A Tutorial on the Probabilistic Framework for Centrality Analysis, Community Detection and Clustering

Cheng-Shang Chang

All rights reserved

Abstract

Analysis of networked data has received a lot of attention lately from many researchers in various fields, including physicists, mathematicians, computer scientists, biologists, and sociologists. As researchers from various research disciplines, they have different viewpoints and thus it is important to have a unified framework for the analysis of networked data. For this, we introduce in the tutorial the recently developed probabilistic framework for the analysis of networked data, including centrality analysis, community detection and clustering. The key insight of this probabilistic framework is to model networked data as a graph and sample the graph to obtain a probability measure for further analysis. In particular, the centrality of a node can be defined by the marginal probability that this node is sampled and a community can be defined as a set of nodes that are more likely to be sampled together. Each sampling method provides a unique viewpoint to a graph and that leads a unique definition of centrality and a unique definition of community for that graph.

In this probabilistic framework, the modularity of a partition of a graph is defined as the average community strength of a randomly selected node. As such, the community detection problem that aims to find a good partition of a graph can be tackled by looking for algorithms that yield large modularity. In this tutorial, we introduce four commonly used modularity maximization algorithms for community detection: (i) the spectral modularity maximization algorithm, (ii) the hierarchical agglomerative algorithm, (iii) the partitional algorithm, and (iv) the fast unfolding algorithm.

Clustering problems and community detection problems are closely related. For a clustering problem, there is a set of data points (or objects) and a similarity (or dissimilarity) measure that measures how similar two data points are. The aim of a clustering algorithm is to cluster these data points so that data points within the same cluster are similar to each other and data points in different clusters are dissimilar. Our approach for clustering is to embed data points into a graph so that it can be treated as a community detection problem. In this tutorial, we introduce the exponentially twisted sampling technique to embed data points in a metric (resp. semi-metric) space to a graph. By doing so, the community detection algorithms based on modularity maximization can also be applied to clustering data points in a metric (resp. semi-metric) space. Unlike the other modularity maximization algorithms, the K-sets algorithm (resp. the K-sets$^+$ algorithm) is a partitional algorithm that clusters data points in a metric (resp. semi-metric) space to maximizes the normalized modularity. The K-sets algorithm converges in the same way as a sequential version of the classical kernel K-means algorithm. However, the key difference is that its input matrix does not need to be positive semi-definite.

keywords: centrality analysis, PageRank, community detection, modularity, K-means, spectral clustering

C.-S. Chang is with the Institute of Communications Engineering, National Tsing Hua University, Hsinchu 30013, Taiwan, R.O.C. Email: cschang@ee.nthu.edu.tw.

Contents

1 Overview

Analysis of networked data has received a lot of attention lately from many researchers in various fields, including physicists, mathematicians, computer scientists, biologists, and sociologists. As these researchers are from different fields and published their papers in different areas of journals, the terminology and the viewpoints of their works could be quite different. In particular, physicists put more emphasis on the discovery of new physical insights such as phase transition and percolation, while mathematicians stress the importance of the mathematical proofs for these newly discovered phenomenons. For computer scientists, the efficiency, in terms of computational complexity, of the algorithms that lead to these new findings is the main interest of their research. On the other hand, biologists and sociologists apply the algorithms to the networks from their data and seek meaningful explanations for their findings. In order for researchers from various research disciplines to carry out their scientific research, it is thus essential to have a *framework* for the analysis of networked data [1].

The aim of this tutorial is to introduce a unified framework for the analysis of networked data, including centrality analysis, community detection and clustering. Centrality analysis [2], [3] has been one of the most important research topics in social network analysis. As networked data are commonly modelled by graphs, there are various notions of centralities defined for ranking the importance of nodes in a graph, including the degree centrality, the eigenvector centrality, the Katz centrality, PageRank, the closeness centrality and the betweenness centrality (see e.g., the book [4]). Among them, PageRank [5], proposed by Google, is perhaps one of the most famous centrality measures for ranking web pages. The key idea behind PageRank is to model the behavior of a web surfer by a random walk (the random surfer model) and then use that to sample the probability for a web surfer to visit a specific web page. The higher the probability for a web surfer to visit a specific web page is, the more important that web page is.

The goal of *community detection* is to find a partition of a graph so as to discover and understand the large-scale structure of a network. However, people have different opinions on what a *good community* should look like. There are several notions for this in the literature: (i) a good community should have more edges within the community than the edges going outside the community (see e.g., [6], [7]) and thus *conductance* might be a good measure (see e.g., [8], [9], [10]), (ii) a good community should be well connected and thus it should be dense [11] and have a high clustering coefficient [12] or lots of k-cliques [13], (iii) a graph with a good community structure should behave quite differently from random graphs and thus modularity and the null model [14], [15] can be used for measuring how well a graph is partitioned, (iv) a good community should be cohesive and cannot be easily divided into disconnected components [16], [17], [18], (v) a good community should have a high probability to trap a random walker inside the community for a certain period of time and thus one can use either data compression techniques for describing the path of the random walker inside the community [19], [20] or stability [21], [22] for measuring how well a graph is partitioned, and (vi) rumors are spread fast within a good community [23]. Based on these notions of communities, many algorithms (see the review papers in [11] and [24]) have been developed in the literature and they might be classified as follows: (i) divisive algorithms (betweenness, spectral partitioning, sweep) [25], [6], [26], [27], [28], [18], (ii) agglomerative algorithms [14], [29], [30], (iii) statistic and machine learning methods (spectral learning [31], kernel-based clustering algorithms [32], [33], exponential families [34], [35]), (iv) data compression algorithms [19], [20] and (v) clique percolation methods [13]. Various comparison studies of these algorithms can be found in [36], [37], [17].

Clustering is one of the most fundamental problems in data analysis and it has a lot of applications in various fields, including Internet search for information retrieval, social network analysis for community detection, and computation biology for clustering protein sequences. The problem of clustering has been studied extensively in the literature (see e.g., the books [38], [39] and the historical review papers [40], [41]). For a clustering problem, there is a set of data points (or objects) and a similarity (or dissimilarity) measure that measures how similar two data points are. The aim of a clustering algorithm is to cluster these data points so that data points within the same cluster are similar to each other and data points in different clusters are dissimilar.

Clustering problems and community detection problems are closely related. In some sense, community detection can be viewed as a clustering problem when data points are in a graph. On the other hand, one can also transform a clustering problem into a community detection problem by embedding data points into a graph. Like community detection algorithms, clustering algorithms can also be divided into two groups: *hierarchical* and *partitional*. Hierarchical algorithms can further be divided into two subgroups: *agglomerative* and *divisive*. Agglomerative hierarchical algorithms, starting from each data point as a sole cluster, recursively merge two *similar* clusters into a new cluster. On the other hand, divisive hierarchical algorithms, starting from the whole set as a single cluster, recursively divide a cluster into two *dissimilar* clusters. As such, there is a hierarchical structure of clusters from either a hierarchical agglomerative clustering algorithm or a hierarchical divisive clustering algorithm. On the other hand, partitional algorithms do not have a hierarchical structure of clusters. Instead, they find all the clusters as a partition of the data points. The K-means algorithm is perhaps the simplest and the most widely used partitional algorithm for data points in a Euclidean space, where the Euclidean distance serves as the natural dissimilarity measure and the centroid of a cluster is selected to represent a cluster. For data points that are in a non-Euclidean space, the K-medoids algorithm (see e.g., [42], [38], [43], [44]) is used as a refinement of the K-mean algorithm. Both the K-means algorithm and the K-medoids algorithm are quite sensitive to the initial choice of the partition. There are some recent works that provide various methods for selecting the initial partition that might lead to performance guarantees [45], [46], [47], [48], [49]. Instead of using the iterative method as in the K-means algorithm, one can also formulate a clustering problem as an optimization problem with respect to a certain objective function and then solve the optimization problem by other methods. This then leads to kernel and spectral clustering methods (see e.g., [50], [51], [52], [53], [54] and [55], [56] for reviews of the papers in this area). Solving the optimization problems formulated from the clustering problems are in general NP-hard and one has to resort to approximation algorithms [57]. In [57], Balcan et al. introduced the concept of approximation stability that assumes all the partitions (clusterings) that have the objective values close to the optimum ones are close to the target partition. Under such an assumption, they proposed efficient algorithms for clustering large datasets.

In Section 2, we first introduce the *probabilistic framework* in our previous works [58], [59] for centrality analysis and community detection for networked data. In social network analysis, centralities [2], [3], [4] have been widely used for ranking the importance of nodes. For instance, movie stars who have a lot of fans can be easily identified as important nodes in social networks by using the degree centrality. However, we generally do not consider movie stars important persons *to us*. On the contrary, family members or friends are much more important *to us*. In view of this, we extend the concept of *centrality* and incorporate the concept of *relative centrality* into the structural analysis of networks. In the probabilistic framework, relative centrality is a (probability) measure that measures how important a set of nodes in a network is with respect to another set of nodes, and it is a generalization of *centrality* that only measures (or ranks) how important a set of nodes in a network is with respect to *the whole set of nodes in the network*. A set (of nodes) that has a much larger relative centrality with respect to itself than its centrality can thus be viewed as a *community*.

People have different views. As such, centrality and community can only be formally defined on top of a specific viewpoint. To obtain a viewpoint of a network, one typical method is to "sample" the network, e.g., edge sampling, random walks, diffusion [15], or random gossiping [23]. Mathematically, each sampling method renders a (probability) measure for a network that enables us to carry out further analysis. In the probabilistic framework, we model a network as a graph G and "sample" the graph to generate a bivariate distribution $p(\cdot,\cdot)$ for a pair of two nodes. The bivariate distribution can be viewed as a normalized *similarity* measure [60] between a pair of two nodes. A graph G associated with a bivariate distribution $p(\cdot,\cdot)$ is then called a *sampled* graph.

In Section 2, we consider the case that the bivariate distribution is *symmetric* first for undirected networks. Under this assumption, the two marginal distributions of the bivariate distribution are the same and they represent the probability that a particular node is selected in the sampled graph. As such, the marginal distribution can be used for defining the *centrality* of a set as the probability that a selected node is in this set. The larger the centrality of a node is, the larger probability the node is selected. Such a probability measure recovers various centrality measures in the literature as special cases, including the degree centrality and the Katz centrality [61], as they simply correspond to various methods of sampling a graph.

An an extension of centrality, the concept of relative centrality of a set of nodes S_1 with respect to another set of nodes S_2 in the probabilistic framework is formally defined as the *conditional* probability that one node of the selected pair of two nodes is in the set S_1 given that the other node is in the set S_2. When the set S_2 is taken to be the whole set of nodes in the graph, then the relative centrality of S_1 with respect to S_2 is simply reduced to the centrality of S_1, which is the probability that one node of the selected pair of two nodes is in the set S_1. Thus, our view for a community is a group of people (nodes) who consider themselves much more important to themselves than to random people on the street, and the *community strength* for a set of nodes S is defined as the difference between its relative centrality with respect to itself and its centrality. Moreover, a set of nodes with a *nonnegative* community strength is called a *community*. With such a definition of community, there are several mathematically equivalent statements of a *community* that can be intuitively explained by their social meanings. There are also several interesting interpretations of the community strength. In particular, the community strength for a certain sampled graph is related to *conductance* that is commonly used for measuring the strength of a small community [8], [9], [10]. Also, for a certain sampled graph, it is related to the probability that a random walker is trapped inside the community for a certain period of time [19], [20].

The original modularity in [25] is a measure to quantify the strength of community structure in a partition of a graph and such a measure has been widely accepted for analyzing community structure in the literature. One of the well-known problems of Newman's modularity is its resolution limit in detecting communities [62]. As such, there are other measures, such as stability in [21], [22], that were proposed for quantifying the strength of community structure in a partition of a graph. However, when the communities in a network span a wide range of sizes, it is not possible to find an optimal value of the resolution parameter to identify simultaneously all the communities in a network [63], [64]. In the framework, the (generalized) *modularity* for a partition of a sampled graph is defined as the average community strength of the community to which a randomly selected node belongs. As such, a high modularity for a partition of a graph implies that there are communities with strong community strengths. The original modularity in [25] and stability in [21], [22] are simply special cases of the (generalized) modularity for certain sampled graphs.

In Section 3, we introduce community detection algorithms. As the modularity for a partition of a network is the average community strength of a randomly selected node, a good partition of a network should have a large modularity. In view of this, one can then tackle the community detection problem by looking for algorithms that yield large modularity. However, the modularity maximization problem is known to be NP-hard [65] and one has to resort to heuristic algorithms.

In this section, we introduce four commonly used modularity maximization algorithms for community detection in *undirected* networks: (i) the spectral modularity maximization algorithm [66], (ii) the hierarchical agglomerative algorithm [14], (iii) the partitional algorithm [67], and (iv) the fast unfolding algorithm [68]. Under the probabilistic framework, the last three algorithms converge in a finite number of steps, and the outputs of the hierarchical agglomerative algorithm and the fast unfolding algorithm are guaranteed to be *communities*. Moreover, the partitional algorithm is a linear time algorithm for large sparse graphs.

We summarize the logic flow of using sampled graphs for centrality analysis and community detection of networked data in Figure 1. First, we model the networked data by a graph $G = (V_g, E_g)$, where V_g is the set of vertices and E_g is the set of edges. Then we provide a viewpoint by sampling the graph with the bivariate distribution $p(\cdot, \cdot)$. This results in a sampled graph. Based on this sampled graph, we can then define the notions of relative centrality and centrality. The notions of community and modularity are built upon the notions of relative centrality and centrality. Ranking algorithms and community detection algorithms can then be developed and written in codes by using sampled graphs as inputs. If one is not satisfied with the community detection result from a sampled graph, e.g., the resolution, one has the freedom to choose another viewpoint for the graph and try out the analysis again. There is no need to rewrite the codes for the community detection algorithm.

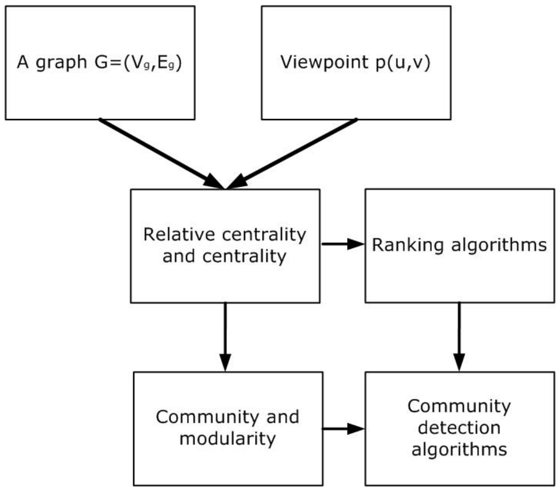


Fig. 1. The logic flow of using sampled graphs for structural analysis and community detection.

In Section 4, we extend the probabilistic framework from undirected networks to *directed* networks, where the sampling bivariate distributions could be *asymmetric*. As pointed out in the recent survey [69], this is important as many networks in sociology, biology, neuroscience and computer science, are *directed* networks. The key insight is that we can relax the assumption from *symmetric* bivariate distributions to bivariate distributions that have the same marginal distributions in [70]. By using such a weaker assumption, the notions of centrality, relative centrality, community and modularity can be defined in the same manner as before.

Another point that needs to be addressed in [69] is algorithm design and evaluation of community detection algorithms in directed networks. Since the bivariate distribution could be *asymmetric*, these community detection algorithms for *undirected* networks cannot be directly applied. For this, one needs to symmetrize the bivariate distribution. Such a symmetrization process not only preserve modularity but also preserve sparsity (of the sampled graph). As such, the computational complexity of these community detection algorithms for directed networks remains the same as that of their undirected counterparts.

The further extension to the setting with general bivariate distributions is possible. However, the results are not as elegant as the setting with bivariate distributions that have the same marginal distributions. The good news is that the notions of *community* and *modularity* can be extended in the same manner. Moreover, the hierarchical agglomerative algorithm, the partitional algorithm, and the fast unfolding algorithm can also be extended to the setting with some minor modifications.

In Section 5, we introduce several commonly used datasets for benchmarking community detection algorithms, including the stochastic block model, the LFR model, the six cluster model and the Stanford Network Analysis Project Collection (SNAP) [71]. To see how the community detection algorithms perform, we consider two methods for sampling a directed network with a bivariate distribution that has the same marginal distributions: (i) PageRank and (ii) random walks with self-loops and backward jumps. Though PageRank [5] has been very successful in ranking nodes in directed networks, the experimental results show that its performance in community detection is not as good as sampling by a random walk with self-loops and backward jumps. This might be due to the fact that PageRank adds weak links in a network and that changes the topology of the network and thus affects the results of community detection. In [72], the authors pointed out a similar problem for the original PageRank and provided various ways to modify PageRank to tackle this problem.

In Section 6, we consider attributed networks. An attributed network is a generalization of a network (graph). In addition to the set of nodes and the set of edges in a (underlining) network, an attributed network could have node attributes and edge attributes that specify the "features" of nodes and edges. For instance, a signed network is an attributed network, where each edge is labelled with either a positive sign or a negative sign to indicate the friendship/enemy relationship between the two ends of an edge. Another typical example of an attributed network is that every node in the network represents a person with different ratings/interests on various topics.

The centrality analysis in attributed networks is to rank the importance of nodes in attributed networks. For this, we extend the probabilistic framework by using the exponentially twisted sampling technique. Like PageRank [5], the probabilistic framework requires a sampling method of a network, called a *viewpoint*. The sampling method of a network is characterized by a probability measure for randomly selecting a path r in the network. In order to take the attributes of nodes and edges into account, one needs a *biased* viewpoint as previously discussed in Personalized PageRank [73], [74]. As such, the sampling method in an attributed network (and the corresponding probability measure) needs to be a *twisted* probability measure of its underlining network. For this, we use a path measure $f(r)$ that maps every path r in an attributed network to a real-valued vector. By specifying the average values of a path measure, we then have a set of constraints for the twisted sampling probability measure. This then leads to the exponentially twisted probability measure [75], [76], [77] that minimizes the Kullback-Leibler distance between the twisted probability measure and the original probability measure under the set of constraints from the average values of the path measure.

Each path measure with specified average values leads to a method of ranking nodes in an attributed network and that method is in general different from the original ranking method in its underlining network. In Section 6, we introduce three path measures in attributed networks and that leads to three new notions of centralities in attributed networks. For signed networks that have both positive edges and negative edges, we demonstrate how the influence centralities can be defined by using a path measure derived from opinions dynamics and how the trust centralities can be defined by using a path measure derived from a chain of trust. In particular, we demonstrate that one may vary the specified average value of the path measure in a signed network so that the influence centrality is turned into positive degree ranking, negative degree ranking and total degree ranking. For attributed networks with node attributes, we also demonstrate how advertisement-specific influence centralities can be defined by using a specific path measure that models influence cascades in such networks.

In Section 7, we give a brief introduction of the clustering problem. For a clustering problem, there is a set of data points (or objects) and a similarity (or dissimilarity) measure that measures how similar two data points are. The aim of a clustering algorithm is to cluster these data points so that data points within the same cluster are similar to each other and data points in different clusters are dissimilar. In this section, we briefly review several commonly used clustering algorithms, including the K-means algorithm for clustering data points in a Euclidean space, the K-medoids algorithm for clustering data points in a non-Euclidean space, and the spectral clustering algorithm for clustering data points with a similarity measure.

In Section 8, we address the clustering problem for data points in a metric/semi-metric space by using the probabilistic framework. For this, we consider a set of n data points, $\Omega = \{x_1, x_2, \ldots, x_n\}$ and a distance measure $d(x, y)$ for any two points x and y in Ω. The distance measure $d(\cdot, \cdot)$ is said be a *semi-metric* if it satisfies the following three properties: (i) $d(x, y) \geq 0$, (ii) $d(x, x) = 0$, and (iii) $d(x, y) = d(y, x)$. If, furthermore, the distance measure satisfies the triangular inequality, then the distance measure is said to be a *metric*. The key idea of sampling the data points in a semi-metric space is to consider a complete graph with n nodes and n self edges and then maps each data point in Ω to a node in the graph with the edge weight between two nodes being the distance between the corresponding two points in Ω. By doing so, the problem of clustering in a semi-metric space is transformed to a community detection problem of a weighted graph.

In this section, we are particularly interested in the exponentially twisted sampling technique in which one can specify the desired average distance from the sampling distribution to detect clusters with various resolutions. The exponentially twisted sampling technique leads to a new *cohesion* measure in terms of the distance measure. Using the cohesion measure, a cluster is defined as a set of points that are cohesive to themselves. For such a definition, there are various equivalent statements and these statements can be explained intuitively. The cohesion measure enables us to develop a partitional algorithm, called the K-sets algorithm, that repeatedly assigns every data point to the closest set in terms of the triangular distance. Moreover, the K-sets algorithm converges in a finite number of iterations. In particular, for $K = 2$, the K-sets algorithm returns two clusters when the algorithm converges.

One interesting property of the cohesion measure is the duality result between a distance metric and a cohesion measure. There is a general definition of a cohesion measure. For each cohesion measure, there is an induced distance metric, called the *dual distance metric*. On the other hand, there is also an induced cohesion measure, called the *dual cohesion measure*, for each distance metric. The dual distance metric of a dual cohesion measure of a distance metric is the distance metric itself. Such a duality result leads to a dual K-sets algorithm for clustering a set of data points with a cohesion measure. The dual K-sets algorithm converges in the same way as a sequential version of the classical kernel K-means algorithm. The key difference is that a cohesion measure does not need to be positive semi-definite.

For data points in a semi-metric space, the distance measure does not necessarily satisfy the triangular inequality. Without the triangular inequality, the K-sets algorithm may not converge at all. Even if it converges, there is no guarantee that the output of the K-sets algorithm are clusters. To tackle this technical challenge, we introduce the K-sets$^+$ algorithm for clustering in a semi-metric space in [78]. In the K-sets$^+$ algorithm, we need to modify the original definition of the triangular distance so that the nonnegativity requirement of the triangular distance can be lifted. For this, we introduce the adjusted triangular distance such that the K-sets$^+$ algorithm converges in a finite number of iterations. Moreover, the K-sets$^+$ algorithm outputs K disjoint sets such that any two sets of these K sets are two disjoint clusters when they are viewed in isolation.

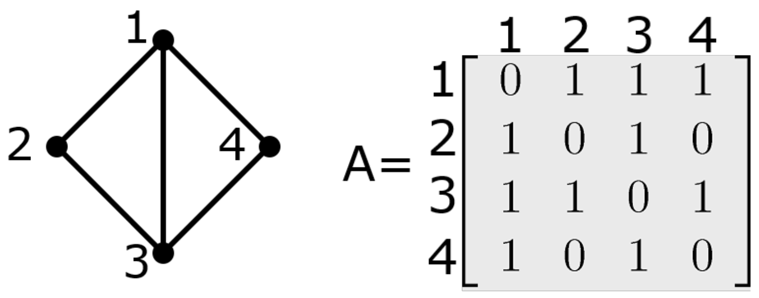
One can further extend the applicability of the K-sets$^+$ algorithm from data points in a semi-

TABLE 1
List of Notations

Network notations	
$G(V_g, E_g)$	A graph with the vertex set V_g and the edge set E_g
n	The number of vertices (nodes) in a graph
m	The number of edges (links) in a graph
m^+	The number of positive edges in a signed network
m^-	The number of negative edges in a signed network
A	The adjacency matrix of a graph
A_{vw} (or a_{vw})	The indicator variable for the edge (v, w)
k_v	The degree of vertex v
k_v^{out}	The outgoing edges of vertex v
k_v^{in}	The incoming edges of vertex v
$p(r)$	The probability of sampling the path r in a graph
$p(v, w)$	The bivariate distribution of a sampled graph
$p_V(v)$	The marginal distribution of the first variable in $p(v, w)$
$p_W(w)$	The marginal distribution of the second variable in $p(v, w)$
$C(S)$	The centrality of the set S
$C(S_1\|S_2)$	The relative centrality of S_1 with respect to S_2
$Str(S) = C(S\|S) - C(S)$	The community strength of a set S
$q(v, w) = p(v, w) - p_V(v)p_V(w)$	The correlation measure between two nodes v and w
$q(S_1, S_2) = \sum_{v\in S_1} \sum_{w\in S_2} q(v, w)$	The *correlation measure* between two sets S_1 and S_2
$q_0(v, w)$	The correlation measure between two nodes v and w with $q_0(v, v) = 0$
$Q(\mathcal{P})) = \sum_{k=1}^{K} q(S_k, S_k)$	The *modularity* for a partition $\mathcal{P} = \{S_1, S_2, \ldots, S_K\}$ of a graph
π_v	The steady state probability for a Markov chain to visit state v
$p_{v,w}$	The transition probability from state v to state w
$\phi(S)$	The conductance of the set S in a graph
$h_V(u)$	The attribute of node u in an attributed network
$h_E(e)$	The attribute of edge e in an attributed network
$f(r)$	The path measure of a path r in an attributed network
Clustering notations	
$\Omega = \{x_1, x_2, \ldots, x_n\}$	The set of all data points
n	The total number of data points
$d(x, y)$	The distance metric (or semi-metric) between two points x and y
$\bar{d}(S_1, S_2)$	The *average* distance between two sets S_1 and S_2 in (164)
$p_\lambda(x, y)$	The exponentially twisted distribution in (150)
$p_\lambda(x)$	The marginal distribution of the bivariate distribution $p_\lambda(x, y)$
$\gamma_\lambda(x, y) = p_\lambda(x, y) - p_\lambda(x)p_\lambda(y)$	The correlation between two points x and y in (155)
$\gamma_\lambda(S_1, S_2) = \sum_{x\in S_1} \sum_{y\in S_2} \gamma_\lambda(x, y)$	The *correlation measure* between two sets S_1 and S_2
$\gamma(x, y)$	The *cohesion measure* (or semi-cohesion measure, similarity measure) between x and y
$\gamma(S_1, S_2) = \sum_{x\in S_1} \sum_{y\in S_2} \gamma(x, y)$	The *cohesion measure* (or semi-cohesion measure) between two sets S_1 and S_2
$\Delta(x, S) = 2\bar{d}(\{x\}, S) - \bar{d}(S, S)$	The *triangular distance* (or the Δ-distance) from a point x to a set S
$\Delta_{\rm a}(x, S)$	The *adjusted Δ-distance* from a point x to a set S in (196)
$Q(\mathcal{P}) = \sum_{k=1}^{K} \gamma(S_k, S_k)$	The *modularity* for a partition $\mathcal{P} = \{S_1, S_2, \ldots, S_K\}$ of Ω
$Q_{\rm nor}(\mathcal{P}) = \sum_{k=1}^{K} \gamma(S_k, S_k)/\|S_k\|$	The *normalized modularity* for a partition $\mathcal{P} = \{S_1, S_2, \ldots, S_K\}$ of Ω

metric space to data points that only have a symmetric similarity measure. A similarity measure is generally referred to as a bivariate function that measures how similar two data points are. There is a natural mapping from a symmetric similarity measure to a distance measure in a semi-metric space and the K-sets$^+$ algorithm that uses this distance measure converges to the same partition as that using the original symmetric similarity measure. Such an extension leads to great reduction of computational complexity for the K-sets$^+$ algorithm.

For the clarity of the presentation, all the mathematical proofs are deferred to the end of this tutorial in Section 9. In Table 1, we provide a list of notations used in this tutorial.



$$A=\begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 1 & 1 & 1 \\ 2 & 1 & 0 & 1 & 0 \\ 3 & 1 & 1 & 0 & 1 \\ 4 & 1 & 0 & 1 & 0 \end{array}$$

(a) The graph (b) The adjacency matrix

Fig. 2. A graph with 4 nodes and 5 edges.

2 The probabilistic framework for undirected networks

2.1 Sampling a network

In the literature, an undirected network is commonly modelled by an undirected graph $G(V_g, E_g)$, where V_g denotes the set of vertices (nodes) in the graph and E_g denotes the set of edges (links) in the graph. Let $n = |V_g|$ be the number of vertices in the graph and index the n vertices from $1, 2, \ldots, n$. Then the graph $G(V_g, E_g)$ can also be characterized by an $n \times n$ adjacency matrix A, where

$$A_{vw} = \begin{cases} 1, & \text{if vertices } v \text{ and } w \text{ are connected,} \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Let $m = |E_g|$ be the number of edges in the graph and k_v be the degree of vertex v. From the adjacency matrix, we then have

$$m = \frac{1}{2}\sum_{v=1}^{n}\sum_{w=1}^{n} A_{vw}, \tag{2}$$

and

$$k_v = \sum_{w=1}^{n} A_{vw}. \tag{3}$$

In Figure 2, we show a graph with 4 nodes and 5 edges.

The main idea of the probabilistic framework is to *sample* a network by randomly selecting two nodes V and W according to a specific bivariate distribution $p(\cdot, \cdot)$. Specifically, for a network with the set of nodes $\{1, 2, \ldots, n\}$, we have

$$\mathsf{P}(V = v, W = w) = p(v, w). \tag{4}$$

Let $p_V(v)$ (resp. $p_W(w)$) be the marginal distribution of the random variable V (resp. W), i.e.,

$$p_V(v) = \sum_{w=1}^{n} p(v, w), \tag{5}$$

and

$$p_W(w) = \sum_{v=1}^{n} p(v, w). \tag{6}$$

If $p(v, w)$ is *symmetric*, then $p_V(v) = p_W(v)$ for all v, and $p_V(v)$ is the probability that a randomly selected node is v.

Definition 1. **(Sampled graph)** A graph $G(V_g, E_g)$ that is sampled by randomly selecting two nodes V and W according to a specific bivariate distribution $p(\cdot, \cdot)$ in (4) is called a *sampled graph* and it is denoted by the two tuple $(G(V_g, E_g), p(\cdot, \cdot))$.

In the following, we introduce several methods to generate sampled graphs by specifying the needed bivariate distributions.

Example 2. **(Uniform edge sampling)** One simple method is to generate V and W as the two end nodes of a *uniformly* selected edge from the graph $G(V_g, E_g)$. For this edge sampling method, we have

$$\mathsf{P}(V = v, W = w) = \frac{1}{2m} A_{vw}, \tag{7}$$

where m is the number of edges in the graph $G(V_g, E_g)$. In particular, if we use the edge sampling for the graph in Figure 2, we have the following bivariate distribution (written in the 4×4 matrix form):

$$\begin{bmatrix} 0 & \frac{1}{10} & \frac{1}{10} & \frac{1}{10} \\ \frac{1}{10} & 0 & \frac{1}{10} & 0 \\ \frac{1}{10} & \frac{1}{10} & 0 & \frac{1}{10} \\ \frac{1}{10} & 0 & \frac{1}{10} & 0 \end{bmatrix}.$$

The problem of using the uniform edge sampling is its limited visibility to "see" more than one step and that might cause a serious problem in community detection (known as the resolution limit [62]). To see the intuition behind this, let us consider the illustrating example in Figure 3. In the figure, there are two clearly separated "communities" A and B. But it is difficult to see whether vertex C should be classified to community A by considering paths with length not greater than 1 (as vertex C has exactly one path with length 1 to each community). However, if we consider paths with length 2, then it is obvious that we should classify vertex C to community A.

