresholding is better than median thresholding. Results are better with weighted virtual network than unweighted if using median thresholding, and better with unweighted virtual network than weighted if using equal-edge thresholding. Therefore, we only provide results of selective methods. We still treat Louvain and SIWO as our baselines. We use CESNA algorithm [99] as our contender. The source code of CESNA is available online [5]. We provide the outcomes of two late-fusion methods: Louvain + unweighted virtual graph with equal-edge thresholding, and Louvain + weighted virtual graph with median thresholding. We present the results of NMI, ARI, running time and ratio of number of communities in Table 4.8, 4.9, 4.10 and 4.11 respectively.

Table 4.8: NMI of different community detection results on facebook network. Late-fusion 1 refers to Louvain + unweighted virtual graph with equal-edge thresholding, Late-fusion 2 refers to Louvain + weighted virtual graph with median thresholding, Late-fusion 3 refers to SIWO + unweighted virtual graph with equal-edge thresholding, and Late-fusion 4 refers to SIWO + weighted virtual graph with median thresholding.

	0	107	348	414	686	698	1684	1912	3437	3980	Average
Louvain	.382	.332	.478	**.609**	.284	.281	.047	**.565**	.181	**.729**	.389
SIWO	.390	.363	.375	.586	.215	.259	.053	.557	.174	.605	.358
CESNA	.263	.249	.307	.586	.238	.564	.438	.450	.176	.552	.382
Late-fusioin 1	**.558**	.355	**.525**	.538	**.463**	**.669**	**.462**	.511	**.310**	.704	**.509**
Late-fusioin 2	.452	.341	.489	.556	.351	.479	.323	.491	.262	.696	.444
Late-fusioin 3	.541	**.364**	.452	.531	.406	.630	.460	.509	**.310**	.648	.485
Late-fusioin 4	.431	.353	.405	.538	.252	.406	.332	.491	.260	.588	.406

[5] `http://snap.stanford.edu/`

Table 4.9: ARI of different community detection results on facebook network. Late-fusion 1 refers to Louvain + unweighted virtual graph with equal-edge thresholding, Late-fusion 2 refers to Louvain + weighted virtual graph with median thresholdingl, Late-fusion 3 refers to SIWO + unweighted virtual graph with equal-edge thresholding, and Late-fusion 4 refers to SIWO + weighted virtual graph with median thresholding.

	0	107	348	414	686	698	1684	1912	3437	3980	Average
Louvain	.143	.148	.303	.558	.110	.000	.000	.461	.000	.398	.209
SIWO	.220	.177	.127	.519	.000	.009	.000	.419	.002	.209	.167
CESNA	.073	.097	.156	.480	.001	.202	.310	.361	.014	.067	.176
Late-fusioin 1	.024	.047	.103	.265	.006	.000	.043	.252	.000	.069	.008
Late-fusioin 2	.061	.079	.129	.413	.063	.000	.048	.235	.000	.084	.110
Late-fusioin 3	.043	.045	.124	.252	.003	.000	.057	.235	.000	.095	.009
Late-fusioin 4	.108	.079	.141	.391	.040	.016	.060	.223	.000	.073	.113

Table 4.10: Running time of different community detection results on facebook network, measured in seconds. Late-fusion 1 refers to Louvain + unweighted virtual graph with equal-edge thresholding, Late-fusion 2 refers to Louvain + weighted virtual graph with median thresholding, Late-fusion 3 refers to SIWO + unweighted virtual graph with equal-edge thresholding, and Late-fusion 4 refers to SIWO + weighted virtual graph with median thresholding.

	0	107	348	414	686	698	1684	1912	3437	3980	Average
Louvain	0.15	1.83	0.12	0.06	0.09	0.02	0.80	1.28	0.31	0.01	0.47
SIWO	0.34	3.78	0.31	0.16	0.17	0.03	1.46	3.79	0.51	0.02	1.06
CESNA	9.76	103.	6.02	2.47	3.12	0.63	38.3	22.9	21.1	0.60	20.8
Late-fusioin 1	0.72	4.68	0.40	0.25	0.24	0.07	1.95	3.83	0.78	0.03	1.30
Late-fusioin 2	2.90	20.0	0.82	0.48	0.44	0.08	8.22	9.41	3.28	0.06	4.57
Late-fusioin 3	1.73	24.4	2.87	0.68	0.76	0.14	5.76	28.5	4.26	0.12	6.92
Late-fusioin 4	9.45	91.4	5.27	1.73	3.14	0.34	44.9	43.4	13.5	0.17	21.3

In terms of NMI, experimental results in Table 4.8 show again that our late-fusion algorithms can significantly improve the community detection accuracy upon Louvain. On average, the late fusion of Louvain and unweighted virtual graph with equal-edge thresholding outperforms Louvain, SIWO and CESNA by 30.8%, 42.2% and 33.2% respectively. The late fusion of Louvain and weighted virtual graph with median thresholding outperforms the three by 14.1%, 24.0% and 16.2% respectively. However all of the methods perform poorly when evaluated by ARI. As for running time, we can see that again

Table 4.11: Ratio of number of communities detected to ground-truth on facebook network. Late-fusion 1 refers to Louvain + unweighted virtual graph with equal-edge thresholding, Late-fusion 2 refers to Louvain + weighted virtual graph with median thresholding, Late-fusion 3 refers to SIWO + unweighted virtual graph with equal-edge thresholding, and Late-fusion 4 refers to SIWO + weighted virtual graph with median thresholding.