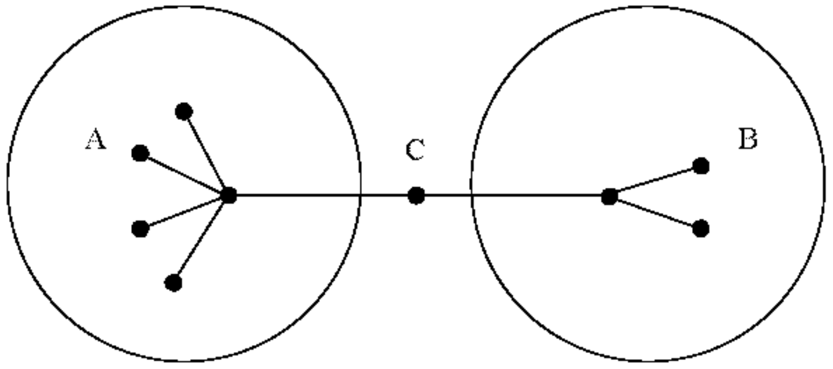


Fig. 3. An illustrating example for vertices that are difficult to classify by considering paths with length not greater than 1.

In view of Figure 3, we seek a more general method is to generate the needed bivariate distribution by randomly selecting the two ends of a *path*. Specifically, let $R_{v,w}$ be the set of paths from v to w and $R = \cup_{v,w \in V_g} R_{v,w}$ be the set of paths in the graph $G(V_g, E_g)$. According to a probability mass function $p(\cdot)$, called the *path sampling distribution*, a path $r \in R$ is selected at random with probability $p(r)$. Let V (resp. W) be the starting (resp. ending) node of a randomly select path by using the path sampling distribution $p(\cdot)$. Then the bivariate distribution

$$p(v, w) = \mathsf{P}(V = v, W = w) = \sum_{r \in R_{v,w}} p(r) \tag{8}$$

is the probability that the ordered pair of two nodes (v, w) is selected. As such, $p(v, w)$ can be viewed as a similarity measure between node v and node w.

One way of randomly selecting the two ends of a path is by mapping the adjacency matrix. Specifically, one can first choose a (matrix) function f that maps an adjacency matrix A to another nonnegative matrix $f(A)$. Then one can define a bivariate distribution from $f(A)$ by

$$\mathsf{P}(V = v, W = w) = \frac{1}{||f(A)||_1} f(A)_{vw}, \tag{9}$$

where $||f(A)||_1 = \sum_v \sum_w |f(A)_{vw}|$ is usual "entrywise" matrix norm of the matrix $f(A)$.

Example 3. **(A random selection of a path with length not greater than 2)** Consider a graph with an $n \times n$ adjacency matrix A and

$$f(A) = \lambda_0 \mathbf{I} + \lambda_1 A + \lambda_2 A^2, \tag{10}$$

where $\mathbf{I}$ is the $n \times n$ identity matrix, and λ_0, λ_1, and λ_2 are three *nonnegative* constants. Then the two random variables V and W in (9) represent the two ends of a randomly selected path with length not greater than 2. To see this, note that there are n paths with length 0 (for the n vertices), $||A||_1$ paths with length 1, and $||A^2||_1$ paths with length 2. Since

$$||f(A)||_1 = \lambda_0 ||\mathbf{I}||_\mathbf{1} + \lambda_\mathbf{1} ||\mathbf{A}||_\mathbf{1} + \lambda_\mathbf{2} ||\mathbf{A^2}||_\mathbf{1},$$

a path with length ℓ is selected with probability $\lambda_\ell / ||f(A)||_1$ for $\ell = 0, 1$, and 2.

As described in [60], randomly selecting the two ends of a path by mapping the adjacency matrix can also be viewed as a way to compute the "similarity" score between a pair of two nodes v and w. In fact, if there is a bounded similarity measure $sim(v, w)$ that gives a high score for a pair of two "similar" nodes v and w, then one can map that similarity measure to a bivariate distribution $p(v, w)$ as follows:

$$p(v, w) = \frac{sim(v, w) - \text{MINsim}}{\sum_{x=1}^{n} \sum_{y=1}^{n} \Big(sim(x, y) - \text{MINsim} \Big)}, \tag{11}$$

where

$$\text{MINsim} = \min_{1 \le x, y \le n} sim(x, y), \tag{12}$$

is the minimum value of all the similarity scores. As such, one can view a sampled graph $(G(V_g, E_g), p(\cdot, \cdot))$ as a graph $G(V_g, E_g)$ associated with a *normalized* similarity measure $p(\cdot, \cdot)$.

Another approach to generate a bivariate distribution from the adjacency matrix A from a graph is to consider a random walk on a graph (see e.g., [76]).

Example 4. **(A random walk on a graph)** Consider a graph with an $n \times n$ adjacency matrix A. As in (2) and (3), let m be the total number of edges and k_v be the degree of vertex v. A random walker on a graph hops from nodes to nodes with respect to time. When a random walker is at node i at time t, it selects uniformly at random one of the neighbors of node i and hops to that neighbor at time $t + 1$. A random walk on such a graph can then be characterized by a Markov chain with the $n \times n$ transition probability matrix $(p_{v,w})$, where

$$p_{v,w} = \frac{1}{k_v} A_{vw} \tag{13}$$

is the transition probability from vertex v to vertex w. The stationary probability that the Markov chain is in vertex v, denoted by π_v, is $k_v / 2m$. Let β_ℓ be the probability that we select a path with length ℓ, $\ell = 1, 2, \ldots$. Then the probability of selecting a random walk (path) with vertices $v = v_1, v_2, \ldots, v_\ell = w$ is

$$\beta_\ell \pi_{v_1} \prod_{i=1}^{\ell-1} p_{v_i, v_{i+1}}. \tag{14}$$

From this, we then have the bivariate distribution

$$p(v, w) = \pi_v \sum_{\ell=1}^{\infty} \beta_\ell \sum_{v_2} \ldots \sum_{v_{\ell-1}} \prod_{i=1}^{\ell-1} p_{v_i, v_{i+1}}. \tag{15}$$

Since A is a *symmetric* matrix, it is easy to see that

$$\pi_v p_{v,w} = \frac{1}{2m} A_{v,w} = \frac{1}{2m} A_{w,v} = \pi_w p_{w,v}$$

for all v and w, and the Markov chain is thus a *reversible* Markov chain [79]. This implies that

$$\pi_{v_1} \prod_{i=1}^{\ell-1} p_{v_i, v_{i+1}} = \pi_{v_\ell} \prod_{i=1}^{\ell-1} p_{v_{i+1}, v_i}$$

and $p(v, w) = p(w, v)$ is thus a symmetric bivariate distribution.

2.2 Centralities and relative centralities

In social network analysis, centralities [2], [3], [4] have been widely used for ranking the importance of nodes in a network. Intuitively, an important node in a graph has a higher probability of being "spotted" than an arbitrary node. Since V and W are two randomly selected nodes in a sample graph $(G(V_g, E_g), p(\cdot, \cdot))$ via the bivariate distribution $p(\cdot, \cdot)$, it seems plausible to define the centrality of a node as the *probability* that a node is selected. This leads us to define the centrality of a set of nodes in a sampled graph in Definition 5 below.

Definition 5. **(Centrality)** For a sampled graph $(G(V_g, E_g), p(\cdot, \cdot))$ with a *symmetric* bivariate distribution $p(\cdot, \cdot)$, the *centrality* of a set of nodes S, denoted by $C(S)$, is defined as the probability that a node in S is selected, i.e.,

$$C(S) = \mathsf{P}(V \in S) = \mathsf{P}(W \in S). \tag{16}$$

Note that

$$\mathsf{P}(W \in S) = \mathsf{P}(W \in S | V \in V_g). \tag{17}$$

Another way to interpret the centrality of a set of nodes S is the *conditional probability* that the randomly selected node W is inside S given that the random selected node V is inside the whole set of nodes in the graph. As $p(\cdot, \cdot)$ can be viewed as a normalized similarity measure, such an observation leads us to define the *relative centrality* of a set of nodes S_1 with respect to another set of nodes S_2 as the conditional probability that the randomly selected node W is inside S_1 given that the random selected node V is inside S_2. Intuitively, a node w is relatively important to a set of node S_2 if the node w has a high probability of being "spotted" from the set of nodes in S_2. Such a definition is formalized in Definition 6 below.

Definition 6. **(Relative centrality)** For a sampled graph $(G(V_g, E_g), p(\cdot, \cdot))$ with a *symmetric* bivariate distribution $p(\cdot, \cdot)$, the *relative centrality* of a set of nodes S_1 with respect to another set of nodes S_2, denoted by $C(S_1|S_2)$, is defined as the conditional probability that the randomly selected node W is inside S_1 given that the random selected node V is inside S_2, i.e.,

$$C(S_1|S_2) = \mathsf{P}(W \in S_1 | V \in S_2). \tag{18}$$

In particular, when $S_2 = V_g$ is the set of all the nodes in the graph, the relative centrality of a set of nodes S_1 with respect to V_g is reduced to the *centrality* of the set of nodes S_1, i.e.,

$$C(S_1|V_g) = \mathsf{P}(W \in S_1|V \in V_g) = \mathsf{P}(W \in S_1) = C(S_1). \tag{19}$$

Note that

$$\begin{aligned} C(S_1|S_2) &= \mathsf{P}(W \in S_1|V \in S_2) = \frac{\mathsf{P}(V \in S_2, W \in S_1)}{\mathsf{P}(V \in S_2)} \\ &= \frac{\sum_{v \in S_2} \sum_{w \in S_1} p(v,w)}{\sum_{v \in S_2} p_V(v)}, \end{aligned} \tag{20}$$

where $p_V(v)$ is the marginal distribution of V in (5). Also,

$$C(S_1) = \mathsf{P}(W \in S_1) = \sum_{w \in S_1} \mathsf{P}(W = w) = \sum_{w \in S_1} p_W(w), \tag{21}$$

where $p_W(w)$ is the marginal distribution of W in (6). Since we assume that $p(\cdot,\cdot)$ is symmetric, we also have

$$C(S_1) = \mathsf{P}(V \in S_1) = \sum_{w \in S_1} p_V(w). \tag{22}$$

As relative centrality is defined as the conditional probability, there are several properties of relative centrality that can be induced from a probability measure. This is shown in the following proposition and its proof is given in Section 9.1.

Proposition 7. For a sampled graph $(G(V_g, E_g), p(\cdot,\cdot))$ with a *symmetric* bivariate distribution $p(\cdot,\cdot)$, the following properties for the relative centrality defined in Definition 6 hold.

(i) $0 \le C(S_1|S_2) \le 1$ and $0 \le C(S_1) \le 1$. Moreover, $C(V_g|S_2) = 1$ and $C(V_g) = 1$.

(ii) (Additivity) If S_1 and S_1' are two disjoint sets, i.e., $S_1 \cap S_1'$ is an empty set, then $C(S_1 \cup S_1'|S_2) = C(S_1|S_2) + C(S_1'|S_2)$ and $C(S_1 \cup S_1') = C(S_1) + C(S_1')$.

(iii) (Monotonicity) If S_1 is a subset of S_1', i.e., $S_1 \subset S_1'$, then $C(S_1|S_2) \le C(S_1'|S_2)$ and $C(S_1) \le C(S_1')$.

(iv) (Reciprocity)

$$C(S_1)C(S_2|S_1) = C(S_2)C(S_1|S_2).$$

As a result, $C(S_1) \ge C(S_2)$ if and only if $C(S_2|S_1) \le C(S_1|S_2)$.

We note that the reciprocity in Proposition 7(iv) implies the following weaker version of reciprocity

$$C(S)C(S^c|S) = C(S^c)C(S|S^c). \tag{23}$$

Such a weaker version will be used for directed networks.

Now we provide several illustrating examples for relative centrality, including the degree centrality, the Katz centrality, and the continuous-time random walk.

Example 8. **(Degree centrality)** Consider the bivariate distribution in (7), i.e.,

$$p(v,w) = \mathsf{P}(V = v, W = w) = \frac{1}{2m} A_{vw}. \tag{24}$$

Thus, we have from (20) and (24) that

$$C(S_1|S_2) = \frac{\sum_{v \in S_2} \sum_{w \in S_1} A_{v,w}}{\sum_{v \in S_2} k_v}. \tag{25}$$

In particular, when S_1 contains a single node w and $S_2 = V_g$ is the set of all the nodes in the graph, the centrality of w, i.e., $C(\{w\})$, is simply $\mathsf{P}(W = w)$. From (24), we have

$$C(\{w\}) = \mathsf{P}(W = w) = \frac{k_w}{2m}. \tag{26}$$

Thus, $C(\{w\})$ is the usual (normalized) degree centrality that counts the number of edges connected to the node w.

As another illustrating example, we show that Katz's centrality can also be derived as a special case of the sampled graph.

Example 9. **(Katz centrality)** For an undirected graph $G(V_g, E_g)$, Katz centrality [61] is based on the assumption that the importance of a vertex is the sum of a fixed constant and the "discounted" importance of vertices it connects to. Specifically, let x_v be the Katz centrality of vertex v. Then

$$x_v = \sum_{w=1}^{n} \lambda A_{vw} x_w + 1, \tag{27}$$

where λ is the discount factor. Let $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$ be the vector of Katz centralities and $\mathbf{1} = (1, 1, \ldots, 1)^T$ be the n-vector of all 1's. Then we can write (27) in the matrix form $\mathbf{x} = \lambda A \mathbf{x} + \mathbf{1}$ and solve for $\mathbf{x}$ to yield

$$\mathbf{x} = (\mathbf{I} - \lambda A)^{-1} \cdot \mathbf{1}, \tag{28}$$

where $\mathbf{I}$ is the $n \times n$ identity matrix. To show that the Katz centrality is the centrality of a particular sampled graph, we choose

$$f(A) = \sum_{i=0}^{\infty} (\lambda A)^i = (\mathbf{I} - \lambda A)^{-1} \tag{29}$$

in (9) and thus we have the bivariate distribution

$$p(v, w) = \frac{1}{||(\mathbf{I} - \lambda A)^{-1}||_1} (\mathbf{I} - \lambda A)^{-1}. \tag{30}$$

The matrix $(\mathbf{I} - \lambda A)^{-1}$ in (29) is called the "Katz similarity" in [4] and the bivariate distribution in (30) can thus be viewed as the normalized Katz similarity. Since A is the adjacency matrix of an undirected graph, we have $A = A^T$ and thus $p(v, w)$ is symmetric. This then leads to

$$C(\{w\}) = p_W(w) = p_V(w) = \frac{\mathbf{e}_w^T}{||(\mathbf{I} - \lambda A)^{-1}||_1} (\mathbf{I} - \lambda A)^{-1} \cdot \mathbf{1}, \tag{31}$$

where $\mathbf{e}_w$ is the column vector that has 1 in its w^{th} element and 0 otherwise. Thus, $C(\{w\})$ is the (normalized) Katz centrality of node w. For this example, we also have from the symmetry of A, (20), and (30) that

$$C(S_1|S_2) = \frac{\mathbf{e}_{S_2}^T \cdot (\mathbf{I} - \lambda A)^{-1} \cdot \mathbf{e}_{S_1}}{\mathbf{e}_{S_2}^T \cdot (\mathbf{I} - \lambda A)^{-1} \cdot \mathbf{1}}, \tag{32}$$

where $\mathbf{e}_S$ is the column vector that has 1 in the elements of the set S and 0 otherwise.

Example 10. **(Continuous-time random walk on an undirected graph)** In Example 4, we consider a random walk on an undirected graph that takes one unit of time to hop from one node to another node. A continuous-time random walk, as a generalization of the random walk in Example 4, takes a random amount of time to hop from one node to another node. The times between two successive hops (the inter-hop times) are independent and exponentially distributed with mean 1. Thus, a continuous-time random walk on an undirected graph $G = (V_g, E_g)$ can be characterized by a continuous-time Markov process with the transition rate A_{vw}/k_v from vertex v to vertex w. Let $\gamma_v(t)$ be the probability that the continuous-time random walk is at vertex v at time t when the walk is started from vertex w at time 0. Since that the transition rate from vertex v to vertex w is A_{vw}/k_v, it then follows that

$$\frac{d\gamma_v(t)}{dt} = \sum_{w=1}^{n} \frac{A_{wv}}{k_w}\gamma_w(t) - \gamma_v(t) = \sum_{w=1}^{n} \frac{A_{vw}}{k_w}\gamma_w(t) - \gamma_v(t), \tag{33}$$

where we use $A = A^T$ in the last equality. Let π_v be the steady probability that the continuous-time random walk is at vertex v. In view of (33),

$$\pi_v = \lim_{t\to\infty} \gamma_v(t) = \frac{k_v}{2m}. \tag{34}$$

Let $\gamma(t) = (\gamma_1(t), \dots, \gamma_n(t))^T$ and $L = (L_{v,w})$ be the normalized graph Laplacian, i.e.,

$$L_{v,w} = \frac{A_{vw}}{k_w} - \delta_{v,w}. \tag{35}$$

We can rewrite (33) in the matrix form as follows:

$$\frac{d\gamma(t)}{dt} = L\gamma(t). \tag{36}$$

This then leads to

$$\gamma(t) = e^{tL}\gamma(0). \tag{37}$$

Since we start from vertex w at time 0, i.e., $\gamma_w(0) = 1$, we then have

$$\gamma_v(t) = (e^{tL})_{vw}. \tag{38}$$

Now we choose the two nodes V and W as the two ends of a path of length t in a stationary continuous-time random walk. Specifically,

$$\begin{aligned} p(v,w) &= \mathsf{P}(V = v, W = w) = \mathsf{P}(W = w)\mathsf{P}(V = v|W = w) \\ &= \pi_w\gamma_v(t) = (e^{tL})_{vw}\frac{k_w}{2m}, \end{aligned} \tag{39}$$

where we use (38) and (34) in the last identity. Since a stationary continuous-time random walk is a reversible Markov process, we also have

$$p(v,w) = \mathsf{P}(V = v, W = w) = \mathsf{P}(V = w, W = v) = p(w,v).$$

For this example, we then have

$$C(S_1|S_2) = \frac{\sum_{v\in S_2}\sum_{w\in S_1}(e^{tL})_{vw}\frac{k_w}{2m}}{\sum_{v\in S_2}\frac{k_v}{2m}}, \tag{40}$$

$$C(S_1) = \sum_{w\in S_1}\frac{k_w}{2m}. \tag{41}$$

2.3 Community and community strength

Intuitively, a community in a social network includes a group of people who consider themselves much more important to themselves than to random people on the street. In view of this, one might consider a community as a group of nodes that have high relative centralities to each other than to random nodes. As such, we propose the following definition of *community strength* based on relative centrality.

Definition 11. **(Community strength)** For a sample graph $(G(V_g, E_g), p(\cdot, \cdot))$ with a symmetric bivariate distribution $p(\cdot, \cdot)$, the *community strength* of a subset set of nodes $S \subset V_g$, denoted by $Str(S)$, is defined as the difference of the relative centrality of S with respect to itself and its centrality, i.e.,

$$Str(S) = C(S|S) - C(S). \tag{42}$$

In particular, if a subset set of nodes $S \subset V_g$ has a nonnegative community strength, i.e., $Str(S) \geq 0$, then it is called a *community*.

To understand the definition of the community strength, let us consider the continuous-time random walk in Example 10. Let $X(t)$ be the node that the continuous-time random walk visits at time t. Then

$$\begin{aligned} &C(S|S) - C(S) \\ &= \mathsf{P}(W \in S | V \in S) - \mathsf{P}(W \in S) \\ &= \mathsf{P}(X(t) \in S | X(0) \in S) - \mathsf{P}(X(t) \in S). \end{aligned} \tag{43}$$

The first term in (43) is the conditional probability that the continuous-time random walk is "trapped" in the set S given that it is started from that set, while the second term is the steady state probability that the random walk is in the set S. In view of (43), the continuous-time random walk is more likely being "trapped" in a set with a strong community strength. Of course, the large the set is, the more likely the random walk will be "trapped" in that set. As such, the community strength has to take into account the steady state probability that the random walk is in the set. By so doing, for the whole set of nodes in the graph, i.e., V_g, the community strength is normalized to 0 as it should not contain any information for the strength of this trivial community.

To further understand the physical meaning of the concept of community strength, we show how it is related to *conductance* for a small community. In the literature, conductance has been widely used for testing whether a community (cut) is good (see e.g., [8], [9], [10]) when the community (cut) is relatively small comparing to the whole graph. The conductance of a set S, denoted by $\phi(S)$, is defined as

$$\frac{\sum_{v \in S} \sum_{w \notin S} A_{v,w}}{\min[\sum_{v \in S} k_v, \sum_{v \notin S} k_v]}. \tag{44}$$

When S is relatively small comparing to the whole graph, we usually have

$$\sum_{v \in S} k_v \leq \sum_{v \notin S} k_v$$

and the conductance of S is reduced to

$$\phi(S) = \frac{\sum_{v \in S} \sum_{w \notin S} A_{v,w}}{\sum_{v \in S} k_v}. \tag{45}$$

In view of (45), a small community S with a small conductance can be viewed as a good community. Now let us consider the bivariate distribution in (7), where V and W represent the two ends of a uniformly selected edge. For this case, we have from (26) that

$$C(S) = \frac{\sum_{v \in S} k_v}{2m},$$

where m is the total number of edges. For a small community, we expect $\sum_{v \in S} k_v << m$ and $C(S) \approx 0$. Then, we have from (25) in Example 8 that

$$\begin{aligned} Str(S) \approx C(S|S) &= \frac{\sum_{v \in S} \sum_{w \in S} A_{v,w}}{\sum_{v \in S} k_v} \\ &= 1 - \frac{\sum_{v \in S} \sum_{w \notin S} A_{v,w}}{\sum_{v \in S} k_v} = 1 - \phi(S). \end{aligned} \tag{46}$$

In view of (46), a small community S with a large community strength has a small conductance and thus can be considered as a good community.

The concept of *community strength* based on relative centrality enables us to look at the definition of a community from various perspectives. For instance, we know from (42) that a community is a set (of nodes) with its relative centrality to itself not less than its centrality. In addition to this, there are various equivalent statements for a community that can be explained by their social meanings. These are as shown in Theorem 12 below. Its proof is given in Section 9.2.

Theorem 12. Consider a sample graph $(G(V_g, E_g), p(\cdot,\cdot))$ with a symmetric bivariate distribution $p(\cdot,\cdot)$, and a set S with $0 < C(S) < 1$. Let $S^c = V_g \backslash S$ be the set of nodes that are not in S. The following statements are equivalent.

(i) The set S is a community, i.e., $Str(S) = C(S|S) - C(S) \geq 0$.

(ii) The relative centrality of S with respect to S is not less than the relative centrality of S with respect to S^c, i.e., $C(S|S) \geq C(S|S^c)$.

(iii) The relative centrality of S^c with respect to S is not greater than the centrality of S^c, i.e., $C(S^c|S) \leq C(S^c)$.

(iv) The relative centrality of S with respect to S^c is not greater than the centrality of S, i.e., $C(S|S^c) \leq C(S)$.

(v) The set S^c is a community, i.e., $Str(S^c) = C(S^c|S^c) - C(S^c) \geq 0$.

(vi) The relative centrality of S^c with respect to S^c is not less than the relative centrality of S^c with respect to S, i.e., $C(S^c|S^c) \geq C(S^c|S)$.

As mentioned before, the social meaning for the first statement in Theorem 12(i) is that a community is a group of people who consider themselves much more important to themselves than to random people on the street. The second statement in Theorem 12(ii) says that a community is a group of people who consider themselves much more important to themselves than to the other people not in the community. The third statement in Theorem 12(iii) says that the other people not in a community are much less important to the people in the community than to random people on the street. The fourth statement in Theorem 12(iv) says that people in a community are much less important to the other people not in the community than to random people on the street. The fifth statement in Theorem 12(v) says that the other people not in a certain community are also a community. Finally, the sixth statement says that the other people not in a community are much more important to themselves than to the people in a community.

There is a tradeoff between computational complexity and resolution for the choice of a bivariate distribution $p(\cdot,\cdot)$ for community detection. For instance, the bivariate distributions in Example 9 and Example 10 require the complete knowledge of the adjacency matrix A of the graph, and its computational complexity in general very high. On other hand, the degree centrality in Example 8 only requires the *local information* that contains the *first* neighbors of v and w for computing the needed relative centralities. However, exploring a graph by only looking at the first neighbors might lead to a very limited view that sometimes prohibits us to tell the difference among all the first neighbors. For example, as shown in Figure 4, all the three nodes w_1, w_2 and w_3 have the same relative centrality to the node v when the degree centrality in Example 8 is used.

If we further take the *second* neighbors into consideration, we see that the three nodes w_1, w_2 and v forms a triangle. As such, it seems that w_1 and w_2 should have larger relative centralities with respect to v than that of w_3. This suggests using random walks with path length not greater than 2 for computing the relative centralities needed for community detection.

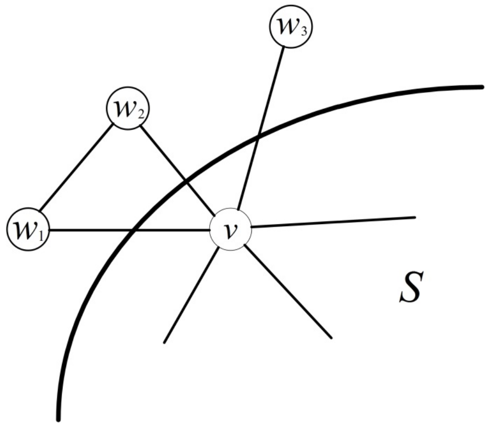


Fig. 4. An illustrating example for relative centralities with three nodes.

Example 13. **(A random walk with path length not greater than 2)** A random walk with path length not greater than 2 is a special case of the random walk in Example 4 and it can be generated by the following two steps: (i) with the probability $k_v/2m$, an initial node v is chosen, (ii) with probability β_i, $i = 1, 2$, a walk from v to w with length i is chosen. Here we add an extra twist by allowing the random walk to select a path with length 0. Specifically, with probability β_0, a walk will remain the same node. As such, we have from (15) that

$$p(v, w) = \frac{\beta_0 k_v \delta_{v,w}}{2m} + \frac{\beta_1}{2m} A_{v,w} + \frac{\beta_2}{2m} \sum_{v_2=1}^{n} \frac{A_{v,v_2} A_{v_2,w}}{k_{v_2}}, \tag{47}$$

where $\beta_0 + \beta_1 + \beta_2 = 1$ and $\beta_i \geq 0$, $i = 1, 2, 3$. Thus,

$$\begin{aligned} C(\{w\}|\{v\}) &= \mathsf{P}(W = w|V = v) \\ &= \frac{\left(\beta_0 k_v \delta_{v,w} + \beta_1 A_{v,w} + \beta_2 \sum_{v_2=1}^{n} \frac{A_{v,v_2} A_{v_2,w}}{k_{v_2}}\right)}{k_v}, \end{aligned} \tag{48}$$

and

$$C(\{w\}) = \frac{k_w}{2m}. \tag{49}$$

In view of (48), we know that $C(\{w\}|\{v\}) = 0$ if node w is neither a first neighbor nor a second neighbor of v.

Also, observe that

$$\begin{aligned} C(S|S) &= \mathsf{P}(W \in S|V \in S) = \frac{\mathsf{P}(W \in S, V \in S)}{\mathsf{P}(V \in S)} \\ &= \frac{\sum_{v \in S} \sum_{w \in S} \left(\beta_0 k_v \delta_{v,w} + \beta_1 A_{v,w} + \beta_2 \sum_{v_2=1}^{n} \frac{A_{v,v_2} A_{v_2,w}}{k_{v_2}}\right)}{\sum_{v \in S} k_v} \\ &= \beta_0 + \beta_1 \frac{\sum_{v \in S} \sum_{w \in S} A_{v,w}}{\sum_{v \in S} k_v} + \beta_2 \frac{\sum_{v \in S} \sum_{w \in S} \sum_{v_2=1}^{n} \frac{A_{v,v_2} A_{v_2,w}}{k_{v_2}}}{\sum_{v \in S} k_v}. \end{aligned} \tag{50}$$

For a small community with $C(S) << 1$, the community strength of a set S can be represented by (50), where the first term is simply a constant β_0 (for normalizing the community strength), the second term is equal to $\beta_1(1 - \text{conductance})$ (see (45)), and the third term is related the clustering coefficient (in terms of the number of triangles) in the set S. In view of this, using a random walk with path length not greater than 2 to sample a network seems to be a good compromise between the viewpoint from conductance [8], [9], [10] and the viewpoint of clustering coefficient [12].

As β_0 is only a constant for normalizing the community strength, one can increase the community strength of a node by increasing β_0. Note that

$$Str(\{w\}) = C(\{w\}|\{w\}) - C(\{w\}) \geq \beta_0 - \frac{k_w}{2m}. \tag{51}$$

Thus, if we choose $\beta_0 \geq \frac{k_{\max}}{2m}$ (with $k_{\max} = \max_{v \in V_g} k_v$ being the maximum degree), then $Str(\{w\}) \geq 0$ for every node w in the sampled graph. As such, every single node has a nonnegative community strength and thus a community by itself.

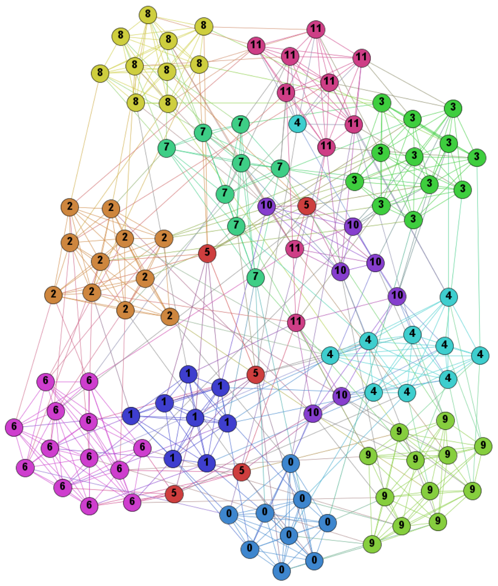
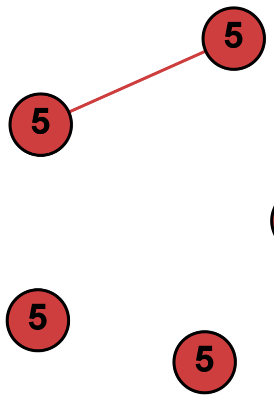


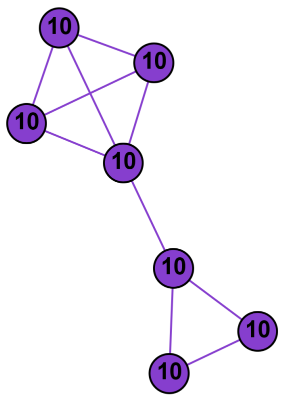
Fig. 5. The network of American football games between Division IA colleges during regular season Fall 2000.

To provide insights of community strength, we consider the dataset from the American football games in the following example.

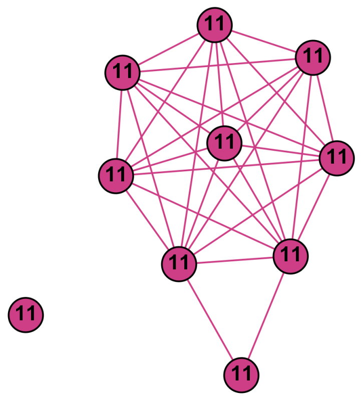
Example 14. **(College football games)** In this example, we use the random walk with path length not greater than 2 in Example 13 to sample the network from the American football games between Division IA colleges during regular season Fall 2000 [80]. We use the following coefficients in (47): $\beta_2 = 0.25$, $\beta_0 = k_{\max} / 2m$, and $\beta_1 = 1 - \beta_0 - \beta_2$, where $k_{\max} = \max_{v \in V_g} k_v$ is the maximum degree. In Figure 5, nodes represent teams and edges represent regular-season games between two teams. The teams are divided into 12 conferences. As shown in Figure 5, the competitions (edges) in the same conference are more intense than those between two different conferences. We compute the community strengths of these 12 conferences to see how strong these conferences (as communities) are. These results are shown in the second column of Table 2 and there are several conferences that have small community strengths, in particular conference 5 only has the community strength 0.04.



(a) conference 5



(b) conference 10



(c) conference 11

Fig. 6. Edges within conferences 5,10, and 11 of American football games.

To further understand this, we plot the edges within conference 5 in Figure 6 (a). There is only one edge between two teams in this conference! We also plot the edges within conferences 10 and 11 in Figure 6 (b) and Figure 6 (c), respectively. These two conferences also have relative low community strengths. As shown in Figure 6 (b), Conference 10 is formed by a single edge between a clique of four nodes and another clique of three nodes. As for conference 11, there is one team that did not play any games with any other teams in the same conference.

TABLE 2
The community strengths of the 12 conferences in the network of the American football games.

conference	0	1	2	3	4	5	6	7	8	9	10	11
strength	0.63	0.54	0.57	0.60	0.46	0.04	0.59	0.52	0.60	0.61	0.24	0.43

2.4 Modularity

In [14], Newman proposed using *modularity* as a metric that measures the quality of a division of a network from the global perspective. Such an index has been widely accepted for analyzing community structure in the literature. By using community strength, we propose a (generalized) modularity index that recovers the original Newman's modularity and stability in [21], [22] as special cases. Since community strength is a metric that measures the quality of the structure of a community from the perspective of the nodes inside the community, it seems reasonable to define modularity as the average community strength of the community to which a randomly selected node belongs.

Definition 15. **(Modularity)** Consider a sampled graph $(G(V_g, E_g), p(\cdot,\cdot))$ with a symmetric bivariate distribution $p(\cdot,\cdot)$. Let $\mathcal{P} = \{S_c, c = 1, 2, \dots, C\}$, be a partition of $\{1, 2, \dots, n\}$, i.e., $S_c \cap S_{c'}$ is an empty set for $c \neq c'$ and $\cup_{c=1}^{C} S_c = \{1, 2, \dots, n\}$. The modularity $Q(\mathcal{P})$ with respect to the partition $\mathcal{P} = \{S_c, c = 1, 2, \dots, C\}$ is defined as the weighted average of the community strength of each subset with the weight being the centrality of each subset, i.e.,

$$Q(\mathcal{P}) = \sum_{c=1}^{C} C(S_c) \cdot Str(S_c). \tag{52}$$

As the centrality of a set S_c is the probability that a randomly selected node is in the set S_c, the modularity index Q in (52) is the average community strength of the community to which a randomly selected node belongs. Note that

$$\begin{aligned} Q(\mathcal{P}) &= \sum_{c=1}^{C} C(S_c) \cdot Str(S_c) = \sum_{c=1}^{C} \Big(\mathsf{P}(V \in S_c, W \in S_c) - \mathsf{P}(V \in S_c)\mathsf{P}(W \in S_c) \Big) \\ &= \sum_{c=1}^{C} \sum_{v \in S_c} \sum_{w \in S_c} (p(v,w) - p_V(v)p_W(w)). \end{aligned} \tag{53}$$

It is easy to see that the (generalized) modularity in Definition 15 is in fact a generalization of the original modularity index in [14] by choosing the bivariate distribution $p(v,w)$ in (7), i.e.,

$$\mathsf{P}(V = v, W = w) = \frac{1}{2m} A_{vw}. \tag{54}$$

For such a choice, the modularity index Q in (52) is

$$\sum_{c=1}^{C} \sum_{v \in S_c} \sum_{w \in S_c} \Big(\frac{1}{2m} A_{vw} - \frac{k_v}{2m} \frac{k_w}{2m} \Big). \tag{55}$$

One of the well-known problems of using Newman's modularity in (55) is its resolution limit in detecting communities [62] smaller than a certain scale. This motivated many researchers to propose multi-scale methods, including the *stability* in [21], [22]. In the following example, we show that stability can also be derived as a special case of a sampled graph.

Example 16. **(Stability in [21], [22])** If we choose the bivariate distribution $p(v,w)$ as in (39), i.e.,

$$p(v,w) = p(w,v) = (e^{tL})_{vw} \pi_w = (e^{tL})_{vw} \frac{k_w}{2m}, \tag{56}$$

then the modularity index Q in (52) is simply the stability previously defined in [21], [22], i.e.,

$$\sum_{c=1}^{C} \sum_{v \in S_c} \sum_{w \in S_c} \Big((e^{tL})_{vw} \frac{k_w}{2m} - \frac{k_v}{2m} \frac{k_w}{2m} \Big). \tag{57}$$

Since V and W in this example are the two ends of a randomly selected path via a continuous-time random walk with time t, the parameter t serves as a multi-scale resolution parameter for the size of the communities. Intuitively, a large (resp. small) resolution parameter t tends to identify large (resp. small) communities. Also, it is further illustrated in [21], [22] that stability is in fact a generalization of other multi-scale methods in [81], [82].

3 Community detection algorithms

As the modularity for a partition of a network is the average community strength of a randomly selected node, a good partition of a network should have a large modularity. In view of this, one can then tackle the community detection problem by looking for algorithms that yield large modularity. However, the modularity maximization problem is known to be NP-hard [65] and one has to resort to heuristic algorithms. In this section, we introduce four commonly used modularity maximization algorithms for community detection in *undirected* networks: (i) the spectral modularity maximization algorithm [66], (ii) the hierarchical agglomerative algorithm [14], (iii) the partitional algorithm [67], and (iv) the fast unfolding algorithm [68]. For this, we define the correlation measure below.

Definition 17. **(Correlation)** For a sampled graph $(G(V_g, E_g), p(\cdot,\cdot))$ with a *symmetric* bivariate distribution $p(\cdot,\cdot)$, the correlation measure between two nodes v and w is defined as follows:

$$q(v,w) = p(v,w) - p_V(v)p_V(w) \tag{58}$$

Define the modularity matrix $\mathbf{Q} = (q(v,w))$ be the $n \times n$ matrix with its $(v,w)^{th}$ element being $q(v,w)$. For any two sets S_1 and S_2, define the correlation measure between these two sets as

$$q(S_1,S_2) = \sum_{v\in S_1}\sum_{w\in S_2} q(v,w). \tag{59}$$

The two sets S_1 and S_2 are said to be positively correlated (resp. negatively correlated) if $q(S_1,S_2) \geq 0$ (resp. $q(S_1,S_2) < 0$).

Since $p(v,w)$ is symmetric, the correlation measure $q(v,w)$ is also symmetric. Moreover, we have

$$\sum_{v=1}^{n} q(v,w) = \sum_{w=1}^{n} q(v,w) = 0. \tag{60}$$

With this correlation measure, we have from (53) that the modularity for the partition $\mathcal{P} = \{S_c, c = 1,2,\dots,C\}$ is

$$Q(\mathcal{P}) = \sum_{c=1}^{C} q(S_c,S_c). \tag{61}$$

Moreover, a set S is a community if and only if $q(S,S) \geq 0$.

3.1 The spectral modularity maximization algorithm

The spectral modularity maximization algorithm in [66] tackles the modularity maximization problem by dividing a graph into *two* groups S_1 and S_2. Let $s_v = 1$ (resp. -1) if node v is assigned to group 1 (resp. 2). Note that $\frac{1}{2}(s_v s_w + 1) = 1$ if v and w are in the same group and 0 otherwise. Since $\sum_{v=1}^{n} q(v,w) = \sum_{w=1}^{n} q(v,w) = 0$, we have from (61) that

$$Q(\mathcal{P}) = q(S_1,S_1) + q(S_2,S_2) = \sum_{v=1}^{n}\sum_{w=1}^{n} q(v,w)\frac{1}{2}(s_v s_w + 1) = \frac{1}{2}\sum_{v=1}^{n}\sum_{w=1}^{n} q(v,w)s_v s_w. \tag{62}$$

Thus, the modularity maximization problem can be transformed into the following optimization problem:

$$\begin{aligned} \max \quad & \sum_{v=1}^{n}\sum_{w=1}^{n} q(v,w)s_v s_w \\ \text{s.t.} \quad & s_v = 1 \text{ or } -1,\ v = 1,2,\dots,n. \end{aligned} \tag{63}$$

Instead of using the integer-valued constraints in (63), one can consider using a weaker real-valued constraint $\sum_{v=1}^{n} s_v^2 = n$. This leads to the following (relaxed) optimization problem:

$$\begin{aligned} \max \quad & \sum_{v=1}^{n}\sum_{w=1}^{n} q(v,w)s_v s_w \\ \text{s.t.} \quad & \sum_{v=1}^{n} s_v^2 = n. \end{aligned} \tag{64}$$

By using a single Lagrange multiplier β, the solution of the above optimization problem can be found by

$$\frac{\partial}{\partial s_u}[\sum_{v=1}^{n}\sum_{w=1}^{n} q(v,w)s_v s_w + \beta(n - \sum_{v=1}^{n} s_v^2)] = 0.$$

Thus, for $u = 1, 2, \ldots, n$,

$$\sum_{w=1}^{n} q(u,w)s_w = \beta s_u. \tag{65}$$

Let s be the $n \times 1$ column vector with its v^{th} element being s_v. Writing (65) in the matrix form yields

$$\mathbf{Q}\mathbf{s} = \beta \mathbf{s}. \tag{66}$$

Thus, s is one of the eigenvectors of the modularity matrix **Q** with the corresponding eigenvalue β. Note that

$$\sum_{v=1}^{n}\sum_{w=1}^{n} q(v,w)s_v s_w = \mathbf{s}^T\mathbf{Q}\mathbf{s} = \beta \mathbf{s}^{\mathbf{s}} = n\beta.$$

Clearly, in order to maximize $\sum_{v=1}^{n}\sum_{w=1}^{n} q(v,w)s_v s_w$, one has to maximize β. Thus, the solution s to the optimization problem in (64) is the eigenvector corresponding to the largest eigenvalue of **Q**. Let s* be the eigenvector corresponding to the largest eigenvalue of **Q**. To find a good solution to the original modularity maximization problem in (63), we can simply choose $s_v = 1$ if $s_v^* \geq 0$ and -1 otherwise.

The spectral modularity maximization algorithm relies on finding the eigenvector of the modularity matrix that corresponds to the largest eigenvalue. Even for a sparse network, the computational complexity is $O(n^2)$ [4] and it is still too high for a large network. To find a partition with more than two groups, one can continue bisecting one of the two groups until the modularity cannot be further increased.

3.2 The hierarchical agglomerative algorithm

Analogous to the hierarchical agglomerative algorithms in [14], [68], we introduce a hierarchical agglomerative algorithm for community detection in the probabilistic framework (see Algorithm 1). The algorithm starts from n sets with each node itself as a set. It then recursively merges two disjoint positively correlated sets to form a new set until either there is a single set left or all the remaining sets are not positively correlated. There are two main differences between a standard hierarchical agglomerative algorithm and ours:

(i) Stopping criterion: in a standard hierarchical agglomerative clustering algorithm, such as single linkage or complete linkage, there is no stopping criterion. Here our algorithm stops when all the remaining sets are not positively correlated.

ALGORITHM 1: The Hierarchical Agglomerative Algorithm

```
**Input:** A sampled graph $(G(V_g, E_g), p(\cdot,\cdot))$ with a symmetric bivariate distribution $p(\cdot,\cdot)$.
**Output:** A partition $\mathcal{P} = \{S_c, c = 1, 2, \ldots, C\}$ for some $C \le n$.
**(H1)** Initially, $C = n$; $S_i = \{i\}$, $i = 1, 2, \ldots, n$;
**(H2)** Compute the correlation measures $q(S_i, S_j) = q(\{i\}, \{j\})$ from (58) for all
  $i, j = 1, 2, \ldots, n$;
while there exists some $i$ and $j$ such that $q(S_i, S_j) > 0$ do
    **(H3)** Merge $S_i$ and $S_j$ into a new set $S_k$, i.e., $S_k = S_i \cup S_j$;
        $q(S_k, S_k) = q(S_i, S_i) + 2q(S_i, S_j) + q(S_j, S_j);$    (67)
    for each $\ell \neq k$ do
        $q(S_k, S_\ell) = q(S_\ell, S_k) = q(S_i, S_\ell) + q(S_j, S_\ell);$    (68)
    end
    $C = C - 1$;
end
Reindex the $C$ remaining sets to $\{S_1, S_2, \ldots, S_C\}$;
```

(ii) Greedy selection: our algorithm only needs to select a pair of positively correlated sets to merge. It does not need to be the most positively correlated pair. This could potentially speed up the algorithm in a large data set.

The hierarchical agglomerative algorithm in Algorithm 1 has the following properties. The proof of Theorem 18 is given in Section 9.3.

Theorem 18.

(i) For the hierarchical agglomerative algorithm in Algorithm 1, the modularity is non-decreasing in every iteration and thus converges to a local optimum.
(ii) When the algorithm converges, every set returned by the hierarchical agglomerative algorithm is indeed a *community*.

The output of a hierarchical agglomerative algorithm can be represented by a dendrogram that depicts the order of the merges as a tree. As an illustrating example for the output of the hierarchical agglomerative algorithm, we use the Zachary's karate club friendship network [83] in Figure 7. The set of data was observed by Wayne Zachary [83] over the course of two years in the early 1970s at an American university. During the course of the study, the club split into two groups because of a dispute between its administrator (node 34 in Figure 7) and its instructor (node 1 in Figure 7).

In Figure 8, we show the dendrogram generated by using the hierarchical agglomerative algorithm (with a particular sampled graph). The algorithm stops when there are three communities left, one led by the administrator (node 34), one led by the instructor (node 1), and person number 9 himself. According to [83], there was an interesting story for person number 9. He was a weak supporter for the administrator. However, he was only three weeks away from a test for black belt (master status) when the split of the club occurred. He would have had to give up his rank if he had joined the administrator's club. He ended up with the instructor's club. We also run an additional step for our algorithm (to merge the pair with the largest correlation measure) even though the remaining three communizes are negatively correlated. The additional step reveals that person number 9 is merged into the instructor's club.

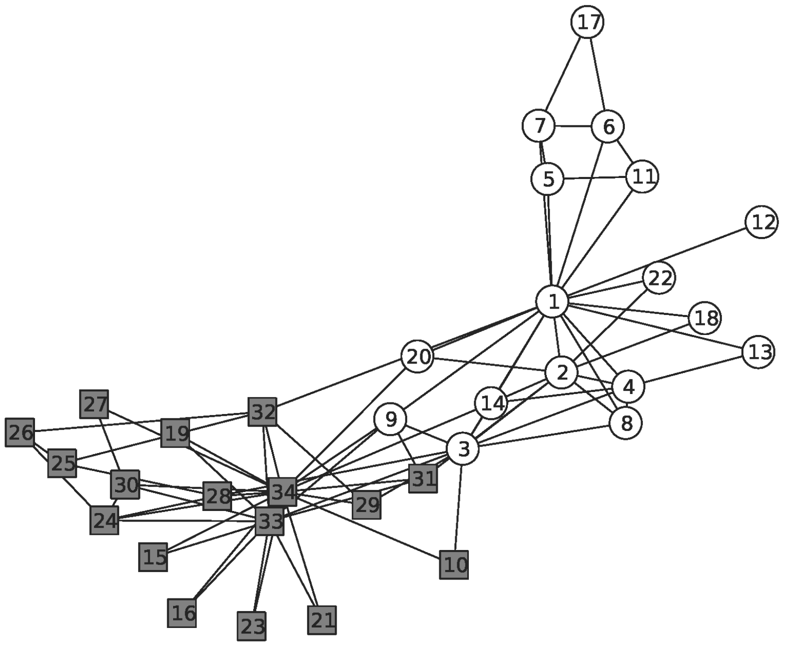


Fig. 7. The network of friendships between individuals in the karate club study of Zachary [83]. The instructor and the administrator are represented by nodes 1 and 34, respectively. Squares represent individuals who ended up with the administrator and circles represent those who ended up with the instructor.

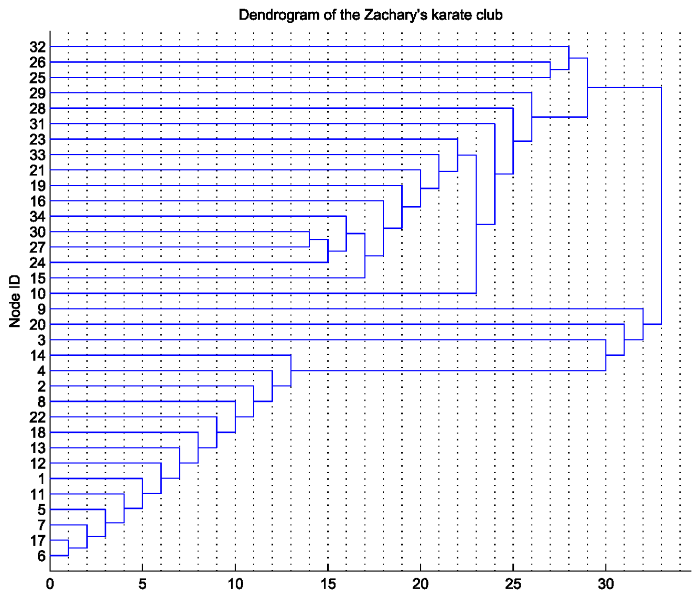


Fig. 8. The dendrogram from the hierarchical agglomerative algorithm for the Zachary karate club friendship network.

For (i) and (ii) of Theorem 18, it is not necessary to specify how we select a pair of two sets with a nonnegative correlation. However, in practice, the greedy selection that selects the two sets with the *largest* correlation measure to merge in (H3) of Algorithm 1 is widely used. Such a selection results in the largest increase of the modularity in each merge. Another greedy selection is to select the two sets with *largest* average correlation measure to merge. One advantage of using such a greedy selection is the *monotonicity* property for the dendrogram produced by a greedy agglomerative clustering algorithm (see [38], Chapter 13.2.3), i.e., the measure for the selection of the two sets in each merge operation is monotonic. With such a monotonicity property, there is no *crossover* in the produced dendrogram. To address the monotonicity property for our hierarchical agglomerative clustering algorithm, we consider the *average* correlation measure between two sets as follows:

Definition 19. **(Average correlation measure)** Define the *average* correlation measure between two sets S_1 and S_2 as

$$\bar{q}(S_1, S_2) = \frac{1}{|S_1| \cdot |S_2|} q(S_1, S_2). \tag{69}$$

To turn our hierarchical agglomerative clustering algorithm into a greedy algorithm, we modify (H3) in the algorithm by selecting the two sets with the *largest* average correlation measure. In the following corollary, we show the monotonicity property of our greedy hierarchical agglomerative clustering algorithm. Its proof is given in Section 9.4.

Corollary 20. For the *greedy* hierarchical agglomerative clustering algorithm that selects the two sets with *largest* average correlation measure to merge in (H3) of Algorithm 1, the average correlation measure of the two selected sets in each merge operation is non-increasing.

Even though the hierarchical agglomerative algorithm in Algorithm 1 produces communities, there are still two drawbacks:
(i) The hierarchical agglomerative algorithm only uses the "merge" operation to increase the modularity. As such, once a "merge" operation is done, there is no way to reverse it. As such, the order of the "merge" operations might have a serious effect on the final outcome.
(ii) In the initialization stage, every node is assumed to be a community itself. There are roughly $O(n)$ communities at the early stage of the hierarchical agglomerative algorithm. Thus, its computational complexity is high at the early stage of the algorithm. For a graph with n nodes, it is well-known (see e.g., [4]) that the computational complexity of a näive implementation is $O(n^3)$ [14] for a greedy hierarchical agglomerative algorithm and it can be reduced to $O(n^2 \log n)$ (or $O(m \log n)$ in a sparse graph) by implementing priority queues [84].

3.3 The partitional algorithm

As mentioned in the previous section, there are two drawbacks of the hierarchical agglomerative algorithm in Algorithm 1. To tackle these problems, we introduce the partitional algorithm in Algorithm 2 that has a much lower computational complexity. The partitional algorithm also allows nodes to move from communities to communities.

We first consider another correlation measure $q_0(\cdot, \cdot)$ from the original correlation measure $q(\cdot, \cdot)$ in (58) by letting $q_0(v, w) = q(v, w)$ for $v \neq w$ and $q_0(v, w) = 0$ for $v = w$. Also let

$$q_0(S_1, S_2) = \sum_{v \in S_1} \sum_{w \in S_2} q_0(v, w). \tag{70}$$

In view of (61), we have for any partition $\mathcal{P} = \{S_c, c = 1, 2, \ldots, C\}$ that

$$Q(\mathcal{P}) = \sum_{c=1}^{C} q(S_c, S_c) = \sum_{c=1}^{C} q_0(S_c, S_c) + \sum_{v=1}^{n} q(v, v). \tag{71}$$

Thus, maximizing the modularity is equivalent to maximizing $\sum_{c=1}^{C} q_0(S_c, S_c)$.

To describe the partitional algorithm in Algorithm 2, we let

$$\text{Nei}(v) = \{w : p(v, w) > 0\} \tag{72}$$

ALGORITHM 2: The Partitional Algorithm

```
Input: A sampled graph $(G(V_g, E_g), p(\cdot,\cdot))$ with a symmetric bivariate distribution $p(\cdot,\cdot)$ and
       the (maximum) number of communities $K$.
Output: A partition $\mathcal{P} = \{S_c, c = 1, 2, \dots, C\}$ for some $C \le K$.
(P0) Initially, choose arbitrarily $K$ disjoint nonempty sets $S_1, \dots, S_K$ as a partition of the
  sampled graph $(G(V_g, E_g), p(\cdot,\cdot))$.
(P1) for $v = 1, 2, \dots, n$ do
    for each neighboring set $S_j$ of node $v$ do
        Compute the correlation measures $q_0(v, S_j)$.
    end
    Find the neighboring set to which node $v$ has the largest correlation measure. Assign
      node $v$ to that set.
end
(P2) Repeat from (P1) until there is no further change.
```

be the set that contains the "neighboring" nodes of v, where $p(v, w)$ is the symmetric bivariate distribution. A set S is called a neighboring set of a node v if there exists a node $w \in S$ such that $p(v, w) > 0$.

The partitional algorithm in Algorithm 2 starts from a (random) initial partition of the nodes into K disjoint sets. In each iteration, every node is presented and assigned to one of its neighboring set to which it has the largest correlation measure (in terms of $q_0(\cdot,\cdot)$). This process is then repeated until there is no further change of the partition. It might happen (in fact very often) that some sets of the initial partition become empty when all their set members are moved to other sets. Intuitively, the "merge" operation that is used in the hierarchical agglomerative algorithm is done in a microscopic manner in the partitional algorithm. As such, the chance for the partitional algorithm to be trapped in a bad "merge" operation is smaller than that for the hierarchical agglomerative algorithm.

A video demo of the results from the partitional algorithm to detect six clusters on a plane from an initial random partition can be found at `http://imgur.com/b5KQo9I`.

The partitional algorithm has the following property and its proof is given in Section 9.5.

Theorem 21. In the partitional algorithm in Algorithm 2, the modularity is non-decreasing when there is a change, i.e., a node is moved from one set to another. Thus, the algorithm converges to a local optimum of the modularity in a finite number of steps.

Unlike the hierarchical agglomerative algorithm in Algorithm 1, the sets returned by the partitional algorithm in Algorithm 2 may not be communities. As such, one can use the output of the partitional algorithm as the input of the hierarchical agglomerative algorithm so that the final outputs are communities. Such a two-step approach is called the partitional-hierarchical algorithm.

Now we turn our attention to the computation complexity of the partitional algorithm in Algorithm 2. Let m be the total number of nonzero entries in the $n \times n$ matrix $P = (p(v, w))$. Even though the total number of nonzero entries in the modularity matrix Q is $O(n^2)$, we will show that the computation complexity of the partitional algorithm is only $O((n + m)I)$, where I is the number of iterations in Algorithm 2. This is done with the memory complexity $O(n + m)$. Thus, Algorithm 2 is a linear time algorithm for a sparse graph with $m = O(n)$ and it is much more efficient than the hierarchical agglomerative algorithm for large sparse graphs.

Let

$$p(S_1, S_2) = \sum_{v \in S_1} \sum_{w \in S_2} p(v, v), \text{ and} \tag{73}$$

$$p_V(S) = \sum_{v \in S} p_V(v). \tag{74}$$

To store the nonzero entries of the $n \times n$ matrix $P = (p(v, w))$ requires $O(m)$ memory complexity. In addition to this, we also store $q(v, v)$, $p_V(v)$ for all $v = 1, 2, \ldots, n$, and $p_V(S_k)$ for all $k = 1, 2, \ldots, K$. Thus, the memory complexity is $O(n+m)$. Now suppose node v is moved from S_1 to S_2. As $p_V(v)$ for all $v = 1, 2, \ldots, n$, are all stored in the memory, one can perform the following updates:

$$p_V(S_1) \Leftarrow p_V(S_1) - p_V(v), \text{ and}$$
$$p_V(S_2) \Leftarrow p_V(S_2) + p_V(v).$$

These updates can be done in $O(1)$ steps. Thus, the total number of updates in each iteration of Algorithm 2 requires $O(n)$ steps.

Now we argue that the computational complexity for the correlation measures in each iteration is $O(n + m)$. Note that

$$\begin{aligned} q(v, S_k) &= \sum_{w \in S_k} q(v, w) \\ &= p(v, S_k) - p_V(v) p_V(S_k) \end{aligned}$$

and that

$$\begin{aligned} p(v, S_k) &= \sum_{w \in S_K} p(v, w) \\ &= \sum_{w \in S_k \cap \text{Nei}(v)} p(v, w). \end{aligned}$$

As now $p_V(v)$ and $p_V(S_k)$ are stored in the memory and there are $|\text{Nei}(v)|$ "neighboring" nodes of v, one can compute the correlation measures $q(v, S_k)$ for all the neighboring sets S_k of node v in $|\text{Nei}(v)|$ steps. The correlation measure $q_0(v, S_k)$ can be computed by using $q(v, S_k)$ for $v \notin S_k$ and $q(v, S_k) - q(v, v)$ for $v \in S_k$. Thus, the computational complexity for the correlation measures for the n nodes in each iteration is $O(n + m)$.

One possible initial partition for the partitional algorithm is to choose $K = n$ and every node is in a different set. Such a choice was in fact previously used in the fast unfolding algorithm in [68]. The fast unfolding algorithm is also a modularity maximization algorithm and it consists of two-phases: the first phase is also a partitional algorithm like the one presented here. The difference is that our partitional algorithm uses the probabilistic framework. Such a framework not only provides the needed physical insights but also allows a much more general treatment for modularity maximization. Once the first phase is completed, there is a second phase of building a new (and much smaller) network whose nodes are the communities found during the previous phase. This will be addressed in the next section.

3.4 Node aggregation

In this section, we introduce the concept of node aggregation that transforms a sampled graph into another smaller sampled graph by aggregating several sets of the nodes in the original sampled graphs into giant nodes. Such a transformation is modularity preserving as it preserves the modularity.

Definition 22. **(Node aggregation)** Consider a sampled graph $(G(V_g, E_g), p(\cdot,\cdot))$ with a symmetric bivariate distribution $p(\cdot,\cdot)$, and a partition $\mathcal{P} = \{S_c, c = 1, 2, \ldots, C\}$ of the sampled graph $(G(V_g, E_g), p(\cdot,\cdot))$. Now we aggregate the nodes in S_c into a giant node c, $c = 1, 2, \ldots, C$, and define the bivariate distribution $\hat{p}(\cdot,\cdot)$ with

$$\hat{p}(c_1, c_2) = \sum_{u \in S_{c_1}} \sum_{v \in S_{c_2}} p(u, v), \tag{75}$$

for $c_1, c_2 = 1, 2, \ldots, C$. Let $\hat{V}$ be the collection of the giant nodes c, $c = 1, 2, \ldots, C$, and $\hat{E}$ be the collection of pairs (c_1, c_2) with $\hat{p}(c_1, c_2) > 0$. The new sampled graph $(G(\hat{V}, \hat{E}), \hat{p}(\cdot,\cdot))$ is called the $\mathcal{P}$-aggregated sampled graph of the sample graph $(G(V_g, E_g), p(\cdot,\cdot))$.

It is clear from Definition 22 that the new sampled graph $(G(\hat{V}, \hat{E}), \hat{p}(\cdot,\cdot))$ also has a symmetric bivariate distribution $\hat{p}(\cdot,\cdot)$. In the following lemma, we further show that the node aggregation in Definition 22 is a modularity preserving transformation. Its proof is given in Section 9.6.