	0	107	348	414	686	698	1684	1912	3437	3980	Average
Louvain	1.32	1.90	0.83	2.33	0.73	0.71	0.89	0.54	1.04	0.78	1.11
SIWO	1.23	2.40	0.50	2.17	0.55	0.64	0.94	0.61	1.04	0.61	1.07
CESNA	0.16	0.44	0.21	0.57	0.71	0.77	0.29	0.06	0.23	0.76	0.42
Late-fusioin 1	10.6	34.3	8.75	10.8	9.55	3.50	23.6	7.39	14.9	2.28	12.6
Late-fusioin 2	5.05	18.5	5.25	6.00	2.36	1.64	11.3	5.21	8.30	2.00	6.56
Late-fusioin 3	9.77	32.4	7.17	9.83	7.91	2.64	22.6	6.79	13.8	1.89	11.5
Late-fusioin 4	4.09	17.7	3.67	5.33	1.09	0.86	9.67	4.93	7.65	1.39	5.64

our late-fusion methods are very competive even compared with Louvain and SIWO. When comparing the ratio of number of communities detected to the ground truth, it is observed that CESNA is prone to merge small communities to larger ones, whereas late-fusion methods tend to over-partition a larger community into smaller communities. The two classic community detection algorithms are able to find the number of communities closer to the ground truth. Moreover, SIWO together with Late-fusion 3 and 4, two methods which have SIWO involved, are able to find more accurate number of communities than Louvain and Late-fuion 1, 2.

4.2.4 Effect of Parameter α

In the Sinanet experiment, we saw the superiority of having a weighting parameter to accordingly leverage the strength of the two sources of information. In this section we dive deeper into the effect of α to the community detection results. To do so, we devise an experiment where we use the graph 1 and 3 introduced in Table 4.1. Since graph 1 is "strong" in terms of structure, we assign weak attribute to it (set $std = 1.5$ in attribute redistribution). On the other hand, considering that the ground-truth communities of graph 4 has a relatively low modularity, we assign graph 4 with attribute that has conspicuous clusters in the feature space ($std = 0.3$). Then we perform our late

fusion algorithm on to these two graphs with varying αs. In our experiment we choose SIWO as F_s and k-means as F_a.

Table 4.12 and 4.13 present the NMI and ARI of late fusion with SIWO and spectral clustering when we vary α. Both NMI and ARI exhibit the same trend for both graph 1 and 4 when we go from the left to the right of the tables. Graph 1 has communities with strong structure and weak attribute, so the accuracy score for NMI and ARI goes up as we put more weight on the structure; On contrary, graph 4 has communities with weak structure and strong attribute, hence the accuracy score decreases as we increase α. We also notice that when α is sufficiently high or low, late fusion becomes equivalent to using community detection or clustering only, which is in accordance with our observation in the Sinanet experiment.

Table 4.12: Effect of α, evaluated by NMI

Graph ID	Only attribute	$\alpha = 0.2$	$\alpha = 0.5$	$\alpha = 0.8$	Only structure
1	0.530	0.530	0.756	0.836	0.836
4	0.867	0.867	0.762	0.526	0.526

Table 4.13: Effect of α, evaluated by ARI

Graph ID	Only attribute	$\alpha = 0.2$	$\alpha = 0.5$	$\alpha = 0.8$	Only structure
1	0.359	0.359	0.513	0.850	0.850
4	0.834	0.834	0.470	0.364	0.364

From the two tables we also notice that the late-fusion cannot outperform Louvain on graph 0, or k-means on graph 3. Therefore, when either of the structure or the attribute is strong enough to determine to communities, there is no need to consider both sources of information.

4.2.5 Complexity of Late Fusion

We use this section to discuss the time complexity and the scalability of our late-fusion method. The whole process of our method is comprised by two major components: the first is to get the two partitions $\mathcal{P}_s$ and $\mathcal{P}_a$, and the second is the fusion process. Therefore, the time complexity of our method is

also dependent on the time complexity of these two components. The complexity of the first components is determined by the chosen community detection algorihtm F_s and clustering algorithm F_a. For example, in our experiments, we choose Louvain or SIWO as F_s, both algorithms run in linear time with the network size. Suppose we pick the k-means algorithm as F_a, it has a complexity of $O(|V|ktr)$ where k is the number of clusters, t is the number of iterations, and r is the number of attributes. When k, t, r is relatively small as opposed to $|V|$, we can regard k-means as linear in the network size, too. As for the second component: getting the affiliation matrix requires to traverse all the nodes in the network, hence the time complexity of this step is $O(|V|)$. The later fusion step only contains matrix operations: a matrix multiplication to compute matrices D_s and D_a, and a matrix addition to get the adjacency matrix D of $G_{integrated}$. These matrix operations are scalable and can easily be parallelized by techniques such as MapReduce [29], like in the case of PageRank [71] for finding authoritative pages when ranking web search results. This means that our additional matrix operations do not increase the overall complexity of the community mining algorithm. After that, the community detection algorithm F_s is applied to $G_{integrated}$, where we use Louvain or SIWO. Therefore, the overall time complexity of the second component is linear in the size of the network. Combining the two components together, we can argue that the computational complexity of our proposed late-fusion method is dominated by the chosen F_s and F_a. Since in practice most of the options for F_s and F_a are very fast, our approach can easily scale up to large networks.

It is a known drawback of attributed community detection algorithms that they are very time-consuming due to the needs to consider node attributes. Our late-fusion method tries to circumvent this problem by taking advantage of the existing community detection and clustering algorithms and combines their results by a simple approach. Next we conduct an experiment to illustrate the running time of the late-fusion method compared to other methods.

We test the running time of four different community detection methods at five graphs with the number of nodes varying from 2000, 4000, 6000, 8000, and

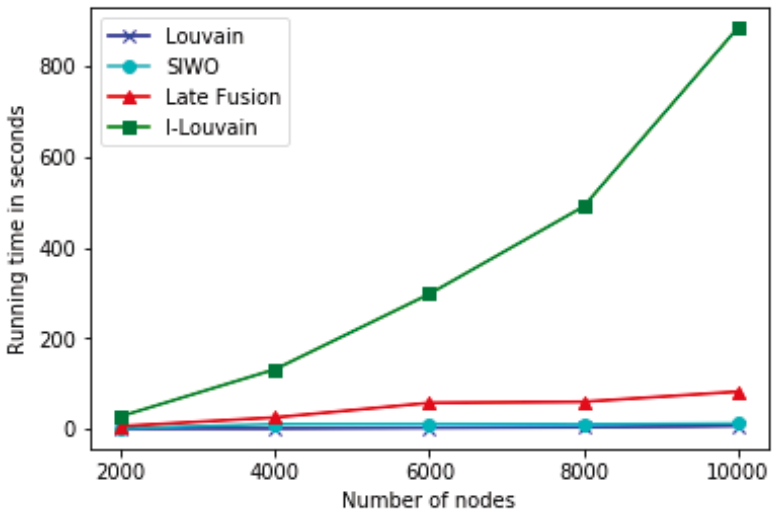


Figure 4.2: Running time of Louvain, SIWO, Late Fusion and I-Louvain on networks of different sizes

10000. These graphs are also generated by the attributed graph generator that we used in the synthetic network experiment. We control the modularity of each graph at the range of $0.64 - 0.66$, and perform attribute redistribution at $std = 0.5$, to avoid confounding factors. For each size, we generate 10 different networks with the same parameter setting and plot the average running time of each method. As we can see in Figure 4.2, it is expected that our late-fusion method is inevitably slower than the two community detection methods that only utilize node connections. However our algorithm runs way faster than the I-Louvain algorithm, albeit both being approximately linear in the growth of network sizes.

4.3 Late Fusion via Consensus Clustering

This section is not considered as a major component of the methodology that we propose, however we have spent a large amount of time exploring in this direction. We would like to present our work on consensus clustering as to demonstrate what we have learned.

Most clustering and community detection algorithms are indeterministic, i.e., if we change the order of nodes that are being processed, we might end up

with a slightly different clustering result. Consensus clustering [28], [89] provides a way to take advantage of the inderterministic property of a clustering algorithm and produce a consensus, more stronger partitioning. Starting with a graph G and a community detection/clustering algorithm F, Fortunato and Hric [35] describe consensus clustering in the following steps:

1. Apply F on G k times, yielding k partitionings.
2. Compute the consensus matrix D, D_{ij} is the number of partitionings in which nodes i and j are assigned to the same community, divided by k.
3. All entries of D below a chosen threshold t are set to zero.
4. Apply F on D k times, yielding k partitionings.
5. If all partitionings are equal, stop. Otherwise go gack to step 2.

Consensus clustering described above can also be translated to serve our purpose. We can use consensus clustering to combine the two sets of communities detected from node connections and attributes. Let F_s be the community detection algorithm applied to the graph structure, and F_a be the clustering algorithm applied to the node attribute matrix A. The adjusted version of consensus clustering for attributed community detection is as follows:

1. Apply F_s on G and F_a on A k times, yielding $2k$ partitionings.
2. For the first k partitionings that are based on G, compute the consensus matrix D_s: each entry D_{ij} is the number of partitions in which nodes i and j of G_s are assigned to the same community, divided by k. Do the same for the second k partitionings and obtain the corresponding consensus matrix D_a.
3. The integrated consensus matrix is $D = \alpha D_s + (1 - \alpha)D_a$ where α is a weighting parameter between 0 and 1.
4. All entries of D below a certain threshold t are set to zero.

5. Apply community detection algorithm F_s to D $2k$ times, yielding $2k$ partitions.

6. If the partitionings are equal, stop. Otherwise re-compute the integrated consensus matrix D: D_{ij} is the number of partitions in which nodes i and j of D are assigned to the same community, divided by $2k$. And go back to step 4.