Lemma 23. Consider a sampled graph $(G(V_g, E_g), p(\cdot,\cdot))$ with a symmetric bivariate distribution $p(\cdot,\cdot)$ and a partition $\mathcal{P} = \{S_c, c = 1, 2, \ldots, C\}$ of the sampled graph $(G(V_g, E_g), p(\cdot,\cdot))$. Suppose $\mathcal{P}^a = \{S_k^a, k = 1, 2, \ldots, K\}$ for some $K \le C$ is a partition of the $\mathcal{P}$-aggregated sampled graph of the sample graph $(G(V_g, E_g), p(\cdot,\cdot))$. Then $\mathcal{P}^a$ induces a partition $\mathcal{P}^o = \{S_k^o, k = 1, 2, \ldots, K\}$ on the sampled graph $(G(V_g, E_g), p(\cdot,\cdot))$, where

$$S_k^o = \cup_{c \in S_k^a} S_c, \tag{76}$$

for $k = 1, 2, \ldots, K$. Let $\hat{Q}(\mathcal{P}^a)$ be the modularity of the partition $\mathcal{P}^a$ of the $\mathcal{P}$-aggregated sampled graph and $Q(\mathcal{P}^o)$ be the modularity of the induced partition $\mathcal{P}^o$ of the sampled graph $(G(V_g, E_g), p(\cdot,\cdot))$. Then

$$\hat{Q}(\mathcal{P}^a) = Q(\mathcal{P}^o). \tag{77}$$

Note from (75) that the number of nonzero entries in the $C \times C$ matrix $\hat{P} = (\hat{p}(\cdot,\cdot))$ is not larger than the number of nonzero entries in the $n \times n$ matrix $P = (p(\cdot,\cdot))$. Thus, such a transformation is also sparsity preserving.

3.5 The fast unfolding algorithm

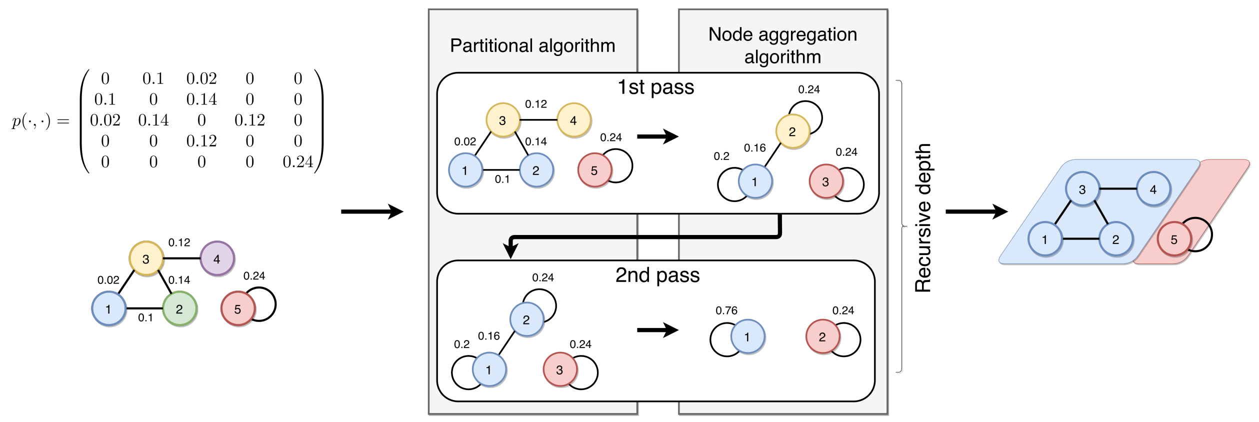
The outputs of the partitional algorithm may not be *communities* when it converges. For this, we consider another phase that uses node aggregation as in the fast unfolding algorithm in [68].

As shown in Lemma 23, node aggregation is a modularity preserving and sparsity preserving transformation. The modularity preserving property in Lemma 23 allows us to further increase the modularity by applying the partitional algorithm in Algorithm 2 again with the new sampled graph after node aggregation as its input. In fact, one can do this two step process, i.e., the partitional algorithm and the node aggregation algorithm, iteratively until the modularity cannot be further increased. This then leads to a recursive modularity maximization algorithm, called the (recursive) fast unfolding algorithm. The details of the fast unfolding algorithm is outlined in Algorithm 3.

In Figure 9, we provide an illustrating example of the fast unfolding algorithm. After the first pass of the partitional algorithm, nodes 1 and 2 are aggregated into a giant node. Also, nodes 3 and 4 are aggregated into another giant node. After node aggregation (and relabelling), we have a sampled graph with three nodes. Now run the partitional algorithm again with this new sampled graph leads to the aggregation of node 1 and node 2 into a giant node. As the modularity cannot be further improved with the two two-node sampled graph, the fast unfolding algorithm stops and return the induced partition,

ALGORITHM 3: The fast unfolding (Louvain) algorithm

```
Input: A sampled graph $(G(V_g, E_g), p(\cdot,\cdot))$ with a symmetric bivariate distribution $p(\cdot,\cdot)$.
Output: A partition $\mathcal{P} = \{S_c, c = 1, 2, \ldots, C\}$ for some $C \le n$.
(F0) Initially, choose the partition $\mathcal{P}$ with $C = n$ and $S_i = \{i\}$, $i = 1, 2, \ldots, n$;
(F1) Run the partitional algorithm in Algorithm 2 with the initial partition $\mathcal{P}$. Let
  $\mathcal{P}' = \{S_1, S_2, \ldots, S_C\}$ be its output partition.
(F2) If $\mathcal{P} = \mathcal{P}'$, then there is no improvement and return $\mathcal{P}$ as the output partition.
(F3) Otherwise, do node aggregation to generate the $\mathcal{P}'$-aggregated sampled graph of the
  sampled graph $(G(V_g, E_g), p(\cdot,\cdot))$. Run the fast unfolding algorithm in Algorithm 3 with the
  $\mathcal{P}'$-aggregated sampled graph as its input. Suppose it outputs the partition
  $\mathcal{P}^a = \{S^a_k, k = 1, 2, \ldots, K\}$ on the $\mathcal{P}'$-aggregated sampled graph. Return the induced
  partition $\mathcal{P}^o = \{S^o_k, k = 1, 2, \ldots, K\}$ on the sampled graph $(G(V_g, E_g), p(\cdot,\cdot))$, where
  $S^o_k = \cup_{c \in S^a_k} S_c$.
```



$p(\cdot,\cdot) = \begin{pmatrix} 0 & 0.1 & 0.02 & 0 & 0 \\ 0.1 & 0 & 0.14 & 0 & 0 \\ 0.02 & 0.14 & 0 & 0.12 & 0 \\ 0 & 0 & 0.12 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.24 \end{pmatrix}$
Partitional algorithm
Node aggregation algorithm
1st pass
2nd pass
Recursive depth

Fig. 9. An illustrating example of the fast unfolding algorithm and node aggregation.

Since the transformation by node aggregation is also sparsity preserving, the computational complexity of each recursive call of Algorithm 3 is $O(m+n)$ (as in the partitional algorithm). Thus, the overall computational complexity depends on the depth of the recursive calls. A rigourous proof for the upper bound on the depth of the recursive calls appears to be difficult. The original fast unfolding algorithm is a special case of ours (by using the uniform edge sampling) and it is generally conjectured to be $O(\log m)$ for the depth of the recursive calls. But we do not have a formal proof for that. However, in our experiments, the depth is quite small.

We have the following properties for Algorithm 3. Its proof is given in Section 9.7.

Theorem 24.

(i) In the fast unfolding algorithm in Algorithm 3, the modularity is non-decreasing when there is a change of the partition. Thus, the algorithm converges to a local optimum of the modularity in a finite number of steps.

(ii) When the algorithm converges, every set returned by Algorithm 3 is indeed a *community*.

The intuition behind the fast unfolding algorithm is to increase the modularity by moving a single node one at a time at the first level and then moving a block of nodes in each recursive level. As it gets deeper, the block size is larger. The fast unfolding algorithm stops at its deepest level when no more blocks of nodes can be moved to increase the modularity. In view of this, it is not necessary to start the fast unfolding algorithm from the initial partition that contains n sets with each node in a set. The modularity can be further increased by running the fast unfolding algorithm again with its first output partition as its input partition. However, the gain from doing another iteration of the fast unfolding algorithm is in general quite limited. Also, as pointed out in [68], two tricks to reduce the computational cost are (i) to stop the partitional algorithm when the gain of modularity is below a certain threshold, and (ii) removing the nodes with degree 1 and then adding them back later.

In comparison with the original fast unfolding algorithm in [68], our fast unfolding algorithm has the following advantages:
(i) Generalization: based on our probabilistic framework, our fast unfolding algorithm provides users the freedom to choose the sampled graph that might be more suitable for their viewpoints and resolution requirements.
(ii) Theoretical guarantees: our framework provides a rigorous proof for the convergence of the fast unfolding algorithm through modularity maximization and node aggregation. Moreover, when the algorithm converges, it returns *communities* that are formally defined in our framework.
(iii) Probabilistic insights: our probabilistic framework defines the concept of *correlation*. The partitional algorithm in the first phase of the fast unfolding algorithm is simply to *repeatedly* move each node to the most positively correlated set. Also, the variables that need to be stored in the memory have clear probabilistic meanings that can then be used for speeding up the computation of correlation. More importantly, the node aggregation phase can be understood as a modularity preserving and sparsity preserving transformation.
(iv) Reduction of computational cost: by setting the diagonal elements of the modularity matrix to be 0, the computation for the largest increase of modularity, ΔQ in [68], is now reduced to the computation of the largest correlation of a node to its neighboring sets.

3.6 Post-processing for weak communities and outliers

If the output partition of the fast unfolding algorithm is satisfactory, then we can simply stop there. If otherwise, we can perform the following post-processing to find another and possibly better initial partition. The idea of post-processing is to examine the contribution of each community to the modularity and reassign the nodes in the communities with small contributions. Specifically, suppose that the fast unfolding algorithm returns a partition of C communities $S_1, S_2, \ldots, S_C$. The contribution of S_c to the modularity is then $q(S_c, S_c)$, where $q(\cdot,\cdot)$ is defined in (59). Then we can sort $q(S_c, S_c)$, $c = 1, 2, \ldots, C$ in the ascending order and perform a "sweep" to cut these C communities (at the maximum gap) into *strong* communities and *weak* communities. Suppose that the strong communities are $S'_1, \ldots, S'_K$ with $K < C$. The nodes in the weak communities are then *iteratively* reassigned to the most *positively correlated* strong community, i.e., node i is assigned to S'_j if $q(i, S'_j) = \max_{1 \le \ell \le K} q(i, S'_\ell)$ and $q(i, S'_j) > 0$. After this is done, we have K communities, $S''_1, \ldots, S''_K$ with $S'_\ell \subset S''_\ell$, $\ell = 1, 2, \ldots, K$. There might be still some nodes that are *negatively correlated* to S''_ℓ for all $\ell = 1, 2, \ldots, K$. The set of nodes $\{i : q(i, S''_\ell) \le 0, \ell = 1, 2, \ldots, K\}$ can be classified as *outliers*. For outliers, we can either remove them from the graph or reassign them to the most correlated strong community. In either case, we have a new initial partition that we can use for another run of the fast unfolding algorithm.

4 The probabilistic framework for directed networks

In the previous section, the bivariate distributions are all assumed to be *symmetric* and that limits its applicability to *undirected* networks. In this section, we relax the *symmetric* assumption for the bivariate distribution so that the framework can be applied to *directed* networks. The key idea of doing this is to assume that the bivariate distribution has *the same marginal distributions*, i.e.,

$$p_V(v) = p_W(v), \quad \text{for all } v. \tag{78}$$

As pointed out in [85], the bivariate distribution that has the same marginal distributions, called *circulation* in [85], plays an important role in analyzing directed networks. Note that a symmetric bivariate distribution has the same marginal distributions and thus the assumption in (78) is much more general.

4.1 Sampling a directed network

One approach for sampling a network with a bivariate distribution that has the same marginal distributions is to sample a network by an *ergodic Markov chain*. From the Markov chain theory (see e.g., [79]), it is well-known that an ergodic Markov chain converges to its steady state in the long run. Hence, the joint distribution of two successive steps of a *stationary and ergodic* Markov chain can be used as the needed bivariate distribution. Specifically, suppose that a network $G(V_g, E_g)$ is sampled by a stationary and ergodic Markov chain $\{X(t), t \geq 0\}$ with the state space $\{1, 2, \ldots, n\}$ being the n nodes in V_g. For this Markov chain, let p_{ij} be the transition probability from state i to state j and π_i be the steady state probability of state i. Then we can choose the bivariate distribution

$$\begin{aligned} \mathsf{P}(V = v, W = w) &= p(v, w) \\ &= \mathsf{P}(X(t) = v, X(t+1) = w). \end{aligned} \tag{79}$$

As the Markov chain is stationary, we have

$$\mathsf{P}(X(t) = v) = \mathsf{P}(X(t+1) = v) = p_V(v) = p_W(v). \tag{80}$$

It is well-known that a random walk on the graph induces a Markov chain with the following state transition probabilities:

$$p_{ij} = \frac{a_{ij}}{k_i^{out}}, \tag{81}$$

where

$$k_i^{out} = \sum_{j=1}^{n} a_{ij}, \tag{82}$$

is the number of outgoing edges from vertex i (here we assume that $k_i^{out} \geq 1$ for all i). In particular, if the graph is an undirected graph, i.e., $a_{ij} = a_{ji}$, then the induced Markov chain is reversible and the steady state probability of state i, i.e., π_i, is $k_i/2m$, where $m = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}$ is the total number of edges of the undirected graph.

Example 25. **(PageRank)** One problem for sampling a *directed* network by a simple random walk is that the induced Markov chain may not be ergodic even when the network itself is weakly connected. One genuine solution for this is to allow random jumps from states to states in a random walk. PageRank [5], proposed by Google, is one such example that has been successfully used for ranking web pages. The key idea behind PageRank is to model the behavior of a web surfer by a random walk (the random surfer model) and then use that to compute the steady state probability for a web surfer to visit a specific web page.

Specifically, suppose that there are n web pages and a web surfer uniformly selects a web page with probability $1/n$. Once he/she is on a web page, he/she continues web surfing with probability λ. This is done by selecting *uniformly* one of the hyperlinks in that web page. On the other hand, with probability $1-\lambda$ he/she starts a new web page *uniformly* among all the n web pages. The transition probability from state i to state j for the induced Markov chain is then

$$p_{ij} = (1-\lambda)\frac{1}{n} + \lambda\frac{a_{ij}}{k_i^{out}}, \tag{83}$$

where $a_{ij} = 1$ if there is a hyperlink pointing from the i^{th} web page to the j^{th} web page and $k_i^{out} = \sum_{j=1}^n a_{ij}$ is the total number of hyperlinks on the i^{th} web page. Let π_i be steady probability of visiting the i^{th} web page by the web surfer. It then follows that

$$\pi_i = (1-\lambda)\frac{1}{n} + \lambda\sum_{j=1}^n \frac{a_{ji}}{k_j^{out}}\pi_j. \tag{84}$$

PageRank then uses π_i as the centrality of the i^{th} web page and rank web pages by their centralities. Unlike the random walk on an undirected graph, the steady state probabilities in (84) cannot be explicitly solved and it requires a lot of computation to solve the system of linear equations. The sampled graph $(G(V_g, E_g), p(\cdot,\cdot))$ by using PageRank then has the following bivariate distribution

$$p(v,w) = \pi_v p_{vw}, \tag{85}$$

where p_{vw} is defined in (83) and π_v is the solution of (84). Such a sampled graph was called LinkRank in [86].

Example 26. **(Random walks with self-loops and backward jumps)** Another way to look at the Markov chain induced by PageRank in (83) is that it is in fact a random walk on a different graph with the adjacency matrix $\tilde{A}$ that is constructed from the original graph with additional edge weights, i.e.,

$$\tilde{A} = (1-\lambda)\frac{1}{n}\mathbf{1} + \lambda D^{-1}A, \tag{86}$$

where $\mathbf{1}$ is an $n\times n$ matrix with all its elements being 1 and $D=(d_{ij})$ is the diagonal matrix with $d_{ii} = k_i^{out}$ for all $i = 1, 2, \ldots, n$.
In view of (86), another solution for the ergodic problem is to consider a random walk on the graph with the adjacency matrix

$$\hat{A} = \lambda_0\mathbf{I} + \lambda_1 A + \lambda_2 A^T, \tag{87}$$

where $\mathbf{I}$ is the $n\times n$ identity matrix and A^T is the transpose matrix of A. The three parameters $\lambda_0, \lambda_1, \lambda_2$ are positive and

$$\lambda_0+\lambda_1+\lambda_2 = 1.$$

A random walk on the graph with the adjacency matrix $\hat{A}$ induces an ergodic Markov chain if the original graph is weakly connected. Also, with the additional edges from the identity matrix and the transpose matrix, such a random walk can be intuitively viewed as a random walk on the original graph with self-loops and backward jumps.

Example 27. **(Diffusion)** In addition to random walks, one can also consider using *diffusion* in a network to obtain the corresponding Markov chain. Specifically, let $k_{\max}^{out} = \max_{1\le i\le n} k_i^{out}$, $\mathbf{I}$ be the

$n \times n$ identity matrix and $D = (d_{ij})$ be the diagonal matrix with $d_{ii} = k_i^{out}$ for all $i = 1, 2, \ldots, n$. The transition probability matrix is then $\mathbf{I} - \alpha(D - A)$, where $\alpha \leq 1/k_{\max}^{out}$ is a normalization coefficient and $D - A$ is the graph Laplacian. For other Laplacian related methods, we refer to the survey paper [69].

In general, one can generate a bivariate distribution that has the same marginal distributions by using a *stationary* and *ergodic* stochastic process, e.g., a semi-Markov process or a hidden Markov process.

4.2 Centralities and relative centralities in directed networks

Under the assumption that the bivariate distribution $p(\cdot, \cdot)$ has the same marginal distributions, the definitions for centralities and relative centralities for directed networks are exactly the same as those for undirected networks.

Definition 28. **(Centrality)** For a sampled graph $(G(V_g, E_g), p(\cdot, \cdot))$ with the bivariate distribution $p(\cdot, \cdot)$ that has the same marginal distributions in (78), the *centrality* of a set of nodes S, denoted by $C(S)$, is defined as the probability that a node in S is selected, i.e.,

$$C(S) = \mathsf{P}(V \in S) = \mathsf{P}(W \in S). \tag{88}$$

Definition 29. **(Relative centrality)** For a sampled graph $(G(V_g, E_g), p(\cdot, \cdot))$ with the bivariate distribution $p(\cdot, \cdot)$ that has the same marginal distributions in (78), the *relative centrality* of a set of nodes S_1 with respect to another set of nodes S_2, denoted by $C(S_1|S_2)$, is defined as the conditional probability that the randomly selected node W is inside S_1 given that the randomly selected node V is inside S_2, i.e.,

$$C(S_1|S_2) = \mathsf{P}(W \in S_1 | V \in S_2). \tag{89}$$

Example 30. **(Relative PageRank)** PageRank defined in Example 25 has been commonly used for ranking the importance of nodes in a directed network. Here we can use Definition 29 to define relative PageRank that can be used for ranking the relative importance of a set of nodes to another set of nodes in a directed network. Specifically, let π_i be the PageRank for node i in (84) and $p_{i,j}$ be the transition probability from state i to state j for the induced Markov chain in (83). Then the relative PageRank of a set S_1 with respect to another set S_2 is

$$\begin{aligned} C(S_1|S_2) &= \mathsf{P}(W \in S_1 | V \in S_2) \\ &= \frac{\mathsf{P}(W \in S_1, V \in S_2)}{\mathsf{P}(V \in S_2)} = \frac{\sum_{i \in S_2} \sum_{j \in S_1} \pi_i p_{ij}}{\sum_{i \in S_2} \pi_i}. \end{aligned} \tag{90}$$

4.3 Community and modularity in directed networks

Once again, under the assumption that the bivariate distribution $p(\cdot, \cdot)$ has the same marginal distributions, the definitions for community and modularity in directed networks are the same as those in undirected networks.

Definition 31. **(Community strength and communities)** For a sampled graph $(G(V_g, E_g), p(\cdot, \cdot))$ with a bivariate distribution $p(\cdot, \cdot)$ that has the same marginal distributions in (78), the *community strength* of a set of nodes $S \subset V_g$, denoted by $Str(S)$, is defined as the difference of the relative centrality of S with respect to itself and its centrality, i.e.,

$$Str(S) = C(S|S) - C(S). \tag{91}$$

In particular, if a subset of nodes $S \subset V_g$ has a nonnegative community strength, i.e., $Str(S) \geq 0$, then it is called a *community*.

Under Definition 31, one can easily show that the equivalent statements for a community in Theorem 12 still holds.

***Definition* 32. (Modularity)** Consider a sampled graph $(G(V_g, E_g), p(\cdot, \cdot))$ with a bivariate distribution $p(\cdot, \cdot)$ that has the same marginal distributions in (78). Let $\mathcal{P} = \{S_c, c = 1, 2, \ldots, C\}$, be a partition of $\{1, 2, \ldots, n\}$, i.e., $S_c \cap S_{c'}$ is an empty set for $c \neq c'$ and $\cup_{c=1}^{C} S_c = \{1, 2, \ldots, n\}$. The modularity $Q(\mathcal{P})$ with respect to the partition S_c, $c = 1, 2, \ldots, C$, is defined as the weighted average of the community strength of each subset with the weight being the centrality of each subset, i.e.,

$$Q(\mathcal{P}) = \sum_{c=1}^{C} C(S_c) \cdot Str(S_c). \tag{92}$$

As before, the modularity in (92) can also be written as follows:

$$\begin{aligned} Q(\mathcal{P}) &= \sum_{c=1}^{C} \mathsf{P}(V \in S_c, W \in S_c) - \mathsf{P}(V \in S_c)\mathsf{P}(W \in S_c) \\ &= \sum_{c=1}^{C} \sum_{v \in S_c} \sum_{w \in S_c} (p(v, w) - p_V(v)p_W(w)). \end{aligned} \tag{93}$$

For sampled graphs with *symmetric* bivariate distributions, there are already various community detection algorithms in Section 3 that find local maxima of the modularity. However, they cannot be directly applied as the bivariate distributions for sampling directed networks could be *asymmetric*. For this, we introduce a modularity preserving transformation that transforms a sampled graph that has the same marginal distributions to another sampled graph that has a symmetric bivariate distribution. The proof of Lemma 33 is given in Section 9.8.

***Lemma* 33.** Consider a sampled graph $(G(V_g, E_g), p(\cdot, \cdot))$ with a bivariate distribution $p(\cdot, \cdot)$ that has the same marginal distributions in (78). Construct the sampled graph $(G(V_g, E_g), \tilde{p}(\cdot, \cdot))$ with the symmetric bivariate distribution

$$\tilde{p}(v, w) = \frac{p(v, w) + p(w, v)}{2}. \tag{94}$$

Let $Q(\mathcal{P})$ (resp. $\tilde{Q}(\mathcal{P})$) be the modularity for the partition $\mathcal{P} = \{S_c, c = 1, 2, \ldots, C\}$ of the sampled graph $(G(V_g, E_g), p(\cdot, \cdot))$ (resp. the sampled graph $(G(V_g, E_g), \tilde{p}(\cdot, \cdot))$). Then

$$\tilde{Q}(\mathcal{P}) = Q(\mathcal{P}). \tag{95}$$

We note the transformation in Lemma 33 is also sparsity preserving. Specifically, let m_0 be the total number of nonzero entries in the $n \times n$ matrix $P = (p(v, w))$. Then the number of nonzero entries in the $n \times n$ matrix $\tilde{P} = (\tilde{p}(v, w))$ is at most $2m_0$. Such a sparsity property is crucial for reducing the computational complexity of the community detection algorithms described in Section 3.

4.4 General bivariate distribution

In this section, we discuss how to extend our framework of sampled graphs to the setting with general bivariate distributions. As the marginal distributions may not be the same, many results derived in the previous section are not as elegant as before. For instance, now we have to distinguish the notion of *centrality* into two types: the out-centrality (with respect to the marginal distribution of V) and the in-centrality (with respect to the marginal distribution of W). The notion of *relative centrality* also needs to be extended in the same manner. However, even with such an extension, the reciprocity property in (23) is not true any more. To see this, let us consider the uniform edge sampling method in a directed network. For such a sampling method, the bivariate distribution is $p(v,w) = a_{vw}/m$, where m is the total number of edges in the directed network. Then $p_V(v) = k_v^{out}/m$ is the out-degree centrality of node v and $p_W(w) = k_w^{in}/m$ the in-degree centrality of node w.

The good news is that the notions of *community* and *modularity* can be extended in the same manner as before. There is no need to distinguish an in-community and an out-community. Specifically, we can still define a set S as a community if

$$\mathsf{P}(V \in S, W \in S) - \mathsf{P}(V \in S)\mathsf{P}(W \in S) \geq 0,$$

and define the modularity $Q(\mathcal{P})$ with respect to the partition S_c, $c = 1, 2, \ldots, C$, by using (53). However, the equivalent characterizations for a community in Theorem 12 are no long valid as the reciprocity property in (23) is not true any more.

With the general definition of the modularity, one can still apply the hierarchical agglomerative algorithm in Algorithm 1, the partitional algorithm in Algorithm 2, and the fast unfolding algorithm in Algorithm in 3 for community detection in a sampled graph with a general bivariate distribution. Moreover, the results in Theorem 18, Theorem 21, and Theorem 24 also hold. However, there are two minor modifications:
(i) One needs to generalize the definition of the correlation measure between two nodes v and w in (58) as follows:

$$q(v,w) = \frac{p(v,w) - p_V(v)p_W(w) + p(w,v) - p_V(w)p_W(v)}{2}. \tag{96}$$

Then use $q(v,w)$ in (96) to construct the correlation matrix. Also, for any two sets S_1 and S_2, the correlation measure between these two sets, i.e., $q(S_1,S_2)$, is still defined as in (59). By doing so, a set S is community if and only if $q(S,S) \geq 0$ and the modularity for a partition $\mathcal{P} = \{S_c, c = 1, 2, \ldots, C\}$ can still be computed by $\sum_{c=1}^{C} q(S_c,S_c)$ as described in (61).
(ii) The definition of "neighboring" nodes of a node v in (72) should be generalized as follows:

$$\text{Nei}(v) = \{w : p(v,w) > 0 \text{ or } p(w,v) > 0\}. \tag{97}$$

Also, a set S is called a "neighboring" set of a node v if there exists a node $w \in S$ such that w is a "neighboring" node of v. In order to show that the partitional algorithm is still a linear-time algorithm, one needs to store in memory the nonzero entries of the $n \times n$ matrix $P = (p(v,w))$, $q(v,v)$, $p_V(v)$, $p_W(v)$ for all $v = 1, 2, \ldots, n$, and $p_V(S_k)$, $p_W(S_k)$ for all $k = 1, 2, \ldots, K$. Then use those to compute the correlation measure

$$\begin{aligned} &q(v,S_k) \\ &= \frac{p(v,S_k) + p(S_k,v) - p_V(v)p_W(S_k) - p_W(v)p_V(S_k)}{2}. \end{aligned}$$

There are two different types of communities for directed networks that are commonly addressed in the literature (see e.g., [87], [88]): structural communities (based on the densities of edges) and flow communities (based on the flow of probabilities). These two types of communities correspond to two different types of sampled graphs in our probabilistic framework. For flow communities, they can be detected by using random walks as a random walker tends to be ”trapped” in a community with a high probability. On the other hand, structural communities can be detected by using uniform edge sampling in [89] to produce communities that are densely connected inside and sparsely connected outside. A very enlightening example was shown in Fig. 4 of [88]. We note that the computation complexity of constructing the desired bivariate distribution to detect a certain type of communities might vary. For instance, the computation complexity of finding $p(v, w)$ by using uniform edge sampling in [89] is $O(1)$. On the other hand, the computation complexity of finding $p(v, w)$ by using a random walk requires solving the steady state probabilities of a Markov chain, which is in general $O(n^3)$.

5 Experimental datasets and results

5.1 The stochastic block model and the hierarchical agglomerative algorithm

The stochastic block model (SBM), as a generalization of the Erdös-Rényi random graph [90], is a commonly used method for generating random graphs that can be used for benchmarking community detection algorithms [91], [92]. In a stochastic block model with q blocks (communities), the total number of nodes in the random graph are evenly distributed to these q blocks. The probability that there is an edge between two nodes within the same block is p_{in} and the probability that there is an edge between two nodes in two different blocks is p_{out}. These edges are generated independently. Let $c_{in} = n \cdot p_{in}$, $c_{out} = n \cdot p_{out}$. Then it is known (see e.g., [93], [94], [91]) that these q communities can be detected (in theory for a large network) if

$$|c_{in} - c_{out}| > q\sqrt{\text{mean degree}}. \tag{98}$$

As an illustrating example of the stochastic block model with two blocks, we use the hierarchical agglomerative algorithm to compare the sampling methods by PageRank in Example 25 and random walks with self-loops and backward jumps in Example 26. In these experiments, the number of nodes n in the stochastic block model is 2000 with 1000 nodes in each of these two blocks. The average degree (the sum of the average in-degree and the average out-degree) of a node is set to be 3. The values of $c_{in} - c_{out}$ of these graphs are in the range from 2.5 to 5.9 with a common step of 0.1. We generate 20 graphs for each $c_{in} - c_{out}$. Isolated vertices are removed. Thus, the exact numbers of vertices used in this experiment might be slightly less than 2000. For PageRank, the parameter λ is chosen to be 0.9. For the random walk with self-loops and backward jumps, the three parameters are $\lambda_0 = 0.05$, $\lambda_1 = 0.75$ and $\lambda_2 = 0.2$. We run the greedy hierarchical agglomerative algorithm in Algorithm 1 until there are only two sets (even when there do not exist nonnegative correlation measures between two distinct sets). We then evaluate the overlap with the true labeling. In Figure 10, we show the experimental results, where each point is averaged over 20 random graphs from the stochastic block model. The error bars are the 95% confidence intervals. From Figure 10, one can see that the performance of random walks with self-loops and backward jumps is better than that of PageRank. One reason for this is that PageRank uniformly adds an edge (with a small weight) between any two nodes and these added edges change the network topology. On the other hand, mapping by a random walk with backward jumps in (87) does not change the network topology when it is viewed as an undirected network.

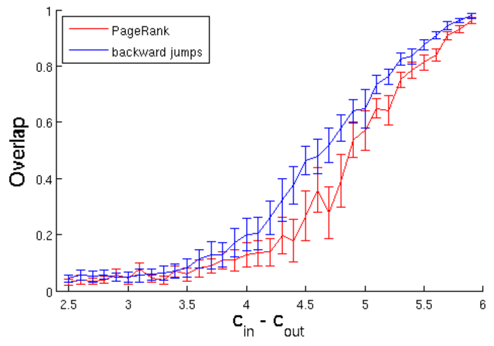


Fig. 10. Community detection of the stochastic block model by using PageRank in (86) and a random walk with self-loops and backward jumps in (87).

5.2 The LFR model and the fast unfolding algorithm

Both the average degrees of nodes and the community sizes in the stochastic block model are constants. To address this problem, the LFR benchmark graphs [95] have the freedom to choose the distributions of node degrees and community sizes. There are several parameters that need to be specified in the LFR benchmark graphs. The node degrees and the (ground truth) community sizes in the LFR model are distributed according to the power law, with exponents γ and β, respectively. The symbol $\langle k \rangle$ denotes the average degree. The mixing parameter μ is used for characterizing how the built-in communities in a graph are mixed. Specifically, each node has a fraction $1-\mu$ (resp. μ) of its edges connected to the other nodes of its community (resp. the other nodes of the network). To generate directed network, the direction of the edge is chosen with a certain probability. In the experiments below, such a probability is chosen to be 0.5.

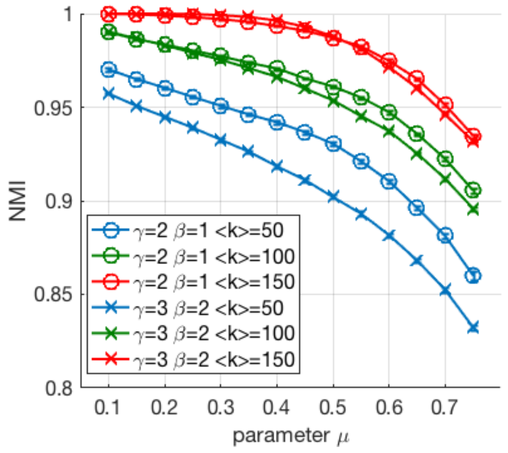


Fig. 11. The NMIs of our fast unfolding algorithm as a function of the mixing parameter μ.

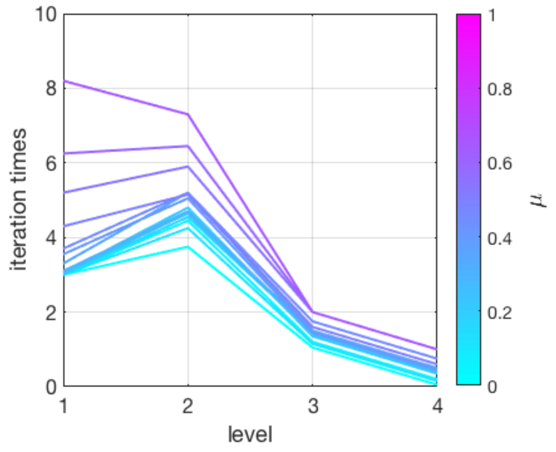


Fig. 12. The number of iterations in each level of recursive calls. The parameters γ, β, $\langle k \rangle$ are set to be 2, 1, 100, respectively.

Now we illustrate the use of the LFR model by the fast unfolding algorithm. Though the fast unfolding algorithm is a nearly-linear time algorithm for large sparse graphs with edge sampling, it suffers from the problem of resolution limit [62] due to its limited visibility to "see" more than one step. To address the problem of resolution limit, *stability* was then proposed in [21], [22] based on continuous-time random walks. As shown in [59], the stability is also a special case of the modularity for a certain sampled graph. However, the sampled graph that corresponds to the stability in [21], [22] is no longer sparse and thus the computational cost of the fast unfolding algorithm is increased dramatically. One reasonable compromise might be to consider the following sampling method with paths of length two:

$$p(u,v) = \frac{\breve{a}_{u,v}}{2\breve{m}}, \tag{99}$$

where

$$\breve{A} = (\breve{a}_{u,v}) = (A + A^T) + 0.5(A + A^T)^2$$

and

$$\breve{m} = \sum_u \sum_v \breve{a}(u,v)/2.$$

The objective of the first experiment for the LFR model is to show that our fast unfolding algorithm can produce good community detection results for various parameter settings. In the first experiment, the two exponents (γ, β) are chosen to be (2,1) and (3,2), respectively. The rest of the parameters, including the number of nodes, the maximum degree, $\langle k \rangle$, are set to be 100,000, 300, and 100, respectively. The values of mixing parameter μ of these graphs are in the range from 0.1 to 0.75 with a common step of 0.05. We generate 20 graphs for each μ. Then, we run our fast unfolding algorithm on each graph and compute the normalized mutual information measure (NMI) by using a built-in function in igraph [96]. The results are shown in Figure 11. The error bars are the 95% confidence intervals. As shown in Figure 11, our fast unfolding algorithm can produce good community detection results when the mixing parameter μ is small. Also, increasing the average degree $\langle k \rangle$ also increases the NMIs of the community detection results. In Figure 12, we show the number of the iterations of performing the partition algorithm at various recursive levels. As shown in Figure 12, the number of iterations increases when the mixing parameter μ increases.

In the second experiment for the LFR model, we show that our fast unfolding algorithm is capable of dealing with large scale networks. In this experiment, γ, β, the number of nodes, the maximum degree, the average degree, and the mixing parameter μ are set to be 2, 1, 10,000,000, 500,100, 0.5, respectively. As such, there are roughly half a billion edges in a graph with 10 million nodes. The number of ground-truth communities is 57187. As it takes more than 150 hours to generate such a large graph, we only generate a single LFR benchmark graph in this experiment. We use a random walk with self-loops to obtain the needed bivariate distribution for our probabilistic framework. In each node, the random walker will either stay at the same node or go along with one of its edges to an adjacent node, with probabilities λ and $1 - \lambda$, respectively. Hence, the transition probability matrix of such a Markov chain can be characterized by $\lambda I + (1 - \lambda)D^{-1}A$. The steady state probability for the random walker to visit node v is $\pi_v = k_v/2m$, where k_v is the degree of node v and m is the total number of edges in the network. In this experiment, we set $\lambda = 2999/3000$ so that the random walker has a very high probability to stay at the same node. We then compare our algorithm with the original fast unfolding algorithm that uses Newman's modularity [68]. The original fast unfolding algorithm is known to be one of the fastest community detection algorithms in the literature. Our algorithm is implemented in C++, while the original fast unfolding algorithm is obtained from built-in functions of igraph [96], which are implemented in C. We perform the experiment on an Acer Altos-T350-F2 machine with two Intel(R) Xeon(R) CPU E5-2690 v2 processors (only one single thread is used). We compare the performance of these two algorithms in Table 4. Note that the I/O time is excluded from the running time in Table 4. As shown in Table 4, our algorithm achieves better performance than the original fast unfolding algorithm, both in accuracy (in terms of NMI) and the running time. To dig further, we observe that the original fast unfolding algorithm (that aims to maximize Newman's modularity) fails to detect small communities because it keeps recursively merging communities until there are only 3395 communities left. This phenomenon is known as the resolution limit previously reported in [62]. To see the intuition behind this, note that Newman's modularity in the original fast unfolding algorithm is equivalent to using the bivariate distribution $p_{vw} = A_{vw}/(2m)$ in our fast unfolding algorithm and this corresponds to the random walker without self-loops. Intuitively, increasing the self-loop probability of the random walker decreases its mobility and the mobility of the random walker can be used to control the size of detected communities. Decreasing (resp. Increasing) the mobility behaves like "zooming in" (resp. "zooming out") on the graph, and results in the detection of "local communities" (resp. "global communities"), which are small (resp. large) in size. Given this, our probabilistic framework has the flexibility to choose the resolution and might be more suitable for practical applications than other algorithms that have fixed resolutions.

5.3 The six cluster model and the process of eliminating communities with weak contributions

We generate a graph with six communities on a plane. We consider six unit circle centered at $(2, 0)$, $(4, 0)$, $(6, 0)$, $(2, 2)$, $(4, 2)$, and $(6, 2)$, respectively. For the first (resp. last) three circles, we generate 1000 (resp. 800) random nodes. A node within a circle is generated by first selecting its distance to its center uniformly in $[0, 1]$ and then its angle uniformly in $[0, 2\pi]$. Clearly, there are six "ground truth clusters" of these 5400 nodes as shown in Figure 13(a). Now we construct a directed (neighborhood) graph by connecting two nodes with a directed edge if their distance is not greater than a constant ϵ. The direction of the edge is chosen with an equal probability of 0.5.

In our first experiment, we choose $\epsilon = 0.5$. We use the random walk with backward jumps for generating the sampled graph. The three parameters are $\lambda_0 = 0$, $\lambda_1 = 0.5$ and $\lambda_2 = 0.5$. In view of (87), such a sampled graph has the following *symmetric* bivariate distribution

$$p(u, v) = \frac{a_{u,v} + a_{v,u}}{2m}, \tag{100}$$

where $A = (a_{u,v})$ is the adjacency matrix of the directed graph and m is the total number of edges. This is also equivalent to the edge sampling [59] of an undirected graph with the adjacency matrix $B = A + A^T$ (with A^T being the transpose of the matrix A) that chooses the two end nodes of a uniformly selected edge. The modularity for edge sampling is thus the same as the Newman's modularity [25]. The modularity of the partition with the six "ground truth clusters" for such a graph is 0.8105. We run the fast unfolding algorithm in Algorithm 3 for such a graph and it outputs a partition of 6 communities with the modularity 0.8106 (see Figure 13(b)). The fast unfolding algorithm in facts ends when the partitional algorithm in Algorithm 2 ends after 6 iterations. In other words, node aggregation is not used for this experiment of the fast unfolding algorithm.

In our second experiment, we construct another graph that is more difficult to detect communities. This is done by choosing $\epsilon = 0.25$ so that the graph is much more sparse than the one with $\epsilon = 0.5$. Once again, we use the sampling in (100) for this new graph. The modularity of the partition with the six "ground truth clusters" for such a graph is 0.8197. We run the fast unfolding algorithm in Algorithm 3 for such a graph and it outputs a partition of 11 communities with the modularity 0.8246 (see Figure 13(c)), which is much larger than the modularity of the ground truth partition 0.8197. As can be seen from Figure 13(a) and (c), the fast unfolding algorithm outputs a partition (with a much larger modularity value) that is not even close to the ground truth partition. In view of this, one should carefully examine the output partition of a modularity maximization algorithm.

For this new graph with $\epsilon = 0.25$, the fast unfolding algorithm needs 5 levels of node aggregations and the partitional algorithm in the first level has 20 iterations, which is significantly larger than the graph with $\epsilon = 0.5$. By examining the 11 communities (see Table 3), we find that there are 5 communities (communities 3,4,6,7 and 9) that have relatively weak contributions to the modularity. This does not mean the community strengths of these 5 communities are small. In fact, their community strengths are comparable to the other 6 communities (communities 1,2,5,8,10 and 11). However, their sizes are relatively smaller. After the process of eliminating communities with weak contributions to the modularity in Section 3.6, we have a partition of six clusters with the modularity 0.8175 (see Figure 13(d)). We then use that partition as the initial partition for another run of the fast unfolding algorithm and it outputs a partition of six clusters with the modularity 0.8200 (see Figure 13(e)). One lesson learned from this experiment is that the fast unfolding algorithm (based on modularity maximization) might produce small communities with relatively strong community strengths if the total number of communities is not known in advance. These small communities are also negatively correlated to the large communities and thus cannot be merged into larger communities. In view of this, post processing that carefully examines the results produced by modularity maximization algorithms (such as the fast unfolding algorithm) is crucial to improving the quality of the community detection result.

In Figure 13(f), we show the partition generated by the fast unfolding algorithm for the sampled graph in (99). Since the matrix A is the adjacency matrix of the graph with $\epsilon = 0.25$, the matrix $\breve{A}$ is capable of "seeing" nodes within the distance 0.5. As shown in Figure 13(a) and (f), its performance is comparable to that by the fast unfolding algorithm for the more dense graph with $\epsilon = 0.5$.

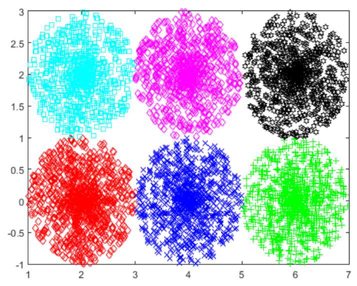
TABLE 3
The 11 communities by the fast unfolding algorithm for a graph with six clusters in Figure 13(c).

community	1	2	3	4	5	6	7	8	9	10	11
contribution	0.1441	0.1397	0.0222	0.0229	0.1345	0.0225	0.0096	0.1116	0.0127	0.1044	0.1004
strength	0.8029	0.8007	0.7984	0.7908	0.7888	0.7961	0.8046	0.8662	0.8641	0.8631	0.8800
size	717	621	352	324	476	372	148	756	230	598	806

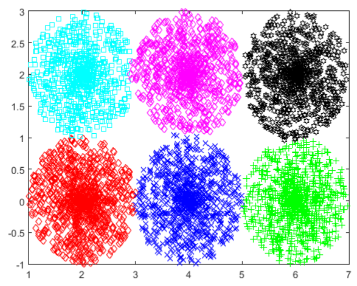
5.4 The SNAP collection

In addition to synthetic networks, there are several well-known real-world networks from the Stanford Network Analysis Project Collection (SNAP) [71]. The collection contains six networks: Amazon, DBLP, Friendster, LiveJournal, Orkut, and Youtube. All the networks come with top 5000 ground-truth communities with the highest quality. Since all the ground-truth communities are *overlapping*, we need a preprocessing step to generate the ground-truth *disjoint* communities in these networks. To do so, one can follow the *maximum independent set (MIS)* method in [97] and use their code to generate the ground-truth *disjoint* communities. For each graph G, the MIS method considers each of the 5000 ground-truth community as a giant vertex, and assigns an edge between two giant vertices if and only if two communities share at least one vertex, i.e., they are two overlapping communities. Specifically, let C_i, $i = 1, 2, \ldots, 5000$, be the 5000 ground-truth communities in G. The MIS method constructs another graph $G'(V', E')$ with $V' = \{C_1, C_2 ... C_{5000}\}$ and $E' = \{(C_i, C_j) | C_i \cap C_j \neq \phi\}$. Then, it finds the maximum independent set of the graph G' and removes all the giant vertices that are not in the maximum independent set. This then leads to a graph with disjoint communities. We summarize the parameters of these six graphs by using the MIS method in Table 5, including the number of vertices, the number of edges and the number of ground-truth communities.

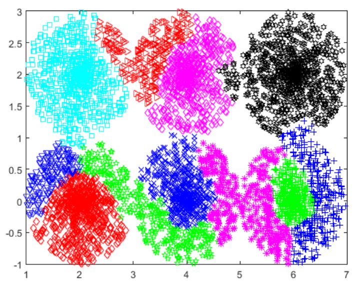
Now we use a random walk with self-loops to obtain the bivariate distribution for our probabilistic framework and the fast unfolding algorithm in Algorithm 3 for community detection. In each node, the random walker will either stay at the same node or go along with one of its edges to an adjacent node, with probabilities λ and $1 - \lambda$, respectively. In this experiment, the



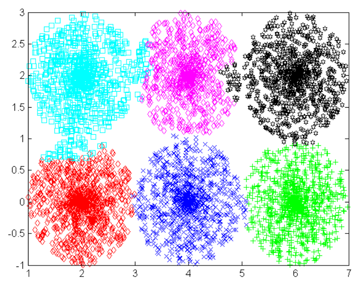
(a) ground truth



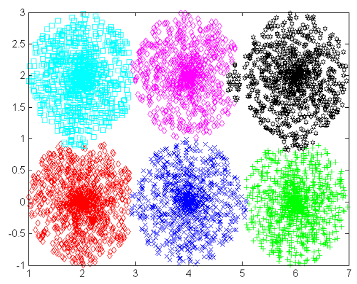
(b) fast unfolding, $\epsilon = 0.5$



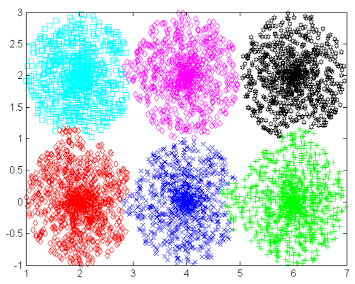
(c) fast unfolding, $\epsilon = 0.25$



(d) eliminating communities, $\epsilon = 0.25$



(e) 2nd fast unfolding, $\epsilon = 0.25$



(f) another sampled graph, $\epsilon = 0.25$

Fig. 13. The community detection result for a graph with six ”ground truth clusters.”

values of $1-\lambda$ are selected from $2^{-\ell}$, $\ell = 0, 1, \ldots, 18$. The NMI results are shown in Figure 14, and the running time for each data point in Figure 14 is within two seconds. Note that for $\ell = 0$, we have $\lambda = 0$ and the random walk with self-loops degenerates to the original fast unfolding algorithm. As such, the first data point of each curve corresponds to that from the original fast unfolding algorithm. As shown in this figure, the NMI values of each curve form an inverted U-shaped curve that gradually increases to its maximum and then decreases. When the value of $1-\lambda$ is lower than a particular threshold, the curve will suddenly level off. To explain this phenomenon, we manually examine the output communities. We find that the number of the communities increases when the value of $1-\lambda$ decreases. This is because the random walker has a higher probability to stay at the same node for a smaller $1-\lambda$. If the value of $1-\lambda$ is below a particular threshold, the algorithm outputs each single node as a community itself. As such, each curve levels off when $1-\lambda$ is below that threshold. Moreover, when each curve reaches its maximum NMI, the output of the algorithm has roughly the same number of communities as that of the ground-truth communities. For the two networks with less than 1000 communities, i.e., Amazon and Orkut, the original fast unfolding performs well since both curves reach their maximums with relatively small values of λ. On the other hand, for the other four networks with more than 1000 communities, i.e., DBLP, Friendster, LiveJournal, and Youtube, the number of communities resulted from the original fast unfolding is smaller than the number of ground-truth communities. For these four networks, the fast unfolding algorithm based on a random walk with self-loops performs much better than the original fast unfolding algorithm. In view of this, when the number of detected communities is smaller than expected, one might consider using another “viewpoint” to detect the ground-truth communities in a real-world network (such as the random walk with self-loops in this experiment).

TABLE 4
Performance comparison between the original fast unfolding algorithm and the fast unfolding algorithm in our probabilistic framework.

performance metric	NMI	running time (hours)
fast unfolding (Newman's modularity)	0.86	1.45
fast unfolding (probabilistic framework)	1	0.50

TABLE 5
Summary of the six networks after the preprocessing step.

	Amazon	DBLP	Friendster	Livejournal	Orkut	Youtube
number of vertices	8258	23479	79803	38713	6722	10206
number of edges	22129	72116	992461	633195	73522	15959
number of communities	992	3021	3529	2520	375	2883

6 Centrality analysis in attributed networks

In this section, we consider attributed networks. An attributed network is a generalization of a network (graph). In addition to the set of nodes and the set of edges in a (underlining) network, an attributed network could have node attributes and edge attributes that specify the "features" of nodes and edges. For instance, a signed network is an attributed network, where each edge is labelled with either a positive sign or a negative sign to indicate the friendship/enemy relationship between the two ends of an edge. Another typical example of an attributed network is that every node in the network represents a person with different ratings/interests on various topics. The problem we would like to address is how we rank nodes in attributed networks. Such a problem will be called the centrality analysis in attributed networks.

Our approach to the centrality analysis in attributed networks is to use the probabilistic framework for structural analysis in networks in [58], [59], [70]. Like PageRank [5], the probabilistic framework requires a sampling method of a network, called a *viewpoint*. The sampling method of a network is characterized by a probability measure for randomly selecting a path r in the network. In order to take the attributes of nodes and edges into account, one needs a *biased* viewpoint as previously discussed in Personalized PageRank [73], [74]. As such, the sampling method in an attributed network (and the corresponding probability measure) needs to be a *twisted* probability measure of its underlining network. For this, we propose using a path measure $f(r)$ that maps every path r in an attributed network to a real-valued vector. By specifying the average values of a path measure, we then have a set of constraints for the twisted sampling probability measure. This then leads to the exponentially twisted probability measure [75], [76], [77] that minimizes the Kullback-Leibler distance between the twisted probability measure and the original probability measure under the set of constraints from the average values of the path measure.

Each path measure with specified average values leads to a method of ranking nodes in an attributed network and that method is in general different from the original ranking method in its underlining network. In this section, we introduce three path measures in attributed networks and that leads to three new notions of centralities in attributed networks. For signed networks that have both positive edges and negative edges, we show how the influence centralities can be defined by using a path measure derived from opinions dynamics and how the trust centralities can be defined by using a path measure derived from a chain of trust. In particular, we show that one may vary the specified average value of the path measure in a signed network so that the influence centrality is turned into positive degree ranking, negative degree ranking and total degree ranking. For attributed networks with node attributes, we also show how

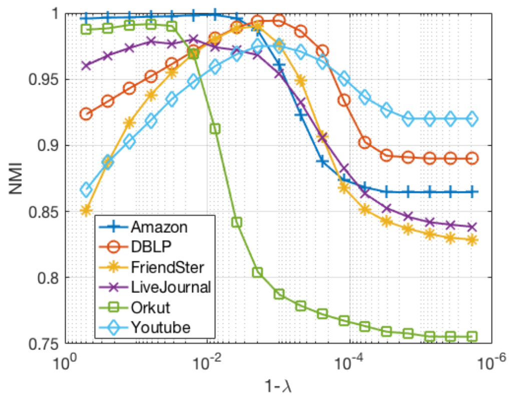


Fig. 14. The NMIs of the six networks from the SNAP collection. The values of $1-\lambda$ are selected from $2^{-\ell}$, $\ell = 0, 1, \ldots, 18$.

advertisement-specific influence centralities can be defined by using a specific path measure that models influence cascades in such networks.

6.1 Exponentially twisted sampling in attributed networks

Now we generalize the probabilistic framework to attributed networks. An attributed network is a generalization of a graph $G(V, E)$ by assigning each node $u \in V$ an attribute $h_V(u)$ and each edge $e \in E$ an attribute $h_E(e)$. As such, an attributed network can be represented as $G(V, E, h_V(\cdot), h_E(\cdot))$, where $h_V(\cdot)$ and $h_E(\cdot)$ are called the node attribute function and the edge attribute function, respectively.

For a path r that traverses a sequence of nodes $\{u_1, u_2, \ldots, u_{k-1}, u_k\}$ in a attributed network, we can define a path measure $f(r)$ as a function of the attributes of the nodes and edges along the path r, i.e.,

$$\{h_V(u_1), \ldots, h_V(u_k), h_E(u_1, u_2), \ldots, h_E(u_{k-1}, u_k)\}.$$

In this section, we assume that a path measure $f(\cdot)$ is a mapping from the set of paths R to an L-dimensional real-valued vector in $\mathcal{R}^L$, i.e.,

$$f(r) = (f_1(r), f_2(r), \ldots, f_L(r)),$$

where $f_i(\cdot)$, $i = 1, 2, \ldots, L$, are real-valued functions.

Now suppose that we use the probability mass function $p_0(r)$ to sample a path r in $G(V, E)$. The question is how the sampling distribution should be changed so that the average path measure is equal to a specified vector $\bar{f}$. In other words, what is the most likely sampling distribution $p(\cdot)$ that leads to the average path measure $\bar{f}$ given that the original sampling distribution is $p_0(\cdot)$? For this, we introduce the Kullback-Leibler distance between two probability mass functions $p(\cdot)$ and $p_0(\cdot)$:

$$D(p\|p_0) = \sum_{r\in R} p(r)\log(\frac{p(r)}{p_0(r)}). \tag{101}$$

The Kullback-Leibler distance is known to be nonnegative and it is zero if and only if $p(\cdot) = p_0(\cdot)$ (see e.g., [76]). Also, according to the Sanov theorem (see e.g., the books [76], [77]), the larger the Kullback-Leibler distance is, the more unlikely for $p_0(\cdot)$ to behave like $p(\cdot)$. Thus, to address the question, we consider the following constrained minimization problem:

$$\begin{aligned} \min \quad & D(p \| p_0) \\ s.t. \quad & \sum_{r \in R} p(r) = 1, \\ & \sum_{r \in R} f(r) p(r) = \overline{f}. \end{aligned} \tag{102}$$

The first constraint states that the total probability must be equal to 1. The second constraint states that the average path measure must be equal to $\bar{f}$ with the new sampling distribution $p(\cdot)$.

The above minimization problem can be solved by using Lagrange's multipliers $\alpha \in \mathcal{R}$ and $\theta \in \mathcal{R}^L$ as follows:

$$\begin{aligned} I = D(p \| p_0) + \alpha(1 - \sum_{r \in R} p(r)) \\ +\theta \cdot (\overline{f} - \sum_{r \in R} f(r) p(r)). \end{aligned} \tag{103}$$

Taking the partial derivative with respect to $p(r)$ yields

$$\frac{\partial I}{\partial p(r)} = \log p(r) + 1 - \log p_0(r) - \alpha - \theta \cdot f(r) = 0. \tag{104}$$

Thus,

$$p(r) = \exp(\alpha - 1) * \exp(\theta \cdot f(r)) * p_0(r). \tag{105}$$

Since $\sum_{r \in R} p(r) = 1$, it then follows that

$$p(r) = C * \exp(\theta \cdot f(r)) * p_0(r), \tag{106}$$

where

$$C = \frac{1}{\sum_{r \in R} \exp(\theta \cdot f(r)) * p_0(r)} \tag{107}$$

is the normalization constant.

To solve the parameter vector θ, we let

$$F = \log(1/C).$$

The quantity F is called the free energy as it is analogous to the free energy in statistical mechanic [4]. Also, the parameter vector θ is called the inverse temperature vector. It is easy to see that for $i = 1, 2, \ldots, L$ that

$$\frac{\partial F}{\partial \theta_i} = \sum_{r \in R} f_i(r) p(r) = \overline{f}_i. \tag{108}$$

These L equations can then be used to solve θ_i, $i = 1, 2, \ldots, L$.

Once we have the sampling distribution in (106), we can define a bivariate distribution $p_{U,W}(u, w)$ as in (4). Specifically, let $R_{u,w}$ be the set of paths from u to w and U (resp. W) be the starting (resp. ending) node of a randomly select path according to the sampling distribution in (106). Then the bivariate distribution

$$p_{U,W}(u, w) = \mathsf{P}(U = u, W = w) = \sum_{r \in R_{u,w}} p(r) \tag{109}$$

is the probability that the ordered pair of two nodes (U, W) is selected and it can be viewed as a similarity measure from node u to node w. Analogous to the discussion of a sampled graph in the previous section, the marginal distribution of the random variable U (resp. W), i.e., $p_U(u)$ (resp. $p_W(w)$), can be viewed as the out-centrality of u (resp. in-centrality of w).

To summarize, in order to define the out-centrality and the in-centrality of an attributed network, one needs (i) the original sampling distribution $p_0(\cdot)$ for the network, and (ii) the path measure $f(\cdot)$ of the attributed network. Once a specified average path measure $\bar{f}$ (or the inverse temperature vector θ) is given, one can have the new sampling distribution $p(\cdot)$ in (106). This then leads to the bivariate distribution in (109). The marginal distributions of that bivariate distribution then correspond to the out-centrality and the in-centrality of the attributed network.

6.2 Influence centralities in signed networks

In this section, we consider a special class of attributed networks, called *signed networks*. A signed network $G = (V, E, h_E(\cdot))$ is an attributed network with an edge attribute function $h_E(\cdot)$ that maps every undirected edge in E to the two signs $\{+, -\}$. We represent the positive (resp. negative) sign by 1 (resp. -1). An edge (u, w) mapped with the $+$ sign is called a *positive* edge, and it is generally used for indicating the *friendship* between the two nodes u and w. On the other hand, an edge mapped with the $-$ sign is called a *negative* edge. A negative edge (u, w) indicates that u and w are *enemies*.

One interesting question for signed networks is how the nodes in signed networks are ranked. Such a question was previously addressed in [98] by using a signed random surfer model that consists of a positive surfer and a negative surfer. Instead of using the signed random surfer model, our idea for this question is to use *opinion dynamics*. If u and w are connected by a positive (resp. negative) edge, then it is very likely that u will have a positive (resp. negative) influence on w and vice versa. As such, if we start from a node u with a positive opinion on a certain topic, then a neighbor of node u connected by a positive (resp. negative) edge will tend to have the same (resp. the opposite) opinion as node u has. Now we can let the opinion propagate through the entire network (via a certain opinion dynamic) and count the (expected) number of nodes that have the same opinion as node u has. If such a number is large, then it seems reasonable to say that node u has a large positive influence on the other nodes in the network. In other words, a node u has a large positive influence if there is a high probability that the other end of a randomly selected path has the same opinion as node u. This then leads us to define the notion of *influence centralities* for ranking nodes in signed networks.

The above argument is based on the general belief that "a friend of my friend is likely to be my friend" and "an enemy of my enemy can be my friend" in [4]. As such, for a path r that traverses a sequence of nodes $\{u_1, u_2, \ldots, u_k\}$ in a signed network, we define the following path measure as the product of the edge signs along the path, i.e.,

$$f(r) = \prod_{(u_i, u_{i+1}) \in r} h_E(u_i, u_{i+1}). \tag{110}$$

Note that $f(r)$ is either 1 or -1 as the edge attribute function $h_E(\cdot)$ that maps every undirected edge in E to $\{1, -1\}$.

As an illustrating example, let us consider sampling the undirected network $G = (V, E)$ by a random walk with path length 1 or 2. Such sampling methods are previously addressed in Example 4 and Example 13.

Example 34. **(Sampling by a random walk with path length 1 or 2)** For an undirected graph $G(V, E)$ with n nodes, let $m = |E|$ be the total number of edges, k_v be the degree of node v,

$v = 1, 2, \ldots, n$, and $A = (a_{i,j})$ be the adjacency matrix. A path r with length 1 can be represented by the two nodes $\{u_1, u_2\}$ it traverses. Similarly, a path with length 2 can be represented by the three nodes $\{u_1, u_2, u_3\}$ it traverses. A random walk with path length not greater than 2 can be generated by the following two steps: (i) with the probability $k_v/2m$, an initial node v is chosen, (ii) with probability β_i, $i = 1, 2$, a walk with length i is chosen. As such, we have

$$p_0(r) = \begin{cases} \frac{\beta_1}{2m} a_{u_1,u_2}, & \text{if } r = \{u_1, u_2\}, \\ \frac{\beta_2}{2m} \frac{a_{u_1,u_2} a_{u_2,u_3}}{k_{u_2}}, & \text{if } r = \{u_1, u_2, u_3\}, \end{cases} \tag{111}$$

where $\beta_1 + \beta_2 = 1$ and $\beta_i \geq 0$, $i = 1, 2$. For an undirected network, we have $a_{i,j} = a_{j,i}$ for all $i, j = 1, 2, \ldots, n$. Thus, in view of (111), we also have

$$p_0(r) = p_0(Rev(r)), \tag{112}$$

where $Rev(r)$ is the reverse path of r. Moreover, if $\beta_2 = 0$, then the random walk has path length 1, and this is equivalent to sampling by uniformly selecting an edge.