We call this approach "late fusion via consensus clustering". To concretely show how it works, we apply this approach to the same four graphs as in Table 4.1, with the same experiment settings as that of the first group of the synthetic network experiments, whose results correspond to Table 4.4. We select Louvain and SIWO as the F_s, spectral clustering and DBSCAN as the F_a. The parameter k described above, which is the number of times we apply an algorithm to the graph under study, is set to 5. This is the ususal option of consensus clustering in practice. The results of NMI, ARI and running time (in seconds) are displayed in Table 4.14

Table 4.14: Late fusion without consensus clustering (CC) vs with CC. Time is measured in seconds.

	Graph 1			*Graph* 2			*Graph* 3			*Graph* 4		
	NMI	ARI	time	NMI	ARI	time	NMI	ARI	time	NMI	ARI	time
Without CC:												
Louvain + SC	.824	.704	7.34	.790	.650	6.68	.784	.618	5.74	.765	.597	7.14
Louvain + DBSCAN	.818	.813	8.64	.742	.720	8.59	.730	.702	8.87	.704	.690	10.6
SIWO + SC	.844	.738	10.3	.806	.689	9.28	.786	.636	7.33	.723	.508	6.46
SIWO + DBSCAN	.818	.813	11.7	.742	.720	10.5	.730	.702	10.2	.704	.690	11.6
With CC:												
Louvain + SC	.830	.770	277	.750	.626	539	.763	.691	320	.816	.755	311
Louvain + DBSCAN	.563	.306	561	.683	.645	398	.654	.524	516	.461	.327	638
SIWO + SC	.772	.633	1627	.720	.592	1883	.721	.595	2071	.000	.000	4310
SIWO + DBSCAN	.384	.145	6111	.000	.000	15302	.274	.045	10106	.000	.000	3845

The upper four rows of Table 4.14 are copied from the last four rows of Table 4.4, which show the results of the late fusion method that we proposed in Chapter 3, i.e., without consensus clustering. The lower four rows present the same experiment on the same set of graphs, except that we use the consensus clustering method described above. As we can see, in general, consensus

clustering deteriorates the accuracy from the late fusion method that we proposed earlier. More importantly, consensus clustering renders the application of community detection computationally expensive. This is due to the fact that community detection and clustering are utilizing two totally different sources of information of the network, hence they produce very different partitionings. The large difference gaps between the partitionings make the convergence to the consensus painfully slow, and can easily go astray. Therefore we forgo pursuing under this direction and recommend the late fusion approach that we proposed in Chapter 3.

Chapter 5

Conclusion

5.1 Summary

In this thesis we provide our work of community detection on attributed networks. We propose a new approach that follows a late-fusion manner. Concretely, for networks with numeric attribute, we obtain two sets of communities by applying a community detection algorithm to the original graph and a clustering algorithm to the attribute matrix. For networks with binary/categorical attribute, we devise a way to create a virtual graph G_a based upon the node similarity in the original graph G. Hence we can get the communities that reflect the clustering of node attributes by applying a community detection algorithm on the virtual graph. Later we combine the two partitions using the fusion method we proposed in Chapter 3, which results in a new set of communities that fulfill our goal of community mining.

We show with extensive experiments that our late-fusion method provides a simple yet powerful approach to the attributed community mining tasks on different types of networks. We conclude our work as follows.

1. We proposed a new late-fusion framework that takes advantage of existing community detection and clustering algorithms and is capable of finding communities with strong node connectivity and attribute homogeneity. The flexibility of the framework enables people to choose from abundant combinations of algorithms and to pick the one that works the best.

2. We designed a novel method to identify communities that reflect node similarity for networks with binary attributes. Our method overcomes the difficulty of applying traditional clustering algorithms to binary features, by generating a virtual network based on the attribute similarity between nodes. This new approach can also be regarded as a new clustering algorithm for data samples with binary or categorical features.

3. We applied the late-fusion framework on 4 synthetic networks and 11 real networks of different sizes and attribute types, and evaluated the outcome communities against ground-truth communities. We showed that on both synthetic and real networks, with numeric or binary attributes, our method consistently gives good results and significantly improves the quality of communities that are identified solely based on node connections.

4. We introduced a weighting parameter α and empirically studied its effect. We showed that the weight α leverages the contribution from structure and attribute to the final communities. It allows for human-control of the strength of structure and attributes in the detection of communities, which brings extra power when prior knowledge about the formation of communities is known.

5. We analyzed the complexity of our method and proves the linear growth of our method with the size of the network, given that the community detection and clustering algorithms employed in late fusion are also linear in the size of network. In practice it is way more efficient than other attributed community detection algorithms such as I-Louvain and CESNA.

5.2 Future Directions

We would like to continue our efforts on the problem of community detection on attributed networks. Especially, based on our experiments, we summarize the following several directions in which we can push forward our study:

1. **Hybrid attributes**: A network can have a hybrid collection of attributes, and these attributes are usually closely correlated as well. For example, the attributes of an online social network member can include salary (numeric), occupation (categorical) and his/her political view (binary). A possible extension of our late-fusion algorithm to networks with hybrid attributes is to convert the numerical attributes in the network to ordinal, i.e, categorical attributes.

2. **Effect of structure and attribute in the formation of network communities** : A much deeper and more thorough study on the contribution of structure and attribute to the formation of network communities is desired. In particular, when the "homophily" effect is no longer significant in the network understudy, we need to adjust our fusion approach and choose the optimal α accordingly. Understanding the interactive effect between node connections and attributes can shed light on how we should appropriately utilize them. Ultimately, this study will lead to more accurate detection algorithm.

3. **Overlapping communities**: So far our late-fusion approach is only capable of detecting non-overlapping communities, due to the detection algorithm F_s in the experiments can only produce a partitioning of the node set V. A starting point towards overlapping community detection could be replacing Louvain and SIWO with algorithms that allow overlaps. Several algorithms able to find overlapping communities are, for instance, clique percolation approach [72], and the method based on overlapping subgraphs [8].

References

[1] L. A. Adamic and B. A. Huberman, "Power-law distribution of the world wide web," *Science*, vol. 287, no. 5461, pp. 2115–2115, 2000. 1

[2] L. A. Adamic, R. M. Lukose, A. R. Puniyani, and B. A. Huberman, "Search in power-law networks," *Physical review E*, vol. 64, no. 4, p. 046 135, 2001. 1

[3] R. Agrawal and H. Jagadish, "Algorithms for searching massive graphs," *IEEE Transactions on Knowledge and Data Engineering*, vol. 6, no. 2, pp. 225–238, 1994. 3

[4] L. Akoglu, H. Tong, B. Meeder, and C. Faloutsos, "Pics: Parameter-free identification of cohesive subgroups in large attributed graphs," in *Proceedings of the 2012 SIAM international conference on data mining*, SIAM, 2012, pp. 439–450. 24

[5] Y. Asim, R. Ghazal, W. Naeem, A. Majeed, B. Raza, and A. K. Malik, "Community detection in networks using node attributes and modularity," *INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS*, vol. 8, no. 1, pp. 382–388, 2017. 25

[6] R. Balasubramanyan and W. W. Cohen, "Block-lda: Jointly modeling entity-annotated text and entity-entity links," in *Proceedings of the 2011 SIAM International Conference on Data Mining*, SIAM, 2011, pp. 450–461. 31

[7] E. R. Barnes, "An algorithm for partitioning the nodes of a graph," *SIAM Journal on Algebraic Discrete Methods*, vol. 3, no. 4, pp. 541–550, 1982. 13

[8] J. Baumes, M. K. Goldberg, M. S. Krishnamoorthy, M. Magdon-Ismail, and N. Preston, "Finding communities by clustering a graph into overlapping subgraphs.," *IADIS AC*, vol. 5, pp. 97–104, 2005. 69

[9] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003. 44

[10] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of statistical mechanics: Theory and experiment*, vol. 2008, no. 10, P10008, 2008. 8, 15, 25, 40, 51

[11] B. Bollobás, *Modern graph theory*. Springer Science & Business Media, 2013, vol. 184. 38

[12] M. P. Boobalan, D. Lopez, and X. Z. Gao, "Graph clustering using k-neighbourhood attribute structural similarity," *Applied Soft Computing*, vol. 47, pp. 216–223, 2016. 51

[13] S. P. Borgatti and M. G. Everett, "Notions of position in social network analysis," *Sociological methodology*, pp. 1–35, 1992. 12

[14] U. Brandes, D. Delling, M. Gaertler, R. Gorke, M. Hoefer, Z. Nikoloski, and D. Wagner, "On modularity clustering," *IEEE transactions on knowledge and data engineering*, vol. 20, no. 2, pp. 172–188, 2008. 15

[15] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, "API design for machine learning software: Experiences from the scikit-learn project," in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 2013, pp. 108–122. 54

[16] B.-f. Chai, J. Yu, C.-y. Jia, T.-b. Yang, and Y.-w. Jiang, "Combining a popularity-productivity stochastic block model with a discriminative-content model for general structure detection," *Physical review E*, vol. 88, no. 1, p. 012 807, 2013. 31

[17] J. Chang and D. Blei, "Relational topic models for document networks," in *Artificial Intelligence and Statistics*, 2009, pp. 81–88. 31

[18] J. Chen and B. Yuan, "Detecting functional modules in the yeast protein–protein interaction network," *Bioinformatics*, vol. 22, no. 18, pp. 2283–2290, 2006. 1

[19] J. Chen, O. R. Zaıane, and R. Goebel, "An unsupervised approach to cluster web search results based on word sense communities," in *Web Intelligence and Intelligent Agent Technology, 2008. WI-IAT'08. IEEE/WIC/ACM International Conference on*, IEEE, vol. 1, 2008, pp. 725–729. 3

[20] Y. Chen, X. Wang, J. Bu, B. Tang, and X. Xiang, "Network structure exploration in networks with node attributes," *Physica A: Statistical Mechanics and its Applications*, vol. 449, pp. 240–253, 2016. 31

[21] D. Combe, C. Largeron, E. Egyed-Zsigmond, and M. Géry, "Combining relations and text in scientific network clustering," in *Proceedings of the 2012 International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012)*, IEEE Computer Society, 2012, pp. 1248–1253. 27