With the sampling distribution $p_0(r)$ from a random walk with path length 1 or 2 in Example 34, we have follows from (106), (110) and (111) that

$$p(r) = \begin{cases} C \cdot e^{\theta h_E(u_1,u_2)} \cdot \frac{\beta_1}{2m} a_{u_1,u_2}, \\ \quad \text{if } r = \{u_1, u_2\}, \\ C \cdot e^{\theta \cdot h_E(u_1,u_2) \cdot h_E(u_2,u_3)} \cdot \frac{\beta_2}{2m} \frac{a_{u_1,u_2} a_{u_2,u_3}}{k_{u_2}}, \\ \quad \text{if } r = \{u_1, u_2, u_3\}. \end{cases} \tag{113}$$

The constant C in (113) is the normalization constant. Summing all the paths from u to w yields the bivariate distribution

$$p_{U,W}(u, w) = C\Big[e^{\theta h_E(u,w)} \cdot \frac{\beta_1}{2m} a_{u,w} + \sum_{u_2 \in V} e^{\theta \cdot h_E(u,u_2) \cdot h_E(u_2,w)} \cdot \frac{\beta_2}{2m} \frac{a_{u,u_2} a_{u_2,w}}{k_{u_2}}\Big]. \tag{114}$$

The marginal distribution of the bivariate distribution, denoted by $P_U(u)$, is called the *influence centrality* of node u (with respect to the inverse temperature θ).

If we only select paths with length 1, i.e., $\beta_2 = 0$ in (114), then there is a closed-form expression for the influence centrality. For this, we first compute the normalization constant C by summing over u and w in (114) and this yields

$$C = \frac{m}{m^+ e^{\theta} + m^- e^{-\theta}}, \tag{115}$$

where m^+ (resp. m^-) is the total number of positive (resp. negative) edges in the graph. Thus, for $\beta_2 = 0$,

$$\begin{aligned} p_U(u) &= \sum_{w \in V} p_{U,W}(u, w) \\ &= \frac{(k_u^+ e^{\theta} + k_u^- e^{-\theta})}{2(m^+ e^{\theta} + m^- e^{-\theta})}, \end{aligned} \tag{116}$$

where k_u^+ (resp. k_u^-) is the number of positive (resp. negative) edges of node u.

Now suppose we require the average path measure $\bar{f}$ to be equal to some fixed constant $-1 < \gamma < 1$. Then we have from (108) that

$$\begin{aligned} \gamma &= \bar{f} = \frac{\partial F}{\partial \theta} \\ &= \frac{m^+ \exp(\theta) - m^- \exp(-\theta)}{m^+ \exp(\theta) + m^- \exp(-\theta)}, \end{aligned} \tag{117}$$

where $F = \log(1/C)$ with C in (115) being the free energy. This then leads to

$$\theta = \ln\left(\sqrt[2]{\frac{m^-(1+\gamma)}{m^+(1-\gamma)}}\right). \tag{118}$$

Now we discuss the connection of the influence centralities with the three degree ranking methods: (i) ranking by the number of positive edges (positive degree), (ii) ranking by the number of negative edges (negative degree), and (iii) ranking by the total number of edges (total degree). When $\gamma \to 1$, we have from (118) that $\theta \to \infty$. As a result from (116), $P_U(u) \to \frac{k_u^+}{2m^+}$ and this corresponds to positive degree ranking. On the other hand, when $\gamma \to -1$, we have $\theta \to -\infty$ and $P_U(u) \to \frac{k_u^-}{2m^-}$. This corresponds to negative degree ranking. Finally, if we choose $\gamma = (m^+ - m^-)/(m^+ + m^-)$, then $\theta = 0$ and $P_U(u) = \frac{k_u}{2m}$. This corresponds to total degree ranking. Thus, different choices of γ lead to different ranking methods. We will illustrate this further in Section 6.5.

6.3 Trust centralities in signed networks

As discussed in the previous section, the influence centralities are based on the general belief that "an enemy of my enemy can be my friend." Such a statement might be valid for modelling opinion dynamics. However, it is not suitable for modelling *trust*. In addition to the interpretation of a signed edge as the friend/enemy relationship, another commonly used interpretation is the trusted/untrusted link. A path r that traverses a sequence of nodes $\{u_1, u_2, \dots, u_k\}$ can be *trusted* if every edge is a trusted link so that there exists a chain of trust. In view of this, the notion of trust centrality in a signed network can be defined by using the path measure f that is the minimum of the edge signs along the path, i.e.,

$$f(r) = \min_{(u_i, u_{i+1}) \in r} h_E(u_i, u_{i+1}). \tag{119}$$

If we use the random walk with path length 1 or 2 to sample a path r in $G(V, E)$ as in (111), then the sampling distribution for the signed network with the path measure in (119) can be written as follows:

$$p(r) = \begin{cases} C \cdot e^{\theta h_E(u_1,u_2)} \cdot \frac{\beta_1}{2m} a_{u_1,u_2}, \\ \quad \text{if } r = \{u_1, u_2\}, \\ \\ C \cdot e^{\theta \cdot \min[h_E(u_1,u_2), h_E(u_2,u_))]} \cdot \frac{\beta_2}{2m} \frac{a_{u_1,u_2} a_{u_2,u_3}}{k_{u_2}}, \\ \quad \text{if } r = \{u_1, u_2, u_3\}. \end{cases} \tag{120}$$

Moreover, we have the following bivariate distribution

$$\begin{aligned} p_{U,W}(u, w) = C\Big[& e^{\theta h_E(u,w)} \cdot \frac{\beta_1}{2m} a_{u,w} \\ + \sum_{u_2 \in V} & e^{\theta \cdot \min[h_E(u,u_2), h_E(u_2,w)]} \cdot \frac{\beta_2}{2m} \frac{a_{u,u_2} a_{u_2,w}}{k_{u_2}} \Big], \end{aligned} \tag{121}$$

where C is the normalization constant. The marginal distribution of the bivariate distribution, denoted by $P_U(u)$, is the *trust centrality* of node u (with respect to the temperature θ). We note that if we only select paths with length 1, i.e., $\beta_2 = 0$, then the trust centrality is the same as the influence centrality in (116).

6.4 Advertisement-specific influence centralities in networks with node attributes

In this section, we consider another class of attributed networks that have node attributes. For a graph $G(V, E)$ with the node attribute function $h_V(u)$ that maps every node u to a vector in $\mathcal{R}^L$

$$(h_{V,1}(u), h_{V,2}(u), \ldots, h_{V,L}(u)). \tag{122}$$

One intuitive way to interpret such an attributed network is to view the graph $G(V, E)$ as a social network with n users and the attribute vector in (122) as the scores of user u on various topics. Now suppose an advertisement z can be represented by a vector of scores $(z_1, z_2, \ldots, z_L)$ with z_i being the score of the i^{th} topic. Then we would like to find out who is the most influential user in the network to pass on the advertisement z. Such a problem was previously studied in [99] for ranking nodes in Twitter. In TwitterRank [99], a two-step approach was used. First, a topic-specific ranking is obtained for each topic by using a random surfer model similar to that in PageRank [5]. The second step is then to take the weighted average over these topic-specific rankings. Specifically, suppose that $RT_i(u)$ is the ranking for topic i and user u. TwitterRank for advertisement z and user u is then defined as the following weighted average:

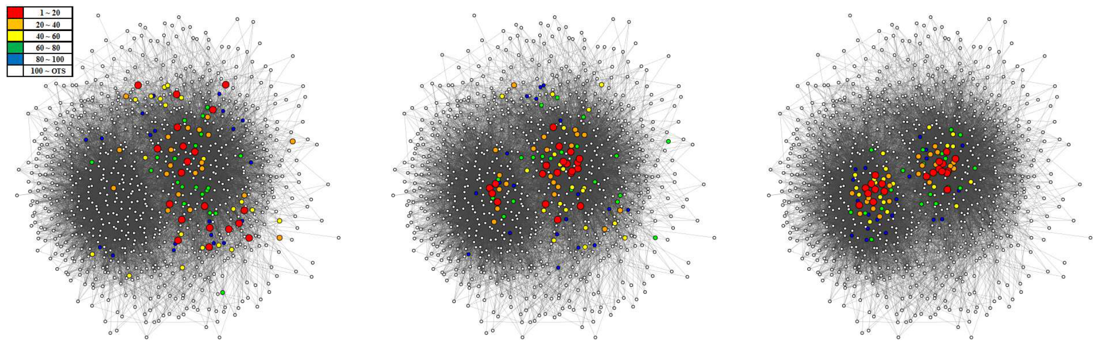
$$\sum_{i=1}^{L} z_i \cdot RT_i(u). \tag{123}$$

One flaw for such a two-step approach is that it neglects the fact that the propagation of a specific advertisement through a user depends on how much a user "likes" the advertisement. To model how much a user "likes" an advertisement, we use the similarity measure from the inner product of the score vector of the user and that of the advertisement. It is possible that in a cascade of two users $\{u_1, u_2\}$, both users like the advertisement because their inner products are large, but user u_1 likes one topic in that advertisement and user u_2 likes another different topic in that advertisement. Such a cascade cannot be modelled by using the two-step approach in TwitterRank [99]. In view of this, it might be better to use a one-step approach for computing advertisement-specific influence centralities. As the influence centralities in the previous section, we propose using opinion dynamics through a path. For a path r that traverses a sequence of nodes $\{u_1, u_2, \ldots, u_{k-1}, u_k\}$ in the attributed network, we define the following path measure

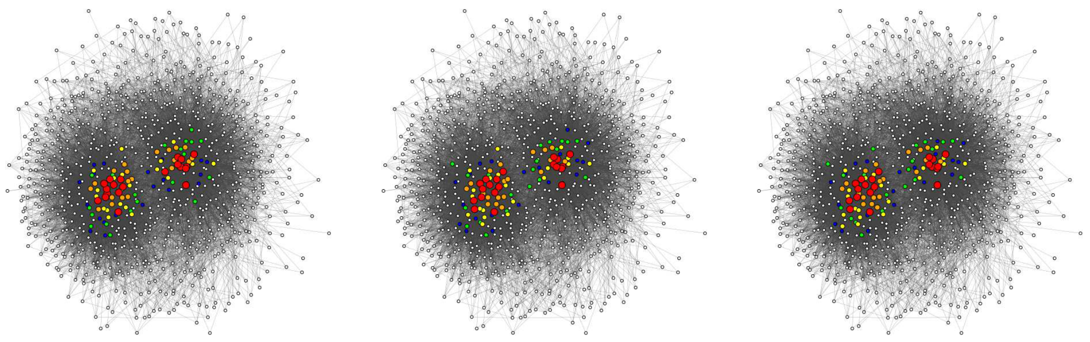
$$f(r) = \min_{u \in r}[\sum_{i=1}^{L} z_i \cdot h_{V,i}(u)]. \tag{124}$$

6.5 Experimental results for the influence centralities in signed networks

In this section, we evaluate the influence centralities in signed networks by using the real dataset from the political blogs in [100]. The network in [100] is a directed network of hyperlinks between weblogs collected around the time of the United States presidential election of 2004. There are 1,490 nodes and 19,090 edges in this dataset. These 1490 nodes can be partitioned into two (ground-truth) communities (parties). In order to have a signed network for our experiment, we add 4,000 negative edges between two nodes chosen randomly from each community. We then symmetrize the adjacency matrix by using the matrix $A + A^T$ (to obtain an undirected network). We also delete the nodes with degree smaller than 7, and remove self edges and multiple edges.



(a) γ = -0.9 (θ=-2.6566) (b) γ =-0.5 (θ=-1.7337) (c) γ =0 (θ=-1.1844)



(d) γ =0.5 (θ=-0.6351) (e) γ =0.9 (θ=0.2878) (f) γ =0.99 (θ=1.4623)

Fig. 15. The ranking result of influence centrality by β_2=0 with different theta

As a result, we obtain a simple undirected signed network with 863 nodes and 16,650 edges, including 15,225 positive edges and 1,425 negative edges.

For our experiment, we use sampling by a random walk with path length 1, i.e., $\beta_2 = 0$ in Example 34. In Figure 15, we show the ranking results of the influence centralities for six values of γ. The corresponding θ is computed from (118). The top 100 nodes are marked with different colors, 1-20 in red, 21-40 in orange, 41-60 in yellow, 61-80 in green and 81-100 in blue. When γ is chosen to be -0.9 in Figure 15(a), the top 100 nodes seem to be uniformly distributed among all the nodes. It is interesting to see that the top 100 nodes gradually move toward the "center" of each community when the value of γ is increased. Also, these top 100 nodes are separated into the two communities, and they are closely packed with each other around the center of each community. In our plots, the nodes near the center of each community have large degrees. As discussed in Section 6.2, the choice of γ is closely related to the three degree ranking methods:(i) ranking by the number of positive edges (positive degree), (ii) ranking by the number of negative edges (negative degree), and (iii) ranking by the total number of edges (total degree). To further illustrate the connection between the influence centrality and the three degree ranking methods, we compute the Jaccard index between the top 100 nodes obtained from the influence centrality and the top 100 nodes obtained from each of the degree centrality method. Recall that the Jaccard index between two sets S_1 and S_2 is computed as follows:

$$J(S_1, S_2) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}. \tag{125}$$

In Figure 16, we plot the three curves of the Jaccard indices with respect to γ. One can see that the Jaccard index for the curve from ranking by negative degree is a decreasing function of γ, while the other two curves are increasing functions of γ. This shows that the influence centrality with γ close to -1 is mostly in line with ranking by negative degree. On the other hand, the influence centrality with γ close to 1 is mostly in line with ranking by positive degree. This is because increasing γ increases the probability that a positive edge is sampled (and decreases the probability that a negative edge is sampled). Note that there is a slight difference between ranking by positive degree and ranking by total degree when γ is close to 1 as ranking by total degree still counts the number of negative edges and those negative edges have little chance being selected when γ is close to 1.

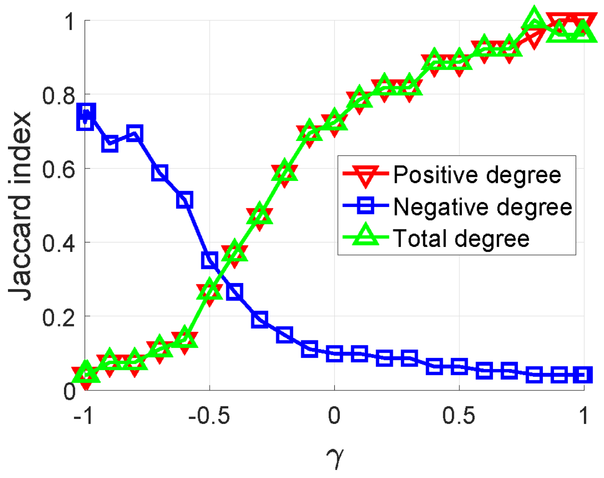
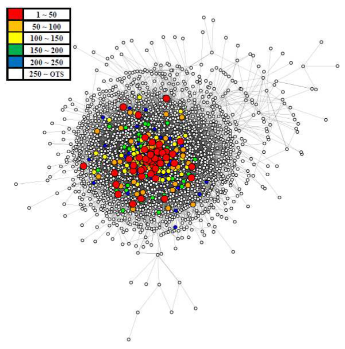


Fig. 16. The three Jaccard index curves between the influence centralities and the three degree centralities for the top 100 nodes.

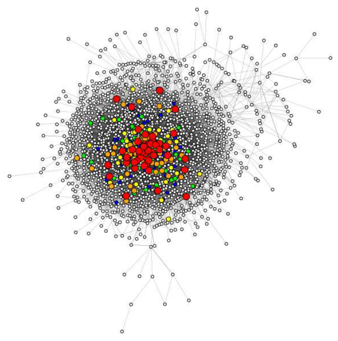
6.6 Experimental results for the advertisement-specific influence centralities

In this section, we evaluate the performance of the advertisement-specific influence centrality by using the MemeTracker dataset [101]. Such a dataset collects the quotes and phrases, called "memes," that frequently appear over time across mass media and personal blogs. To obtain the advertisement information from these memes, we use Carrot2 [102], an open source clustering engine, to classify memes into fifteen topics including "People", "Going", "Know", "Years", "Way", "United States", "States", "Life", "Believe", "Lot", "Love", "America", "Country", "Barack Obama" and "Obama". As "United States" and "Barack Obama" are clearly subsets of the two topics, "States" and "Obama", they are merged into these two topics. Therefore, we obtain a dataset with thirteen topics. As in the previous experiment, we delete the nodes with degree smaller than 7, and remove self edges and multiple edges. This then leads to an attributed network with 2082 nodes and 16,503 edges.

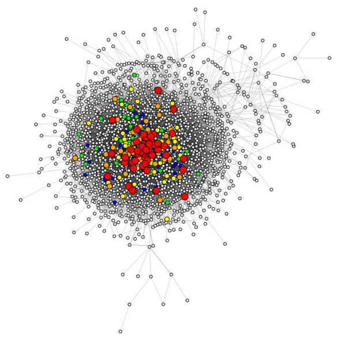
Again, we use sampling by a random walk with path length 1, i.e., $\beta_2 = 0$ in Example 34. The inverse temperature θ is set to be 0.2 (as the top 250 nodes do not change when θ is larger than 0.2 in our experiments). In Figure 17, we show the ranking results for the six most frequently-used phrases, i.e., "Going", "Know", "People", "Years", "America", and "Obama". As shown in Figure 17, different topics lead to different ranking results. Additional experimental results for the ranking method that combines various topics can be found in the full report [103].



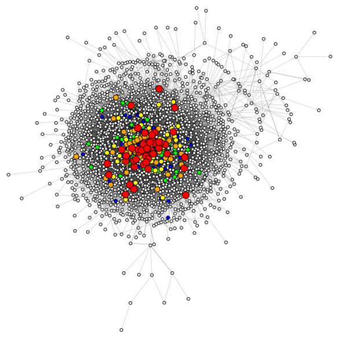
(a) Going



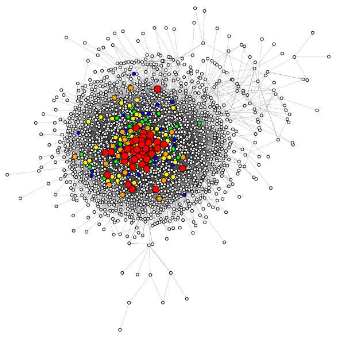
(b) Know



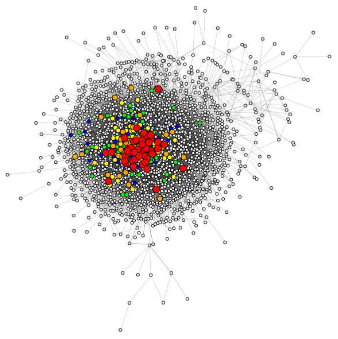
(c) People



(a) Years



(b) America



(c) Obama

Fig. 17. The ranking results of advertisement-specific influence centralities with different topics

7 Introduction to clustering

7.1 Literature review

Clustering is one of the most fundamental problems in data analysis and it has a lot of applications in various fields, including Internet search for information retrieval, social network analysis for community detection, and computation biology for clustering protein sequences. The problem of clustering has been studied extensively in the literature (see e.g., the books [38], [39] and the historical review papers [40], [41]). For a clustering problem, there is a set of data points (or objects) and a similarity (or dissimilarity) measure that measures how similar two data points are. The aim of a clustering algorithm is to cluster these data points so that data points within the same cluster are similar to each other and data points in different clusters are dissimilar.

As stated in [41], clustering algorithms (like community detection algorithms) can be divided into two groups: *hierarchical* and *partitional*. Hierarchical algorithms can further be divided into two subgroups: *agglomerative* and *divisive*. Agglomerative hierarchical algorithms, starting from each data point as a sole cluster, recursively merge two *similar* clusters into a new cluster. On the other hand, divisive hierarchical algorithms, starting from the whole set as a single cluster, recursively divide a cluster into two *dissimilar* clusters. As such, there is a hierarchical structure of clusters from either a hierarchical agglomerative clustering algorithm or a hierarchical divisive clustering algorithm.

Partitional algorithms do not have a hierarchical structure of clusters. Instead, they find all the clusters as a partition of the data points. The K-means algorithm is perhaps the simplest and the most widely used partitional algorithm for data points in a Euclidean space, where the Euclidean distance serves as the natural dissimilarity measure. The K-means algorithm starts from an initial partition of the data points into K clusters. It then repeatedly carries out the Lloyd iteration [104] that consists of the following two steps: (i) generate a new partition by assigning each data point to the closest cluster center, and (ii) compute the new cluster centers. The Lloyd iteration is known to reduce the sum of squared distance of each data point to its cluster center in each iteration and thus the K-means algorithm converges to a local minimum. The new cluster centers can be easily found if the data points are in a Euclidean space (or an inner product space). However, it is in general much more difficult to find the representative points for clusters, called medoids, if data points are in a non-Euclidean space. The refined K-means algorithms are commonly referred as the K-medoids algorithm (see e.g., [42], [38], [43], [44]). As the K-means algorithm (or the K-medoids algorithm) converges to a local optimum, it is quite sensitive to the initial choice of the partition. There are some recent works that provide various methods for selecting the initial partition that might lead to performance guarantees [45], [46], [47], [48], [49]. Instead of using the Lloyd iteration to minimize the sum of squared distance of each data point to its cluster center, one can also formulate a clustering problem as an optimization problem with respect to a certain objective function and then solve the optimization problem by other methods. This then leads to kernel and spectral clustering methods (see e.g., [50], [51], [52], [53], [54] and [55], [56] for reviews of the papers in this area). Solving the optimization problems formulated from the clustering problems are in general NP-hard and one has to resort to approximation algorithms [57]. In [57], Balcan et al. introduced the concept of approximation stability that assumes all the partitions (clusterings) that have the objective values close to the optimum ones are close to the target partition. Under such an assumption, they proposed efficient algorithms for clustering large datasets.

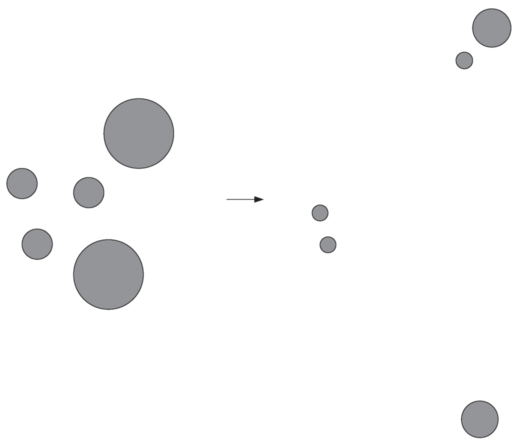


Fig. 18. A consistent change of 5 clusters.

Though there are already many clustering algorithms proposed in the literature, clustering theories that justify the use of these clustering algorithms are still unsatisfactory. As pointed out in [105], there are three commonly used approaches for developing a clustering theory: (i) an axiomatic approach that outlines a list of axioms for a clustering function (see e.g., [106], [107], [108], [109], [110], [111]), (ii) an objective-based approach that provides a specific objective for a clustering function to optimize (see e.g., [112], [57], and (iii) a definition-based approach that specifies the definition of clusters (see e.g, [113], [114], [115]). In [109], Kleinberg adopted an axiomatic approach and showed an impossibility theorem for finding a clustering function that satisfies the following three axioms:

(i) Scale invariance: if we scale the dissimilarity measure by a constant factor, then the clustering function still outputs the same partition of clusters.

(ii) Richness: for any specific partition of the data points, there exists a dissimilarity measure such that the clustering function outputs that partition.

(iii) Consistency: for a partition from the clustering function with respect to a specific dissimilarity measure, if we increase the dissimilarity measure between two points in different clusters and decrease the dissimilarity measure between two points in the same cluster, then the clustering function still outputs the same partition of clusters. Such a change of a dissimilarity measure is called a *consistent* change.

The impossibility theorem is based on the fundamental result that the output of any clustering function satisfying the scale invariance property and the consistency property is in a collection of *antichain* partitions, i.e., there is no partition in that collection that in turn is a *refinement* of another partition in that collection. As such, the richness property cannot be satisfied. In [112], it was argued that the impossibility theorem is *not* an inherent feature of clustering. The key point in [112] is that the consistency property may not be a desirable property for a clustering function. This can be illustrated by considering a consistent change of 5 clusters in Figure 18. The figure is redrawn from Figure 1 in [112] that originally consists of 6 clusters. On the left hand side of Figure 18, it seems reasonable to have a partition of 5 clusters. However, after the consistent change, a new partition of 3 clusters might be a better output than the original partition of 5 clusters. As such, they abandoned the three axioms for *clustering functions* and proposed another three similar axioms for Clustering-Quality Measures (CQM) (for measuring the quality of a partition). They showed the existence of a CQM that satisfies their three axioms for CQMs.

As for the definition-based approach, most of the definitions of a single cluster in the literature are based on loosely defined terms [38]. One exception is [113], where Ester et al. provided a precise definition of a single cluster based on the concept of density-based reachability. A point p is said to be directly density-reachable from another point q if point p lies within the ϵ-neighborhood of point q and the ϵ-neighborhood of point q contains at least a minimum number of points. A point is said to be density-reachable from another point if they are connected by a sequence of directly density-reachable points. Based on the concept of density-reachability, a cluster is defined as a maximal set of points that are density-reachable from each other. An intuitive way to see such a definition for a cluster in a set of data points is to convert the data set into a graph. Specifically, if we put a directed edge from one point p to another point q if point p is directly density-reachable from point q, then a cluster simply corresponds to a *strongly connected component* in the graph. One of the problems for such a definition is that it requires specifying two parameters, ϵ and the minimum number of points in a ϵ-neighborhood. As pointed out in [113], it is not an easy task to determine these two parameters.

Mapping data points and the associated similarity (or dissimilarity) measure to a graph has several advantages, including (i) rich theories and algorithms: there are many well-known graph theories and algorithms that can be used, (ii) sparsification: the adjacency matrix of a graph can be made to be sparse by considering neighboring points within a limited distance, and (iii) manifold discovery: it is possible to map high dimensional data in a Euclidean space to a graph to discover the embedded manifold by using the geodesic distance in the graph. Clustering in a graph is commonly referred to as the *community detection* problem in the literature and there are many algorithms that have been proposed for the community detection problem (see e.g., the review papers in [11], [24] and the comparison studies of these algorithms in [36], [37], [17]). Most importantly, the probabilistic framework developed for structural network analysis can now be used for clustering.

In this following, we briefly review several commonly used clustering algorithms. Readers are referred to the book [39] for more detailed references.

```
ALGORITHM 4: The $K$-means Algorithm
    Input: A data set $\Omega = \{x_1, x_2, \dots, x_n\}$ in $\mathcal{R}^p$ and the number of sets $K$.
    Output: A partition of sets $\{S_1, S_2, \dots, S_K\}$.
    (0) Initially, choose arbitrarily $K$ centroids $a_1, a_2, \dots, a_K$.
    (1) For each data point $x$, compute $(x - a_k)^T(x - a_k)$ for each centroid $a_k$. Find the centroid
      to which the point $x$ is closest. Assign that point $x$ to the set of that centroid.
    (2) Adjust the centroid of every set by using (127).
    (3) Repeat from (1) until there is no further change.
```

7.2 K-means: clustering data points in a Euclidean space

The K-means algorithm (see Algorithm 4) is commonly used to solve the clustering problem for data points in a Euclidean space (or more generally an inner product space). In the K-means algorithm, initially K centroids are chosen to represent K clusters. Then there are two iterative steps in the K-means algorithm: (i) assigning each point to the closest centroid, and (ii) recompute the centroid of each cluster after each assignment. Specifically, suppose that $\Omega = \{x_1, x_2, \dots, x_n\}$ contains data points in $\mathcal{R}^p$, and that the number of sets K for the partition of Ω is given in advance. The K-means algorithm finds a partition $S_1, S_2, \dots, S_k$ that achieves a local optimum of the objective function

$$\sum_{k=1}^{K} \sum_{x \in S_k} (x - a_k)^T (x - a_k), \tag{126}$$

where

$$a_k = \frac{1}{|S_k|} \sum_{x \in S_k} x \tag{127}$$

is the centroid of the set of points in S_k (see e.g., [41]). The centroid a_k acts as a *representative* point for S_k. Let $c(x)$ be the index of the set to which x belongs. Then one can write the objective function in (126) as follows:

$$\sum_{k=1}^{K} \sum_{x \in S_k} (x - a_k)^T (x - a_k) = \sum_{x \in \Omega} (x - a_{c(x)})^T (x - a_{c(x)}). \tag{128}$$

Thus, the objective function is equivalent to minimize the sum of the square of the distance between each point and its representative point. It is well-known that the computational complexity of the K-means algorithm is $O(pKnI)$, where I is the number of iterations. One problem of the K-means algorithm is the choice of the initial centroid in Step (0). For the choice of the initial centroids, it is suggested that the initial centroids should not too close to each other (see e.g., [46], [47], [48], [49] for additional references on this).

To see how the K-means algorithm works. Let S'_k, $k = 1, 2, \dots, K$ be the K sets after the first step. Also, let $c'(x)$ be the index of the set to which x belongs after the first step. In view of the minimization step for points, we have

$$(x - a_{c(x)})^T (x - a_{c(x)}) \geq (x - a_{c'(x)})^T (x - a_{c'(x)})$$

and thus

$$\sum_{x \in \Omega} (x - a_{c(x)})^T (x - a_{c(x)}) \geq \sum_{x \in \Omega} (x - a_{c'(x)})^T (x - a_{c'(x)}) = \sum_{k=1}^{K} \sum_{x \in S'_k} (x - a_k)^T (x - a_k).$$

Let a'_k be the centroid of S'_k, $k = 1, 2, \ldots, K$. One key property of a centroid is that

$$\sum_{x \in S'_k} (x - a'_k)^T (x - a'_k) \le \sum_{x \in S'_k} (x - y)^T (x - y),$$

for any point y. Thus,

$$\sum_{k=1}^{K} \sum_{x \in S'_k} (x - a_k)^T (x - a_k) \ge \sum_{k=1}^{K} \sum_{x \in S'_k} (x - a'_k)^T (x - a'_k) = \sum_{x \in \Omega} (x - a_{c'(x)})^T (x - a_{c'(x)}).$$

After these two steps, we then have

$$\sum_{x \in \Omega} (x - a_{c(x)})^T (x - a_{c(x)}) \ge \sum_{x \in \Omega} (x - a_{c'(x)})^T (x - a_{c'(x)})$$

and the value of the objective function is non-increasing in each iteration.

Instead of viewing the K-means algorithm as a minimization problem for the objective function in (128), it can also viewed as a maximization problem for a “similarity” measure. Let

$$g(x, y) = x^T y, \tag{129}$$

and

$$g(S_1, S_2) = \sum_{x \in S_1} \sum_{y \in S_2} g(x, y), \tag{130}$$

for any two sets S_1 and S_2. In addition to minimizing the objective function in (128), the K-means algorithm is also known to maximize the following objective function:

$$\sum_{k=1}^{K} \frac{g(S_k, S_k)}{|S_k|}, \tag{131}$$

where $|S_k|$ is the number of elements in S_k. To see this, we note from (127) that

$$\begin{aligned}
&\sum_{x \in S_k} (x - a_k)^T (x - a_k) \\
&= \sum_{x \in S_k} x^T x - \frac{1}{|S_k|} \sum_{x \in S_k} \sum_{y \in S_k} x^T y \\
&= \sum_{x \in S_k} g(x, x) - \frac{g(S_k, S_k)}{|S_k|}.
\end{aligned}$$

Thus, maximizing $\sum_{k=1}^{K} \frac{g(S_k, S_k)}{|S_k|}$ is equivalent to minimizing $\sum_{k=1}^{K} \sum_{x \in S_k} (x - a_k)^T (x - a_k)$.

Note that the objective function in the maximization problem in (131) is normalized with respect to the sizes of the output sets. As such, the K-means algorithm tends to output “clusters” with even sizes. This could be a problem if the sizes of the clusters that one would like to detect are very different. To illustrate this, we show an example in Figure 19, where there are two clusters centered at $(0, 0)$ and $(6, 0)$, respectively. The cluster centred at $(0, 0)$ consists of 5000 points within the radius of 4, and the cluster centered at $(6, 0)$ consists of 500 points within the radius of 1. As shown in Figure 19, the K-means algorithm fails to detect the desired clusters. The centroids of the two clusters from the K-means algorithm are $(0.0178, -0.0302)$ and $(3.9781, -0.4706)$, respectively..

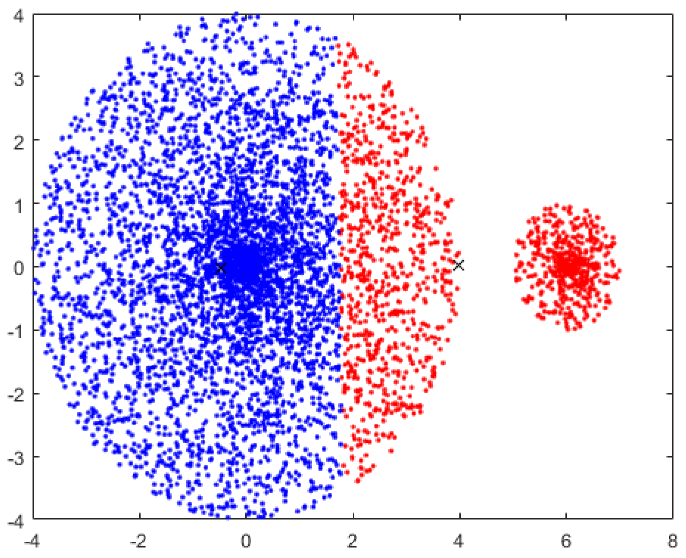


Fig. 19. An illustrating example of the K-means algorithm for the problem with uneven cluster sizes.

7.3 K-medoids: clustering data points in a non-Euclidean space

In a non-Euclidean space, there is no notion of centroid that can be used for representing a cluster of points. Instead, various notions of medoids (or clusteroids) were proposed to represent clusters, e.g., (i) the sum of the distance to the other points in the cluster, (ii) the maximum distance to another in the cluster, and (iii) the sum of the squares of the distances to the other points in the clusters (see e.g., the book [39]).

In particular, the K-medoids algorithm (see Algorithm 5) is an extension of the K-means algorithm to a non-Euclidean space with a distance measure $d(\cdot,\cdot)$ that uses the sum of the distance to the other points in the cluster to find the the medoid of a cluster. Note that there are two key properties in the minimization process of the K-means algorithm: (i) use the centroid as the representative point for a set and assign each point to the nearest centroid, and (ii) identify the new representative point, i.e., the new centroid, after the assignment. As there is no centroid for a set in a non-Euclidean space, one quick fix is to represent a set by the point in the set that minimizes the sum of the distance between a point in the set and the representative point. The representative point is then called a *medoid*. Specifically, consider a set S_k. Then the medoid of S_k, denoted by m_k, is

$$m_k = \arg\min_{y \in S_k} \sum_{x \in S_k} d(x,y). \tag{132}$$

Following the same argument as in the K-means algorithm, the K-medoids algorithm also reduces in each iteration the value of the objective function, i.e., $\sum_{x\in\Omega} d(x, m_{c(x)})$ with $c(x)$ denoting the index of the set to which x belongs to. As such,t he K-medoids algorithm converges to a local optimum.

7.4 Spectral clustering: clustering data points with a similarity measure

In this section, we consider a set of n data points, $\Omega = \{x_1, x_2, \ldots, x_n\}$. We assume there is a *symmetric* similarity measure $g(x,y)$ (i.e., $g(x,y) = g(y,x)$) for any two points x and y in Ω and

ALGORITHM 5: The K-medoids Algorithm

```
Input: A data set $\Omega = \{x_1, x_2, \ldots, x_n\}$ with a distance measure $d(\cdot,\cdot)$ and the number of sets
    $K$.
Output: A partition of sets $\{S_1, S_2, \ldots, S_K\}$.
(0) Initially, choose arbitrarily $K$ medoids $m_1, m_2, \ldots, m_K \in \Omega$.
(1) For each data point $x$, find the medoid to which the point $x$ is closest. Assign that point
    $x$ to the set of that medoid.
(2) Find the new medoid of every set by using (132).
(3) Repeat from (1) until there is no further change.
```

each data point x is associated with a *positive* weight $w(x)$ (i.e., $w(x) > 0$). For any two sets S_1 and S_2, let

$$g(S_1, S_2) = \sum_{x \in S_1} \sum_{y \in S_2} g(x, y). \tag{133}$$

Also, for any set S, let

$$w(S) = \sum_{x \in S} w(x). \tag{134}$$

A K-way partition of Ω consists of K *disjoint nonempty* sets $S_1, S_2, \ldots, S_K$ such that the union of these K sets is Ω (i.e., (i) $|S_i| \geq 1$ for all i, (ii) $S_i \cap S_j$ is an empty set for $i \neq j$, and (iii) $\cup_{k=1}^K S_k = \Omega$). Let $\mathcal{F}$ be the collection of all K-way partitions of Ω. Motivated by the maximization problem for the K-means algorithm in (131), we consider the following maximization problem:

$$\max \sum_{k=1}^{K} \frac{g(S_k, S_k)}{w(S_k)} \tag{135}$$

$$s.t. \quad \{S_1, S_2, \ldots, S_n\} \in \mathcal{F}. \tag{136}$$

A K-way partition that has a very high objective value of this optimization problem may be considered as a good solution of the clustering problem of Ω as data points in the same set are similar to each other.

It is well-known that the optimization problem in (135) is NP-complete and is related to the trace maximization problem [50], [116], [117]. An $n \times K$ matrix H is called a partition matrix of Ω if $H \in \{0,1\}^{n \times K}$, $\mathbf{1}_n^T H > 0$, and $H\mathbf{1}_K = \mathbf{1}_n$, where $\mathbf{1}_K$ and $\mathbf{1}_n$ are K dimensional and n dimensional column vectors with all its elements being 1. One can view a partition matrix H as a bi-adjacency matrix of a bipartite graph that maps the n data points to the K sets of a K-way partition $\{S_1, S_2, \ldots, S_K\}$, and thus there is a one-to-one mapping between a partition matrix and a K-way partition. Let H_k be the k^{th} column vector of H (that represents the set of data points in S_k). Also, let $G = (g(x,y))$ be the $n \times n$ similarity matrix and $D = diag(w(x_1), w(x_2), \ldots w(x_n))$. Then the optimization problem in (135) can be represented equivalently as follows:

$$\begin{aligned} \max \quad & \sum_{k=1}^{K} \frac{H_k^T G H_k}{H_k^T D H_k} \\ s.t. \quad & H \in \{0,1\}^{n \times K}, \\ & \mathbf{1}_n^T H > 0, \\ & H\mathbf{1}_K = \mathbf{1}_n. \end{aligned} \tag{137}$$

As shown in [116], one can use the scaled partition matrix

$$\tilde{H} = H(H^T D H)^{-\frac{1}{2}} \tag{138}$$

ALGORITHM 6: The Spectral Clustering Algorithm

```
Input: A data set $\Omega = \{x_1, x_2, \dots, x_n\}$, a *symmetric* similarity matrix $G = (g(\cdot,\cdot))$, a positive
        weight vector $\mathbf{w} = (w(x_1), w(x_2), \dots, w(x_n))$ and the number of sets $K$.
Output: A partition of sets $\{S_1, S_2, \dots, S_K\}$.
(0) Let $D = diag(w(x_1), w(x_2), \dots w(x_n))$.
(1) Let $\mathbf{Z}_1, \mathbf{Z}_2, \dots, \mathbf{Z}_K$ be the $K$ largest eigenvectors of the matrix $D^{-\frac{1}{2}} G D^{-\frac{1}{2}}$ (chosen to be
  orthogonal to each other). Form the $n \times K$ matrix $\mathbf{Z} = (z_{i,j}) = [\mathbf{Z}_1, \mathbf{Z}_2, \dots, \mathbf{Z}_K]$ by stacking
  these $K$ eigenvectors in columns.
(2) Form the $n \times K$ matrix $\tilde{\mathbf{Z}} = (\tilde{z}_{i,j})$ from $\mathbf{Z}$ by renormalizing each row in $\tilde{\mathbf{Z}}$ to have unit
  length, i.e.,
      $\tilde{z}_{i,j} = \frac{z_{i,j}}{(\sum_{\ell=1}^{K} z_{i,\ell}^2)^{\frac{1}{2}}}$.
(3) Treat each row of $\tilde{\mathbf{Z}}$ as a point in $\mathcal{R}^K$ and use the $K$-means algorithm to cluster these $n$
  points.
(4) Assign $x_i$ to the set $S_k$ if row $i$ of $\tilde{\mathbf{Z}}$ is assigned to the $k^{th}$ cluster by the $K$-means
  algorithm.
```

to represent the objective function in (137) by tr$(\tilde{H}^T G \tilde{H})$, where tr denotes the trace of a matrix. Since

$$\tilde{H}^T D \tilde{H} = (H^T D H)^{-\frac{1}{2}} H^T D H (H^T D H)^{-\frac{1}{2}} = \mathbf{I}_K,$$

one can relax the integer constraints in the optimization problem in (137) and consider the following maximization problem:

$$\begin{aligned} &\max \mathrm{tr}(\tilde{H}^T G \tilde{H}) \\ &s.t. \quad \tilde{H}^T D \tilde{H} = \mathbf{I}_K. \end{aligned} \tag{139}$$

To convert the trace maximization problem to the standard form, one can further let

$$\hat{H} = D^{\frac{1}{2}} \tilde{H}. \tag{140}$$

This then leads to

$$\begin{aligned} &\max \mathrm{tr}(\hat{H}^T D^{-\frac{1}{2}} G D^{-\frac{1}{2}} \hat{H}) \\ &s.t. \quad \hat{H}^T \hat{H} = \mathbf{I}_K. \end{aligned} \tag{141}$$

Since the matrix $D^{-\frac{1}{2}} G D^{-\frac{1}{2}}$ is a real symmetric matrix, it is diagonalizable. As stated in [55], a version of the Rayleigh-Ritz theorem shows that the solution of the trace maximization problem in (141) is given by choosing $\hat{H}$ as the matrix which contains the largest K eigenvectors of $D^{-\frac{1}{2}} G D^{-\frac{1}{2}}$ as columns. As the trace maximization problem in (141) is a relaxation of the optimization problem in (135), this also gives an upper bound on the objective value of the optimization problem in (135). Specifically, let $\lambda_1, \lambda_2, \dots, \lambda_K$ be the K largest eigenvalues of the matrix $D^{-\frac{1}{2}} G D^{-\frac{1}{2}}$. Then the objective value of the optimization problem in (135) is bounded above by $\sum_{k=1}^{K} \lambda_k$.

The rest of the problem is to map the solution for the trace optimization problem in (141) back to the solution space for the optimization problem in (137) (that requires the solutions to be binary). For this, we follow the approach in [118] that treats the largest K eigenvectors of $D^{-\frac{1}{2}} G D^{-\frac{1}{2}}$ as the features of the n points and applies the K-means algorithm to cluster these n points. The details of the spectral clustering algorithm is given in Algorithm 6.

To see the insight of the spectral clustering algorithm, let us consider the case with equal weights, i.e., $w(x_i) = 1$ for all i. Let ϕ be the mapping from Ω to $\mathcal{R}^K$ such that for $i = 1, 2, \ldots, n$,

$$\phi(x_i) = (\tilde{z}_{i,1}, \tilde{z}_{i,2}, \ldots, \tilde{z}_{i,K}). \tag{142}$$

Also, let

$$\tilde{g}(x, y) = \phi(x)^T \phi(y). \tag{143}$$

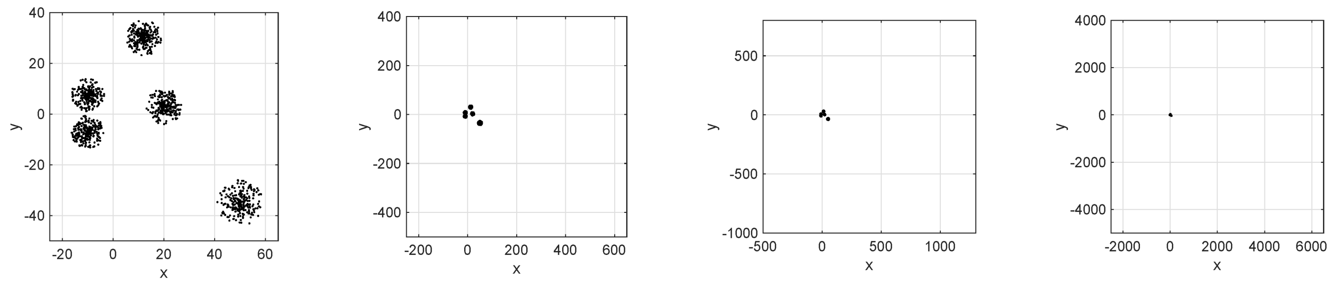
Note that the K-means algorithm in Step (3) of Algorithm 6 is to maximize

$$\sum_{k=1}^{K} \frac{\tilde{g}(S_k, S_k)}{|S_k|}. \tag{144}$$

In other words, the spectral clustering algorithm simply transforms the optimization problem in (135) to the optimization problem in (144) and hope that a good partition for the transformed optimization problem is also a good partition of the original optimization problem. Such a transformation would be exact if the matrix G is a positive definite (kernel) matrix and $K = n$. In such a scenario, the spectral clustering algorithm reduces to the kernel K-means in [32]. However, in practice K is much smaller than n, such a transform can be good if there is a significant spectral gap so that the similarity matrix G can be well approximated by the matrix factorization

$$G \approx \tilde{\mathbf{Z}}\tilde{\mathbf{Z}}^T, \tag{145}$$

where $\tilde{Z}$ is the $n \times K$ matrix in Step (2) of Algorithm 6. In view of (145), one might construct $\tilde{\mathbf{Z}}$ by using the largest L eigenvectors of G for some $L \neq K$. In particular, for $K = 2$, one might simply select the largest eigenvector of G and cluster the positive elements of that vector in one set and the rest in the other [66].



(a) Scale 1 (b) Scale 2 (c) Scale 3 (d) Scale 4

Fig. 20. A dataset plotted with different scales of the two axes.

8 Clustering data points in a metric/semi-metric space

Clustering is in general considered as an ill-posed problem. To see that there does not exist the so-called "ground-truth" clusters for this ill-posed problem, in Fig. 20 we plot a set of 1250 data points on a plane with different scales of the two axes. A quick glance at this figure might yield five (resp. four, three and one) clusters in Fig. 20 (a) (resp. (b), (c) and (d)). As such, the answer of the number of clusters in the same dataset depends on how one "views" this dataset.

To provide a viewpoint, one needs a sampling method for the dataset. For this, we consider a set of data points $\Omega = \{x_1, x_2, \ldots, x_n\}$ in a semi-metric space, in which there is a semi-metric that measures the distance between two points. Our idea of sampling the data points in a semi-metric space is to consider a complete graph with n nodes and n self edges and then maps each data point in Ω to a node in the graph with the edge weight between two nodes being the distance between the corresponding two points in Ω. By doing so, the problem of clustering in a semi-metric space is transformed to a community detection problem of a weighted graph. There are several well-known sampling techniques developed for community detections as described in the previous sections. In this section, we are particularly interested in the exponentially twisted sampling technique in which one can specify the desired average distance from the sampling distribution to detect clusters with various resolutions.

8.1 Exponentially twisted sampling

In this section, we use exponentially twisted sampling to sample the set of data points in a semi-metric space.

We consider a set of n data points, $\Omega = \{x_1, x_2, \ldots, x_n\}$ and a distance measure $d(x, y)$ for any two points x and y in Ω. The distance measure $d(\cdot, \cdot)$ is said to be a *semi-metric* if it satisfies the following three properties:

(D1) (Nonnegativity) $d(x, y) \geq 0$.
(D2) (Null condition) $d(x, x) = 0$.
(D3) (Symmetry) $d(x, y) = d(y, x)$.

If, furthermore, the triangular inequality in (D4) below is satisfied, then the distance measure is said to be a *metric*.

(D4) (Triangular inequality) $d(x, y) \leq d(x, z) + d(z, y)$.

For this set of data points in a semi-metric space, we can view it as a weighted complete graph with n self edges and the weight of an edge between two points is the corresponding distance of the two points. If we sample two points $X$and Y *uniformly*, then we have the following sampling bivariate distribution:

$$p(x, y) = \frac{1}{n^2}, \tag{146}$$

for all $x, y \in \Omega$. Using such a sampling distribution, the average distance between two randomly selected points is

$$\mathsf{E}_p[d(X,Y)] = \frac{1}{n^2}\sum_{x\in\Omega}\sum_{y\in\Omega} d(x,y). \tag{147}$$

Suppose we would like to change another sampling distribution $p_\lambda(x,y)$ so that the average distance between two randomly selected points, denoted by $\bar{d}$, is smaller than $\mathsf{E}_p[d(X,Y)]$ in (147). By doing so, a pair of two points with a shorter distance is selected more often than another pair of two points with a larger distance. For this, we consider the following minimization problem:

$$\begin{aligned} \min \quad & D(p_\lambda \| p) \\ s.t. \quad & \sum_{x\in\Omega}\sum_{y\in\Omega} p_\lambda(x,y) = 1, \\ & \sum_{x\in\Omega}\sum_{y\in\Omega} d(x,y)\cdot p_\lambda(x,y) = \bar{d}, \end{aligned} \tag{148}$$

where $D(p_\lambda \| p)$ is the Kullback-Leibler distance between the two probability mass functions $p_\lambda(x,y)$ and $p(x,y)$, i.e.,

$$D(p_\lambda \| p) = \sum_{x\in\Omega}\sum_{y\in\Omega} p_\lambda(x,y)\log\left(\frac{p_\lambda(x,y)}{p(x,y)}\right). \tag{149}$$

The solution of such a minimization problem is known to be the exponentially twisted distribution as follows (see [76]):

$$p_\lambda(x,y) = C * \exp(\lambda \cdot d(x,y)) * p(x,y), \tag{150}$$

where

$$C = \frac{1}{\sum_{x\in\Omega}\sum_{y\in\Omega}\exp(\lambda\cdot d(x,y)) * p(x,y)} \tag{151}$$

is the normalization constant. As the distance measure $d(\cdot,\cdot)$ is symmetric and $p(x,y) = 1/n^2$, we know that $p_\lambda(x,y) = p_\lambda(y,x)$. Moreover, the parameter λ can be solved by the following equation:

$$\frac{\partial F}{\partial \lambda} = \sum_{x\in\Omega}\sum_{y\in\Omega} d(x,y)\cdot p_\lambda(x,y) = \bar{d}, \tag{152}$$

where $F = \log(1/C)$ is the free energy.

If we choose $\lambda < 0$, then $\bar{d} \le \mathsf{E}_p[d(X,Y)]$. In Fig. 21, we plot the average distance $\bar{d}$ as a function of λ for the dataset in Fig. 20. Clearly, as $\lambda \to \infty$, $\bar{d}$ approaches to the maximum distance between a pair of two points. On the other hand, as $\lambda \to -\infty$, $\bar{d}$ approaches to the minimum distance between a pair of two points. The plot in Fig. 21 allows us to solve (152) numerically.

We note that the following transformation for data points in a Euclidean space to a graph was previously proposed in [118] for spectral clustering and in [51] for support vector clustering :

$$A_{x,y} = \exp(-\|x-y\|^2/2\sigma^2),$$

where $\|x-y\|$ is the Euclidean distance between the two points x and y, and σ is the scaling parameter. In view of this, $\|x-y\|^2$ is a semi-metric and our exponentially twisted sampling is a generalization of the transformation in [118], [51] from data points in a Euclidean space to data points in a semi-metric space.

Now with the exponentially twisted distribution, we can define sampled graphs as before.

***Definition* 35. (Sampled graph)** Let G be the complete graph with n nodes and n self edges. Each node in G corresponds to a specific data point in Ω and the edge weight between two

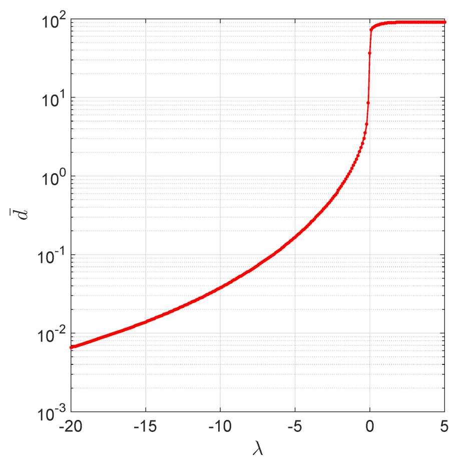


Fig. 21. The average distance $\bar{d}$ as a function of λ for the dataset in Fig. 20.

nodes is the distance measure between the corresponding two points in Ω. The graph G sampled by randomly selecting an ordered pair of two nodes (X, Y) according to the symmetric bivariate distribution $p_\lambda(x, y)$ in (150) is called a *sampled graph* and it is denoted by the two-tuple $(G, p_\lambda(\cdot, \cdot))$.

Let

$$p_\lambda(x) = \sum_{y \in \Omega} p_\lambda(x, y) \tag{153}$$

be the marginal probability that the point x is selected by using the sampling distribution $p_\lambda(\cdot, \cdot)$. The higher the probability is, the more important that point is. As such, $p_\lambda(x)$ can be used for ranking data points according to the sampling distribution $p_\lambda(\cdot, \cdot)$, and it is called the *centrality* of point x for the sampled graph $(G, p_\lambda(\cdot, \cdot))$. For example, if we choose very small $\lambda < 0$ and replace $\exp(\lambda d(x, y))$ by $1 + \lambda d(x, y)$ in (150), then it is easy to see that ranking by using $p_\lambda(x)$ is the same as ranking by using

$$-\sum_{y \in \Omega} d(x, y). \tag{154}$$

Such a ranking method is known as the *closeness centrality* in the literature (see e.g., the book [4]).

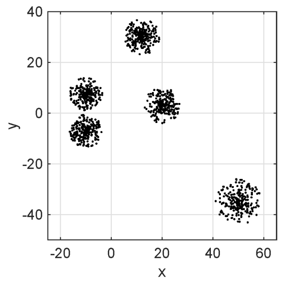
The exponentially twisted sampling technique embeds a set of data points in a semi-metric space into a *sampled graph* in Definition 35. As before, various notions for structural analysis of a sampled graph can be defined and analyzed, including centrality, community and modularity. Moreover, the problem of clustering in a semi-metric space is transformed to a community detection problem of a weighted graph and that can be solved by using modularity maximization algorithms in Section 3.

Definition 36. **(Correlation, Community and Modularity)** For a sampled graph $(G, p_\lambda(\cdot, \cdot))$, the correlation between two points x and y (cf. (58)) is defined as follows:

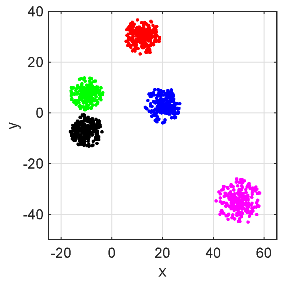
$$\gamma_\lambda(x, y) = p_\lambda(x, y) - p_\lambda(x) p_\lambda(y), \tag{155}$$

where $p_\lambda(x, y)$ is in (150). Using (146), (150), (151) and (153) in (155) yields

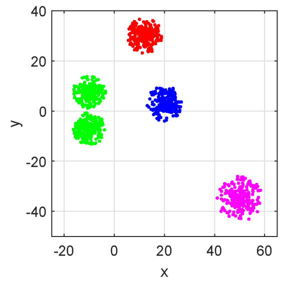
$$\begin{aligned} \gamma_\lambda(x, y) &= \frac{e^{\lambda d(x,y)}}{\sum_{z_1 \in \Omega} \sum_{z_2 \in \Omega} e^{\lambda d(z_1, z_2)}} - \frac{\sum_{z_1 \in \Omega} e^{\lambda d(x, z_1)}}{\sum_{z_1 \in \Omega} \sum_{z_2 \in \Omega} e^{\lambda d(z_1, z_2)}} \frac{\sum_{z_2 \in \Omega} e^{\lambda d(z_2, y)}}{\sum_{z_1 \in \Omega} \sum_{z_2 \in \Omega} e^{\lambda d(z_1, z_2)}} \\ &= \frac{\sum_{z_1 \in \Omega} \sum_{z_2 \in \Omega} \left(e^{\lambda (d(z_1, z_2) + d(x, y))} - e^{\lambda (d(x, z_1) + d(z_2, y))} \right)}{(\sum_{z_1 \in \Omega} \sum_{z_2 \in \Omega} e^{\lambda d(z_1, z_2)})^2} \end{aligned} \tag{156}$$



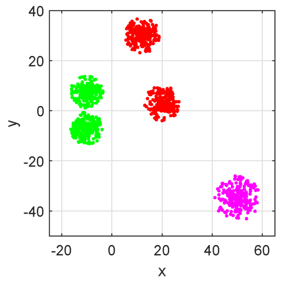
(a) The dataset in Fig. 20



(b) $\lambda = -0.5$



(c) $\lambda = -0.01$



(d) $\lambda = -0.0001$

Fig. 22. An illustrating example for various resolutions of $\bar{d}$ (data points in different clusters are marked with different colors).

Define the modularity matrix $\mathbf{Q} = (\gamma_\lambda(x, y))$ be the $n \times n$ matrix with its $(x, y)^{th}$ element being $\gamma_\lambda(x, y)$. Moreover, the correlation between two sets S_1 and S_2 (cf. (59)) is defined as follows:

$$\gamma_\lambda(S_1, S_2) = \sum_{x \in S_1} \sum_{y \in S_2} \gamma_\lambda(x, y). \tag{157}$$

Two sets S_1 and S_2 are said to be positively correlated if $\gamma_\lambda(S_1, S_2) \geq 0$. In particular, if a subset of points $S \subset \Omega$ is positively correlated to itself, i.e., $\gamma_\lambda(S, S) \geq 0$, then it is called a *community* or a *cluster*.

Let $\mathcal{P} = \{S_k, k = 1, 2, \ldots, K\}$, be a partition of Ω, i.e., $S_k \cap S_{k'}$ is an empty set for $k \neq k'$ and $\cup_{k=1}^K S_k = \Omega$. The modularity $Q(\mathcal{P})$ with respect to the partition S_k, $k = 1, 2, \ldots, K$, (cf. (61)) is defined as

$$Q(\mathcal{P}) = \sum_{k=1}^{K} \gamma_\lambda(S_k, S_k). \tag{158}$$

As an illustrating example, we use the partitional-hierarchical algorithm in Section 3 for the dataset in Fig. 20 (a). As shown in Fig. 21, one can select a particular λ for the sampling distribution $p_\lambda(\cdot, \cdot)$ so that the average distance is $\bar{d}$. In Table 6, we list the average distance $\bar{d}$ for various choices of λ. Now we feed the modularity matrix obtained from the sampling distribution $p_\lambda(\cdot, \cdot)$ into the partitional-hierarchical Algorithm. It is clear to see from Fig. 22 that various choices of λ lead to various resolutions of the partitional-hierarchical algorithm. Specifically, for $\lambda = -0.5$ (and $\bar{d} = 2.6055$), there are five clusters detected by the partitional-hierarchical algorithm. Data points in different clusters are marked with different colors. For $\lambda = -0.01$ (and $\bar{d} = 31.1958$), there are four clusters detected by the partitional-hierarchical algorithm. Finally, for $\lambda = -0.0001$ (and $\bar{d} = 36.6545$), there are only three clusters detected by the partitional-hierarchical algorithm. These clustering results obtained by using different sampling distributions are in line with those plotted by using different scales of the two axes in Fig. 20.

TABLE 6
The average distance $\bar{d}$ for various choices of λ.

λ	-0.5	-0.01	-0.0001	0	1
$\bar{d}$	2.6055	31.1958	36.6545	36.7121	87.8800

8.2 Cohesion measure in a metric space

Let us consider the correlation measure $\gamma_\lambda(x, y)$ in (156) when λ is very small. Using the first order approximation $e^{\lambda z} \approx 1 + \lambda z + o(\lambda)$ in (156) yeilds

$$\begin{aligned}\gamma_\lambda(x,y) &\approx \frac{\sum_{z_1\in\Omega}\sum_{z_2\in\Omega}\Big((1+\lambda(d(z_1,z_2)+d(x,y))+o(\lambda)) - (1+\lambda(d(x,z_1)+d(z_2,y))+o(\lambda))\Big)}{(\sum_{z_1\in\Omega}\sum_{z_2\in\Omega}(1+\lambda d(z_1,z_2)+o(\lambda)))^2} \\ &\approx (-\lambda)\Big(\frac{1}{n}\sum_{z_2\in\Omega} d(z_2,y) + \frac{1}{n}\sum_{z_1\in\Omega} d(x,z_1) - \frac{1}{n^2}\sum_{z_2\in\Omega}\sum_{z_1\in\Omega} d(z_2,z_1) - d(x,y)\Big) + o(\lambda). \qquad (159)\end{aligned}$$

Thus, when we choose a very small negative λ, the correlation measure $\gamma_\lambda(x, y)$ is proportional to

$$\begin{aligned}\gamma(x,y) &= \frac{1}{n}\sum_{z_2\in\Omega} d(z_2,y) + \frac{1}{n}\sum_{z_1\in\Omega} d(x,z_1) - \frac{1}{n^2}\sum_{z_2\in\Omega}\sum_{z_1\in\Omega} d(z_2,z_1) - d(x,y) \qquad (160) \\ &= \frac{1}{n^2}\sum_{z_2\in\Omega}\sum_{z_1\in\Omega}\Big(d(x,z_1)+d(z_2,y)-d(z_1,z_2)-d(x,y)\Big). \qquad (161)\end{aligned}$$

The function $\gamma(x, y)$ is called a cohesion measure (resp. semi-cohesion measure) if the distance measure $d(x, y)$ is a metric (resp. semi-metric) in [67], [78]. Moreover, two points x and y are said to be *cohesive* (resp. *incohesive*) if $\gamma(x, y) \geq 0$ (resp. $\gamma(x, y) < 0$). A cohesion measure satisfies the following properties in Proposition 37 and its proof is given in Section 9.9.