[22] D. Combe, C. Largeron, M. Géry, and E. Egyed-Zsigmond, "I-louvain: An attributed graph clustering method," in *International Symposium on Intelligent Data Analysis*, Springer, 2015, pp. 181–192. 25, 51

[23] J. D. Cruz and C. Bothorel, "Information integration for detecting communities in attributed graphs," in *Computational Aspects of Social Networks (CASoN), 2013 Fifth International Conference on*, IEEE, 2013, pp. 62–67. 35, 36, 41

[24] J. D. Cruz, C. Bothorel, and F. Poulet, "Entropy based community detection in augmented social networks," in *Computational aspects of social networks (cason), 2011 international conference on*, IEEE, 2011, pp. 163–168. 24

[25] ——, "Semantic clustering of social networks using points of view.," in *CORIA*, 2011, pp. 175–182. 26

[26] T. Dang and E. Viennet, "Community detection based on structural and attribute similarities," in *International conference on digital society (icds)*, 2012, pp. 7–14. 25

[27] J.-J. Daudin, F. Picard, and S. Robin, "A mixture model for random graphs," *Statistics and computing*, vol. 18, no. 2, pp. 173–183, 2008. 31

[28] R. Dazeley, J. L. Yearwood, B. H. Kang, and A. V. Kelarev, "Consensus clustering and supervised classification for profiling phishing emails in internet commerce security," in *Pacific Rim Knowledge Acquisition Workshop*, Springer, 2010, pp. 235–246. 64

[29] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008. 7, 62

[30] Y. Dourisboure, F. Geraci, and M. Pellegrini, "Extraction and classification of dense communities in the web," in *Proceedings of the 16th international conference on World Wide Web*, ACM, 2007, pp. 461–470. 1

[31] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise.," in *Kdd*, vol. 96, 1996, pp. 226–231. 8, 22

[32] G. W. Flake, S. Lawrence, C. L. Giles, and F. M. Coetzee, "Self-organization and identification of web communities," *Computer*, vol. 35, no. 3, pp. 66–70, 2002. 1

[33] S. Fortunato, "Community detection in graphs," *Physics reports*, vol. 486, no. 3, pp. 75–174, 2010. 19

[34] S. Fortunato and M. Barthelemy, "Resolution limit in community detection," *Proceedings of the National Academy of Sciences*, vol. 104, no. 1, pp. 36–41, 2007. 16, 17

[35] S. Fortunato and D. Hric, "Community detection in networks: A user guide," *Physics Reports*, vol. 659, pp. 1–44, 2016. 11, 19, 64

[36] S. Z. Gharaghooshi, "Community structure in complex networks," Master's thesis, University of Alberta, Nov. 2018. 8, 17, 51

[37] M. Girvan and M. E. Newman, "Community structure in social and biological networks," *Proceedings of the national academy of sciences*, vol. 99, no. 12, pp. 7821–7826, 2002. 1, 18

[38] A. Grover and J. Leskovec, "Node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2016, pp. 855–864. 33, 44

[39] P. W. Holland, K. B. Laskey, and S. Leinhardt, "Stochastic blockmodels: First steps," *Social networks*, vol. 5, no. 2, pp. 109–137, 1983. 19

[40] L. Hubert and P. Arabie, "Comparing partitions," *Journal of classification*, vol. 2, no. 1, pp. 193–218, 1985. 36, 37

[41] D. R. Hunter and K. Lange, "A tutorial on mm algorithms," *The American Statistician*, vol. 58, no. 1, pp. 30–37, 2004. 35

[42] C. Jia, Y. Li, M. B. Carson, X. Wang, and J. Yu, "Node attribute-enhanced community detection in complex networks," *Scientific Reports*, vol. 7, 2017. 27, 28, 44, 49, 56

[43] L. Kaufman and P. J. Rousseeuw, *Finding groups in data: An introduction to cluster analysis*. John Wiley & Sons, 2009, vol. 344. 20

[44] B. W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *The Bell system technical journal*, vol. 49, no. 2, pp. 291–307, 1970. 13

[45] M. Kim and J. Leskovec, "Multiplicative attribute graph model of real-world networks," *Internet mathematics*, vol. 8, no. 1-2, pp. 113–160, 2012. 5

[46] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990. 26, 36, 41

[47] G. Kossinets and D. J. Watts, "Empirical analysis of an evolving social network," *Science*, vol. 311, no. 5757, pp. 88–90, 2006. 5

[48] T. La Fond and J. Neville, "Randomization tests for distinguishing social influence and homophily effects," in *Proceedings of the 19th international conference on World wide web*, ACM, 2010, pp. 601–610. 5

[49] C. Largeron, P.-N. Mougel, R. Rabbany, and O. R. Zaıane, "Generating attributed networks with communities," *PloS one*, vol. 10, no. 4, e0122777, 2015. 48, 52

[50] P. F. Lazarsfeld, R. K. Merton, *et al.*, "Friendship as a social process: A substantive and methodological analysis," *Freedom and control in modern society*, vol. 18, no. 1, pp. 18–66, 1954. 5

[51] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney, "Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters," *Internet Mathematics*, vol. 6, no. 1, pp. 29–123, 2009. 13

[52] J. Leskovec and J. J. Mcauley, "Learning to discover social circles in ego networks," in *Advances in neural information processing systems*, 2012, pp. 539–547. 49

[53] H. Li, Z. Nie, W.-C. Lee, L. Giles, and J.-R. Wen, "Scalable community discovery on textual data with relations," in *Proceedings of the 17th ACM conference on Information and knowledge management*, ACM, 2008, pp. 1203–1212. 30

[54] Y. Li, C. Jia, and J. Yu, "A parameter-free community detection method based on centrality and dispersion of nodes in complex networks," *Physica A: Statistical Mechanics and its Applications*, vol. 438, pp. 321–334, 2015. 28

[55] Y. Li, C. Sha, X. Huang, and Y. Zhang, "Community detection in attributed graphs: An embedding approach," in *AAAI*, 2018. 33

[56] Y. Liu, A. Niculescu-Mizil, and W. Gryc, "Topic-link lda: Joint models of topic and author community," in *Proceedings of the 26th annual international conference on machine learning*, ACM, 2009, pp. 665–672. 31

[57] S. Lloyd, "Least squares quantization in pcm," *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982. 8, 21

[58] F. Lorrain and H. C. White, "Structural equivalence of individuals in social networks," *The Journal of mathematical sociology*, vol. 1, no. 1, pp. 49–80, 1971. 12

[59] R. D. Luce, "Connectivity and generalized cliques in sociometric group structure," *Psychometrika*, vol. 15, no. 2, pp. 169–190, 1950. 12

[60] R. D. Luce and A. D. Perry, "A method of matrix analysis of group structure," *Psychometrika*, vol. 14, no. 2, pp. 95–116, 1949. 11

[61] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, Oakland, CA, USA, vol. 1, 1967, pp. 281–297. 19, 21

[62] M. McPherson, L. Smith-Lovin, and J. M. Cook, "Birds of a feather: Homophily in social networks," *Annual review of sociology*, vol. 27, no. 1, pp. 415–444, 2001. 5

[63] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *ArXiv preprint arXiv:1301.3781*, 2013. 33

[64] R. J. Mokken, "Cliques, clubs and clans," *Quality and quantity*, vol. 13, no. 2, pp. 161–173, 1979. 12

[65] J. Neville, M. Adler, and D. Jensen, "Clustering relational data using attribute and link information," in *Proceedings of the text mining and link analysis workshop, 18th international joint conference on artificial intelligence*, San Francisco, CA: Morgan Kaufmann Publishers, 2003, pp. 9–15. 26

[66] M. E. Newman, "Assortative mixing in networks," *Physical review letters*, vol. 89, no. 20, p. 208 701, 2002. 1

[67] ——, "Modularity and community structure in networks," *Proceedings of the national academy of sciences*, vol. 103, no. 23, pp. 8577–8582, 2006. 25

[68] M. E. Newman and A. Clauset, "Structure and inference in annotated networks," *Nature Communications*, vol. 7, p. 11 863, 2016. 31

[69] M. E. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical review E*, vol. 69, no. 2, p. 026 113, 2004. 12, 14, 18

[70] M. E. Newman and E. A. Leicht, "Mixture models and exploratory analysis in networks," *Proceedings of the National Academy of Sciences*, vol. 104, no. 23, pp. 9564–9569, 2007. 31

[71] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web.," Stanford InfoLab, Tech. Rep., 1999. 62

[72] G. Palla, I. Derényi, I. Farkas, and T. Vicsek, "Uncovering the overlapping community structure of complex networks in nature and society," *Nature*, vol. 435, no. 7043, p. 814, 2005. 69

[73] B. Perozzi, L. Akoglu, P. Iglesias Sánchez, and E. Müller, "Focused clustering and outlier detection in large attributed graphs," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2014, pp. 1346–1355. 30

[74] P. Pons and M. Latapy, "Computing communities in large networks using random walks.," *J. Graph Algorithms Appl.*, vol. 10, no. 2, pp. 191–218, 2006. 16

[75] R. Rabbany and O. R. Zaïane, "Evaluation of community mining algorithms in the presence of attributes," in *Trends and Applications in Knowledge Discovery and Data Mining*, Springer, 2015, pp. 152–163. 36

[76] ——, "A general clustering agreement index: For comparing disjoint and overlapping clusters.," in *AAAI*, 2017, pp. 2492–2498. 38

[77] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi, "Defining and identifying communities in networks," *Proceedings of the National Academy of Sciences*, vol. 101, no. 9, pp. 2658–2663, 2004. 18

[78] L. F. Ribeiro, P. H. Saverese, and D. R. Figueiredo, "Struc2vec: Learning node representations from structural identity," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2017, pp. 385–394. 33

[79] A. W. Rives and T. Galitski, "Modular organization of cellular networks," *Proceedings of the National Academy of Sciences*, vol. 100, no. 3, pp. 1128–1133, 2003. 1

[80] M. Rosvall and C. T. Bergstrom, "Maps of random walks on complex networks reveal community structure," *Proceedings of the National Academy of Sciences*, vol. 105, no. 4, pp. 1118–1123, 2008. 16, 40

[81] Y. Ruan, D. Fuhry, and S. Parthasarathy, "Efficient community detection in large networks using content and links," in *Proceedings of the 22nd international conference on World Wide Web*, ACM, 2013, pp. 1089–1098. 27

[82] M. T. Schaub, J.-C. Delvenne, S. N. Yaliraki, and M. Barahona, "Markov dynamics as a zooming lens for multiscale community detection: Non clique-like communities and the field-of-view limit," *PloS one*, vol. 7, no. 2, e32210, 2012. 17

[83] V. Spirin and L. A. Mirny, "Protein complexes and functional modules in molecular networks," *Proceedings of the National Academy of Sciences*, vol. 100, no. 21, pp. 12 123–12 128, 2003. 1

[84] M. Steenstrup, "Cluster-based networks," in *Ad hoc networking*, Addison-Wesley Longman Publishing Co., Inc., 2001, pp. 75–138. 3

[85] K. Steinhaeuser and N. V. Chawla, "Community detection in a large real-world social network," in *Social computing, behavioral modeling, and prediction*, Springer, 2008, pp. 168–175. 26

[86] A. Strehl and J. Ghosh, "Cluster ensembles—a knowledge reuse framework for combining multiple partitions," *Journal of machine learning research*, vol. 3, no. Dec, pp. 583–617, 2002. 36

[87] P. R. Suaris and G. Kedem, "An algorithm for quadrisection and its application to standard cell placement," *IEEE Transactions on Circuits and Systems*, vol. 35, no. 3, pp. 294–303, 1988. 13

[88] Y. Sun, C. C. Aggarwal, and J. Han, "Relation strength-aware clustering of heterogeneous information networks with incomplete attributes," *Proceedings of the VLDB Endowment*, vol. 5, no. 5, pp. 394–405, 2012. 31

[89] A. Topchy, A. K. Jain, and W. Punch, "Clustering ensembles: Models of consensus and weak partitions," *IEEE transactions on pattern analysis and machine intelligence*, vol. 27, no. 12, pp. 1866–1881, 2005. 64

[90] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007. 18

[91] J. H. Ward Jr, "Hierarchical grouping to optimize an objective function," *Journal of the American statistical association*, vol. 58, no. 301, pp. 236–244, 1963. 16

[92] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world'networks," *Nature*, vol. 393, no. 6684, p. 440, 1998. 17

[93] D. S. Wilks, *Statistical methods in the atmospheric sciences*. Academic press, 2011, vol. 100. 19

[94] J. Xie, S. Kelley, and B. K. Szymanski, "Overlapping community detection in networks: The state-of-the-art and comparative study," *Acm computing surveys (csur)*, vol. 45, no. 4, p. 43, 2013. 13

[95] Z. Xu, Y. Ke, Y. Wang, H. Cheng, and J. Cheng, "A model-based approach to attributed graph clustering," in *Proceedings of the 2012 ACM SIGMOD international conference on management of data*, ACM, 2012, pp. 505–516. 31

[96] ——, "Gbagc: A general bayesian framework for attributed graph clustering," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 9, no. 1, p. 5, 2014. 31

[97] J. Yang and J. Leskovec, "Overlapping community detection at scale: A nonnegative matrix factorization approach," in *Proceedings of the sixth ACM international conference on Web search and data mining*, ACM, 2013, pp. 587–596. 32

[98] ——, "Defining and evaluating network communities based on ground-truth," *Knowledge and Information Systems*, vol. 42, no. 1, pp. 181–213, 2015. 38

[99] J. Yang, J. McAuley, and J. Leskovec, "Community detection in networks with node attributes," in *Data Mining (ICDM), 2013 IEEE 13th international conference on*, IEEE, 2013, pp. 1151–1156. 31, 32, 51, 58

[100] T. Yang, Y. Chi, S. Zhu, Y. Gong, and R. Jin, "Directed network community detection: A popularity and productivity link model," in *Proceedings of the 2010 SIAM International Conference on Data Mining*, SIAM, 2010, pp. 742–753. 31

[101] T. Yang, R. Jin, Y. Chi, and S. Zhu, "Combining link and content for community detection: A discriminative approach," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2009, pp. 927–936. 31

[102] T. Yano, W. W. Cohen, and N. A. Smith, "Predicting response to political blog posts with topic models," in *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, Association for Computational Linguistics, 2009, pp. 477–485. 43

[103] H. Zanghi, S. Volant, and C. Ambroise, "Clustering based on random graph model embedding vertex features," *Pattern Recognition Letters*, vol. 31, no. 9, pp. 830–836, 2010. 31

[104] Y. Zhou, H. Cheng, and J. X. Yu, "Graph clustering based on structural/attribute similarities," *Proceedings of the VLDB Endowment*, vol. 2, no. 1, pp. 718–729, 2009. 27, 28

[105] ——, "Clustering large attributed graphs: An efficient incremental approach," in *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, IEEE, 2010, pp. 689–698. 27, 29