Proposition 37.

(i) (Symmetry) The cohesion measure is symmetric, i.e., $\gamma(x, y) = \gamma(y, x)$.
(ii) (Self-cohesiveness) Every data point is cohesive to itself, i.e., $\gamma(x, x) \geq 0$.
(iii) (Self-centredness) Every data point is more cohesive to itself than to another point, i.e., $\gamma(x, x) \geq \gamma(x, y)$ for all $y \in \Omega$.
(iv) (Zero-sum) The sum of the cohesion measures between a data point to all the points in Ω is zero, i.e., $\sum_{y\in\Omega} \gamma(x, y) = 0$.

These four properties can be understood intuitively by viewing a cohesion measure between two points as a "binding force" between those two points. The symmetric property ensures that the binding force is reciprocated. The self-cohesiveness property ensures that each point is self-binding. The self-centredness property further ensures that the self binding force is always stronger than the binding force to the other points. In view of the zero-sum property, we know for every point x there are points that are incohesive to x and each of these points has a negative binding force to x. Also, there are points that are cohesive to x (including x itself from the self-cohesiveness property) and each of these points has a positive force to x. As such, the binding force will naturally "push" points into "clusters."

To further understand the intuition of the cohesion measure, we can think of z_1 and z_2 in (161) as two random points that are used as reference points. Then two points x and y are cohesive if $d(x, z_1) + d(z_2, y) \geq d(z_1, z_2) + d(x, y)$ for two reference points z_1 and z_2 that are randomly chosen from Ω. In Figure 23, we show an illustrating example for such an intuition in $\mathcal{R}^2$. In Figure 23(a), point x is close to one reference point z_1 and point y is close to the other reference point z_2. As such, $d(x, z_1) + d(z_2, y) \leq d(z_1, z_2)$ and thus these two points x and y are incohesive. In Figure 23(b), point x is not that close to z_1 and point y is not that close to z_2. However, x and y are on the two opposite sides of the segment between the two reference points z_1 and z_2. As such, there are two triangles in this graph: the first triangle consists of the three points x, z_1, and w, and the second triangle consists of the three points y, z_2, and w. From the triangular inequality, we then have $d(w, z_1)+d(x, w) \geq d(x, z_1)$ and $d(y, w)+d(w, z_2) \geq d(y, z_2)$. Since $d(w, z_1)+d(w, z_2) = d(z_1, z_2)$

and $d(x, w) + d(y, w) = d(x, y)$, it then follows that $d(z_1, z_2) + d(x, y) \geq d(x, z_1) + d(y, z_2)$. Thus, points x and y are also incohesive in Figure 23(b). In Figure 23(c), point x is not that close to z_1 and point y is not that close to z_2 as in Figure 23(b). Now x and y are on the same side of the segment between the two reference points z_1 and z_2. There are two triangles in this graph: the first triangle consists of the three points x, y, and w, and the second triangle consists of the three points z_1, z_2, and w. From the triangular inequality, we then have $d(x, w) + d(w, y) \geq d(x, y)$ and $d(w, z_1) + d(w, z_2) \geq d(z_1, z_2)$. Since $d(x, w) + d(w, z_1) = d(x, z_1)$ and $d(w, y) + d(w, z_2) = d(z_2, y)$, it then follows that $d(x, z_1) + d(y, z_2) \geq d(z_1, z_2) + d(x, y)$. Thus, points x and y are cohesive in Figure 23(c). In view of Figure 23(c), it is intuitive to see that two points x and y are cohesive if *they both are far away from the two reference points and they both are close to each other*.

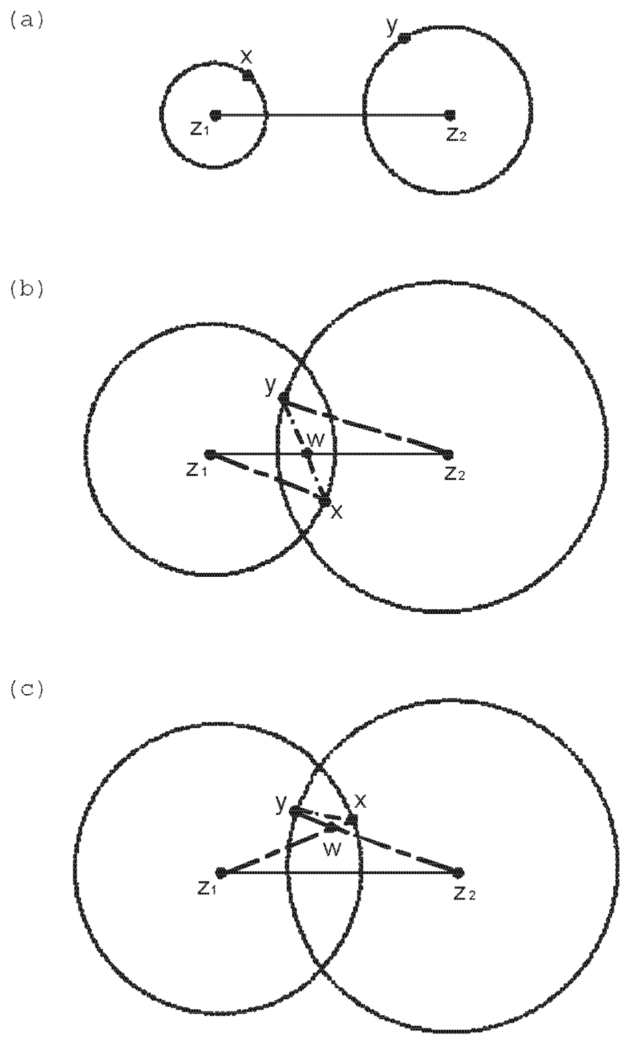


Fig. 23. Illustrating examples of the cohesion measure in $\mathcal{R}^2$: (a) incohesive as $d(x, z_1) + d(y, z_2) \leq d(z_1, z_2)$, (b) incohesive as $d(w, z_1) + d(x, w) \geq d(x, z_1)$ and $d(y, w) + d(w, z_2) \geq d(y, z_2)$, and (c) cohesive $d(x, w) + d(w, y) \geq d(x, y)$ and $d(w, z_1) + d(w, z_2) \geq d(z_1, z_2)$.

Analogous to Definition 36, we can extend the cohesion measure between two points to the cohesion measure between two sets and define clusters in metric spaces.

Definition 38. **(Cohesion measure between two sets and clusters)** Define the *cohesion measure* between two sets S_1 and S_2, denoted by $\gamma(S_1, S_2)$, as the sum of the cohesion measures of all the pairs of two points (with one point in S_1 and the other point in S_2), i.e.,

$$\gamma(S_1, S_2) = \sum_{x \in S_1} \sum_{y \in S_2} \gamma(x, y). \tag{162}$$

Two sets S_1 and S_2 are said to be *cohesive* (resp. *incohesive*) if $\gamma(S_1, S_2) \geq 0$ (resp. $\gamma(S_1, S_2) < 0$). A nonempty set S is called a *cluster* if it is cohesive to itself, i.e.,

$$\gamma(S, S) \geq 0. \tag{163}$$

One can also show a theorem for various equivalent statements for what a cluster is in a metric space.

Theorem 39. Consider a *nonempty* set S that is not equal to Ω. Let $S^c = \Omega \backslash S$ be the set of points that are not in S. Also, let $\bar{d}(S_1, S_2)$ be the average distance between two randomly selected points with one point in S_1 and another point in S_2, i.e.,

$$\bar{d}(S_1, S_2) = \frac{1}{|S_1| \times |S_2|} \sum_{x \in S_1} \sum_{y \in S_2} d(x, y). \tag{164}$$

The following statements are equivalent.

(i) The set S is a cluster, i.e., $\gamma(S, S) \geq 0$.
(ii) The set S^c is a cluster, i.e., $\gamma(S^c, S^c) \geq 0$.
(iii) The two sets S and S^c are incohesive, i.e., $\gamma(S, S^c) \leq 0$.
(iv) The set S is more cohesive to itself than to S^c, i.e., $\gamma(S, S) \geq \gamma(S, S^c)$.
(v) $2\bar{d}(S, \Omega) - \bar{d}(\Omega, \Omega) - \bar{d}(S, S) \geq 0$.
(vi) $2\bar{d}(S, S^c) - \bar{d}(S, S) - \bar{d}(S^c, S^c) \geq 0$.

One surprise finding in Theorem 39(ii) is that the set S^c is also a cluster. This shows that the points inside S are cohesive and the points outside S are also cohesive. Thus, there seems a boundary between S and S^c from the cohesion measure. Another surprise finding is in Theorem 39(vi). One usually would expect that a cluster S should satisfy $\bar{d}(S, S) \leq \bar{d}(S, S^c)$. But it seems our definition of a cluster is much weaker than that.

Analogous to the definition of modularity in (158) for the correlation measure, we can also define the modularity for the cohesion measure below.

Definition 40. **(Modularity and normalized modularity)** Let $\gamma(x, y)$ be a semi-cohesion measure of $\Omega = \{x_1, x_2, \ldots, x_n\}$and $\mathcal{P} = \{S_k, k = 1, 2, \ldots, K\}$ be a partition of Ω. The modularity $Q(\mathcal{P})$ with respect to the partition $\mathcal{P} = \{S_k, k = 1, 2, \ldots, K\}$ is defined as follows

$$Q(\mathcal{P}) = \sum_{k=1}^{K} \gamma(S_k, S_k). \tag{165}$$

Moreover, the normalized modularity $Q_{\text{nor}}(\mathcal{P})$ with respect to the partition $\mathcal{P} = \{S_k, k = 1, 2, \ldots, K\}$ is defined as follows:

$$Q_{\text{nor}}(\mathcal{P}) = \sum_{k=1}^{K} \frac{1}{|S_k|} \gamma(S_k, S_k). \tag{166}$$

The community detection algorithms in Section 3, including (i) the spectral modularity maximization algorithm, (ii) the hierarchical agglomerative algorithm, (iii) the partitional algorithm, and (iv) the fast unfolding algorithm [68], can be applied by maximizing modularity in (165). On the other hand, the spectral clustering algorithm in Section 7.4 can be applied by maximizing normalized modularity in (166) (if K is fixed and known in advance). The spectral clustering algorithm in Section 7.4 consists of two steps: (i) embedding data points into a Euclidean space by spectral decomposition and (ii) the K-means clustering algorithm. In the next section, we will introduce a single step clustering algorithm like the K-means algorithm and it does not require the step of embedding data points into a Euclidean space.

8.3 The K-sets algorithm: clustering data points in a metric space

In this section, we introduce a clustering algorithm, called the K-sets algorithm in [67], for clustering data points in a metric space. The K-sets algorithm is similar to the classical K-means algorithm by iteratively assigning each point to the nearest set (until it converges). Such an approach requires a measure that can measure how close a point x to a set S is. In the K-means algorithm, such a measure is defined as the square of the distance between x and the centroid of S. However, there is no centroid for a set in a non-Euclidean space and we need to come up with another measure.

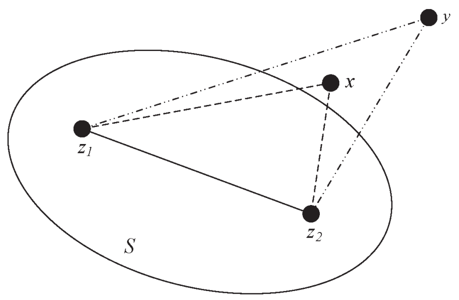


Fig. 24. An illustration of the triangular distance in $\mathcal{R}^2$.

Our idea for measuring the distance from a point x to a set S is to randomly choose two points z_1 and z_2 from S and consider the three sides of the triangle x, z_1 and z_2. Note that the triangular inequality guarantees that $d(x, z_1) + d(x, z_2) - d(z_1, z_2) \geq 0$. Moreover, if x is close to z_1 and z_2, then $d(x, z_1) + d(x, z_2) - d(z_1, z_2)$ should also be small. We illustrate such an intuition in Figure 24, where there are two points x and y and a set S in $\mathcal{R}^2$. Such an intuition leads to the following definition of triangular distance from a point x to a set S.

Definition 41. **(Triangular distance)** The *triangular distance* from a point x to a set S, denoted by $\Delta(x, S)$, is defined as follows:

$$\Delta(x, S) = \frac{1}{|S|^2} \sum_{z_1 \in S} \sum_{z_2 \in S} \Big(d(x, z_1) + d(x, z_2) - d(z_1, z_2) \Big). \tag{167}$$

In the following lemma, we show several properties of the triangular distance and its proof is given in Section 9.11.

Lemma 42.

(i)

$$\Delta(x, S) = 2\bar{d}(\{x\}, S) - \bar{d}(S, S) \geq 0. \tag{168}$$

(ii)

$$\Delta(x, S) = \gamma(x, x) - \frac{2}{|S|}\gamma(\{x\}, S) + \frac{1}{|S|^2}\gamma(S, S). \tag{169}$$

ALGORITHM 7: The K-sets Algorithm

```
Input: A data set $\Omega = \{x_1, x_2, \ldots, x_n\}$, a distance metric $d(\cdot,\cdot)$, and the number of sets $K$.
Output: A partition of sets $\{S_1, S_2, \ldots, S_K\}$.
(0) Initially, choose arbitrarily $K$ disjoint nonempty sets $S_1, \ldots, S_K$ as a partition of $\Omega$.
(1) for $i = 1, 2, \ldots, n$ do
    Compute the triangular distance $\Delta(x_i, S_k)$ for each set $S_k$ by using (168).
    Find the set to which the point $x_i$ is closest in terms of the triangular distance.
    Assign point $x_i$ to that set.
end
(2) Repeat from (1) until there is no further change.
```

(iii) Let $\mathcal{P} = \{S_k, k = 1, 2, \ldots, K\}$ be a partition of Ω. Then

$$\sum_{k=1}^{K}\sum_{x\in S_k}\Delta(x, S_k) = \sum_{x\in\Omega}\gamma(x,x) - Q_{\text{nor}}(\mathcal{P}). \tag{170}$$

(iv) Let $\mathcal{P} = \{S_k, k = 1, 2, \ldots, K\}$ be a partition of Ω, and $c(x)$ be the index of the set to which x belongs, i.e., $x \in S_{c(x)}$. Then

$$\begin{aligned}\sum_{k=1}^{K}\sum_{x\in S_k}\Delta(x, S_k) &= \sum_{k=1}^{K}\sum_{x\in S_k}\bar{d}(\{x\}, S_k) \\ &= \sum_{x\in\Omega}\bar{d}(\{x\}, S_{c(x)}).\end{aligned} \tag{171}$$

The first property of this lemma is to represent triangular distance by the average distance. The second property is to represent the triangular distance by the cohesion measure. Such a property plays an important role for the duality result in Section 8.4. The third property shows that the optimization problem for maximizing the normalized modularity $Q_{\text{nor}}(\mathcal{P})$ is equivalent to the optimization problem that minimizes the sum of the triangular distance of each point to its set. The fourth property further shows that such an optimization problem is also equivalent to the optimization problem that minimizes the sum of the average distance of each point to its set. Note that $\bar{d}(\{x\}, S_k) = \frac{1}{|S_k|}\sum_{y\in S_k} d(x,y)$. The objective for maximizing the normalized modularity $Q_{\text{nor}}(\mathcal{P})$ is also equivalent to minimize

$$\sum_{k=1}^{K}\frac{1}{|S_k|}\sum_{x\in S_k}\sum_{y\in S_k} d(x,y).$$

This is different from the K-median objective, the K-means objective and the min-sum objective addressed in [57].

In the following, we propose a partitional clustering algorithm, called the K-sets algorithm in Algorithm 7, based on the triangular distance. The algorithm is very simple. It starts from an arbitrary partition of the data points that contains K disjoint sets. Then for each data point, we assign the data point to the closest set in terms of the triangular distance. We repeat the process until there is no further change. Unlike the Lloyd iteration that needs two-step minimization, the K-sets algorithm only takes one-step minimization. This might give the K-sets algorithm the computational advantage over the K-medoids algorithms [42], [38], [43], [44].

In the following theorem, we show the convergence of the K-sets algorithm. Moreover, for $K = 2$, the K-sets algorithm yields two clusters. Its proof is given in Section 9.12.

Theorem 43.

(i) In the K-sets algorithm (based on the triangular distance), the normalized modularity is increasing when there is a change, i.e., a point is moved from one set to another. Thus, the algorithm converges to a local optimum of the normalized modularity.

(ii) Let $S_1, S_2, \ldots, S_K$ be the K sets when the algorithm converges. Then for all $i \neq j$, the two sets S_i and S_j are two clusters if these two sets are viewed in isolation (by removing the data points not in $S_i \cup S_j$ from Ω).

An immediate consequence of Theorem 43 (ii) is that for $K = 2$, the two sets S_1 and S_2 are clusters when the algorithm converges. However, we are not able to show that for $K \geq 3$ the K sets, $S_1, S_2, \ldots, S_K$, are clusters in Ω. On the other hand, we are not able to find a counterexample either. All the numerical examples that we have tested for $K \geq 3$ yield K clusters.

8.4 Duality between a cohesion measure and a distance metric

In this section, we show the duality result between a cohesion measure and a distance metric. In the following, we first provide a *general* definition for a cohesion measure.

Definition 44. A measure between two points x and y, denoted by $\beta(x,y)$, is called a *cohesion measure* for a set of data points Ω if it satisfies the following three properties:

(C1) (Symmetry) $\beta(x,y) = \beta(y,x)$ for all $x, y \in \Omega$.

(C2) (Zero-sum) For all $x \in \Omega$, $\sum_{y\in\Omega} \beta(x,y) = 0$.

(C3) (Triangular inequality) For all x, y, z in Ω,

$$\beta(x,x) + \beta(y,z) - \beta(x,z) - \beta(x,y) \geq 0. \tag{172}$$

In the following lemma, we show that the specific cohesion measure defined in Section 8.2 indeed satisfies (C1)–(C3) in Definition 44. Its proof is given in Section 9.13.

Lemma 45. Suppose that $d(\cdot,\cdot)$ is a distance metric for Ω, i.e., $d(\cdot,\cdot)$ satisfies (D1)–(D4). Let

$$\begin{aligned}\beta(x,y) \;=\; & \frac{1}{n}\sum_{z_2\in\Omega} d(z_2,y) + \frac{1}{n}\sum_{z_1\in\Omega} d(x,z_1) \\ & -\frac{1}{n^2}\sum_{z_2\in\Omega}\sum_{z_1\in\Omega} d(z_2,z_1) - d(x,y). \end{aligned} \tag{173}$$

Then $\beta(x,y)$ is a cohesion measure for Ω.

We know from (160) that the cohesion measure $\gamma(\cdot,\cdot)$ defined in Section 8.2 has the following representation:

$$\gamma(x,y) = \frac{1}{n}\sum_{z_2\in\Omega} d(z_2,y) + \frac{1}{n}\sum_{z_1\in\Omega} d(x,z_1)$$
$$-\frac{1}{n^2}\sum_{z_2\in\Omega}\sum_{z_1\in\Omega} d(z_2,z_1) - d(x,y).$$

As a result of Lemma 45, it also satisfies (C1)–(C3) in Definition 44. As such, we call the cohesion measure $\gamma(\cdot,\cdot)$ defined in Section 8.2 the *dual cohesion measure* of the distance metric $d(\cdot,\cdot)$.

On the other hand, if $\beta(x,y)$ is a cohesion measure for Ω, then there is an induced distance metric and it can be viewed as the *dual distance metric* of the cohesion measure $\beta(x,y)$. This is shown in the following lemma and its proof is given in Section 9.14.

ALGORITHM 8: The dual K-sets Algorithm

```
Input: A data set $\Omega = \{x_1, x_2, \ldots, x_n\}$, a cohesion measure $\gamma(\cdot,\cdot)$, and the number of sets $K$.
Output: A partition of sets $\{S_1, S_2, \ldots, S_K\}$.
(0) Initially, choose arbitrarily $K$ disjoint nonempty sets $S_1, \ldots, S_K$ as a partition of $\Omega$.
(1) for $i = 1, 2, \ldots, n$ do
    Compute the triangular distance $\Delta(x_i, S_k)$ for each set $S_k$ by using (178).
    Find the set to which the point $x_i$ is closest in terms of the triangular distance.
    Assign point $x_i$ to that set.
end
(2) Repeat from (1) until there is no further change.
```

Lemma 46. Suppose that $\beta(\cdot,\cdot)$ is a cohesion measure for Ω. Let

$$d(x,y) = (\beta(x,x) + \beta(y,y))/2 - \beta(x,y). \tag{174}$$

Then $d(\cdot,\cdot)$ is a distance metric that satisfies (D1)–(D4).

In the following theorem, we show the duality result. Its proof is given in Section 9.15.

Theorem 47. Consider a set of data points Ω. For a distance metric $d(\cdot,\cdot)$ that satisfies (D1)–(D4), let

$$\begin{aligned} d^*(x,y) &= \frac{1}{n}\sum_{z_2\in\Omega} d(z_2,y) + \frac{1}{n}\sum_{z_1\in\Omega} d(x,z_1) \\ &\quad -\frac{1}{n^2}\sum_{z_2\in\Omega}\sum_{z_1\in\Omega} d(z_2,z_1) - d(x,y) \end{aligned} \tag{175}$$

be the dual cohesion measure of $d(\cdot,\cdot)$. On the other hand, For a cohesion measure $\beta(\cdot,\cdot)$ that satisfies (C1)–(C3), let

$$\beta^*(x,y) = (\beta(x,x) + \beta(y,y))/2 - \beta(x,y) \tag{176}$$

be the dual distance metric of $\beta(\cdot,\cdot)$. Then $d^{**}(x,y) = d(x,y)$ and $\beta^{**}(x,y) = \beta(x,y)$ for all $x, y \in \Omega$.

8.5 The dual K-sets algorithm

For the K-sets algorithm, we need to have a distance metric. In view of the duality theorem between a cohesion measure and a distance metric, we propose the dual K-sets algorithm in Algorithm 8 that uses a cohesion measure. As before, for a cohesion measure $\gamma(\cdot,\cdot)$ between two points, we define the cohesion measure between two sets S_1 and S_2 as

$$\gamma(S_1,S_2) = \sum_{x\in S_1}\sum_{y\in S_2} \gamma(x,y). \tag{177}$$

Also, note from (169) that the triangular distance from a point x to a set S can be computed by using the cohesion measure as follows:

$$\Delta(x,S) = \gamma(x,x) - \frac{2}{|S|}\gamma(\{x\},S) + \frac{1}{|S|^2}\gamma(S,S). \tag{178}$$

As a direct result of the duality theorem in Theorem 47 and the convergence result of the K-sets algorithm in Theorem 43, we have the following convergence result for the dual K-sets algorithm.

Corollary 48. As in (166), we define the normalized modularity as $\sum_{k=1}^{K} \frac{1}{|S_k|}\gamma(S_k, S_k)$. For the dual K-sets algorithm, the normalized modularity is increasing when there is a change, i.e., a point is moved from one set to another. Thus, the algorithm converges to a local optimum of the normalized modularity. Moreover, for $K = 2$, the dual K-sets algorithm yields two clusters when the algorithm converges.

8.6 Connections to the kernel K-means algorithm

In this section, we show the connection between the dual K-sets algorithm and the kernel K-means algorithm in the literature (see e.g., [117], [119]). As shown in Section 7.4, the kernel K-means algorithm is a special case of the spectral clustering algorithm when the similarity measure is a positive definite matrix. In [119], it was shown that a general weighted kernel K-means objective is mathematically equivalent to a weighted graph clustering objective, including the ratio cut, normalized cut, and ratio association criteria.

Let us consider the $n \times n$ matrix $\Gamma = (\gamma_{i,j})$ with $\gamma_{i,j} = \gamma(x_i, x_j)$ being the cohesion measure between x_i and x_j. Call the matrix Γ the cohesion matrix (corresponding to the cohesion measure $\gamma(\cdot,\cdot)$). Since $\gamma(x_i, x_j) = \gamma(x_j, x_i)$, the matrix Γ is symmetric and thus has real eigenvalues $\lambda_k, k = 1, 2, \ldots, n$. Let $\mathbf{I}$ be the $n \times n$ identity matrix and $\sigma \geq -\min_{1 \leq k \leq n} \lambda_k$. Then the matrix $\tilde{\Gamma} = \sigma\mathbf{I} + \Gamma$ is positive semi-definite as its n eigenvalues $\tilde{\lambda}_k = \sigma + \lambda_k$, $k = 1, 2, \ldots, N$ are all nonnegative. Let $v_k = (v_{k,1}, v_{k,2}, \ldots, v_{k,n})^T$, $k = 1, 2, \ldots, n$, be the eigenvector of Γ corresponding to the eigenvalue λ_k. Then v_k is also the eigenvector of $\tilde{\Gamma}$ corresponding to the eigenvalue $\tilde{\lambda}_k$. Thus, we can decompose the matrix $\tilde{\Gamma}$ as follows:

$$\tilde{\Gamma} = \sum_{k=1}^{n} \tilde{\lambda}_k v_k v_k^T, \tag{179}$$

where v_k^T is the transpose of v_k. Now we choose the mapping $\phi : \Omega \mapsto \mathcal{R}^n$ as follows:

$$\phi(x_i) = \Big(\sqrt{\tilde{\lambda}_1} v_{1,i}, \sqrt{\tilde{\lambda}_2} v_{2,i}, \ldots, \sqrt{\tilde{\lambda}_n} v_{n,i}\Big)^T, \tag{180}$$

for $i = 1, 2 \ldots, n$. Note that

$$\begin{aligned} \phi(x_i)^T \cdot \phi(x_j) &= \sum_{k=1}^{n} \tilde{\lambda}_k v_{k,i} v_{k,j} \\ &= (\tilde{\Gamma})_{i,j} = \sigma\delta_{i,j} + \gamma(x_i, x_j), \end{aligned} \tag{181}$$

where $\delta_{i,j} = 1$ if $i = j$ and 0 otherwise.

The "centroid" of a set S can be represented by the corresponding centroid in $\mathcal{R}^n$, i.e.,

$$\frac{1}{|S|}\sum_{y \in S} \phi(y), \tag{182}$$

and the square of the "distance" between a point x and the "centroid" of a set S is

$$\begin{aligned} &\Big(\phi(x) - \frac{1}{|S|}\sum_{y\in S}\phi(y)\Big)^T \cdot \Big(\phi(x) - \frac{1}{|S|}\sum_{y\in S}\phi(y)\Big) \\ &= (1 - \frac{2}{|S|}1_{\{x\in S\}} + \frac{1}{|S|})\sigma + \gamma(x,x) - \frac{2}{|S|}\gamma(\{x\}, S) \\ &\quad + \frac{1}{|S|^2}\gamma(S,S), \end{aligned} \tag{183}$$

where $1_{\{x \in S\}}$ is the indicator function that has value 1 if x is in S and 0 otherwise. In view of (169), we then have

$$\begin{aligned}&\Big(\phi(x) - \frac{1}{|S|}\sum_{y \in S}\phi(y)\Big)^T \cdot \Big(\phi(x) - \frac{1}{|S|}\sum_{y \in S}\phi(y)\Big) \\ &= (1 - \frac{2}{|S|}1_{\{x \in S\}} + \frac{1}{|S|})\sigma + \Delta(x, S),\end{aligned} \tag{184}$$

where $\Delta(x, S)$ is the triangular distance from a point x to a set S. Thus, the square of the "distance" between a point x and the "centroid" of a set S is $(1 - \frac{1}{|S|})\sigma + \Delta(x, S)$ for a point $x \in S$ and $(1 + \frac{1}{|S|})\sigma + \Delta(x, S)$ for a point $x \not\in S$. In particular, when $\sigma = 0$, the dual K-sets algorithm is the same as the sequential kernel K-means algorithm for the kernel $\tilde{\Gamma}$. Unfortunately, the matrix $\tilde{\Gamma}$ may not be positive semi-definite if σ is chosen to be 0. As indicated in [117], a large σ decreases (resp. increases) the distance from a point x to a set S that contains (resp. does not contain) that point. As such, a point is more unlikely to move from one set to another set and the kernel K-means algorithm is thus more likely to be trapped in a local optimum.

To summarize, the dual K-sets algorithm operates in the same way as a sequential version of the classical kernel K-means algorithm by viewing the matrix Γ as a kernel. However, there are two key differences between the dual K-sets algorithm and the classical kernel K-means algorithm: (i) the dual K-sets algorithm guarantees the convergence even though the matrix Γ from a cohesion measure is not positive semi-definite, and (ii) the dual K-sets algorithm can only be operated *sequentially* and the kernel K-means algorithm can be operated in batches. To further illustrate the difference between these two algorithms, we show in the following two examples that a cohesion matrix may not be positive semi-definite and a positive semi-definite matrix may not be a cohesion matrix.

Example 49. *In this example, we show there is a cohesion matrix Γ that is not a positive semi-definite matrix.*

$$\Gamma = \begin{pmatrix} 0.44 & 0.04 & 0.04 & 0.04 & -0.56 \\ 0.04 & 0.64 & -0.36 & -0.36 & 0.04 \\ 0.04 & -0.36 & 0.64 & -0.36 & 0.04 \\ 0.04 & -0.36 & -0.36 & 0.64 & 0.04 \\ -0.56 & 0.04 & 0.04 & 0.04 & 0.44 \end{pmatrix}. \tag{185}$$

The eigenvalues of this matrix are -0.2, 0, 1, 1, and 1.

Example 50. *In this example, we show there is a positive semi-definite matrix $M = (m_{i,j})$ that is not an cohesion matrix.*

$$M = \begin{pmatrix} 0.375 & -0.025 & -0.325 & -0.025 \\ -0.025 & 0.875 & -0.025 & -0.825 \\ -0.325 & -0.025 & 0.375 & -0.025 \\ -0.025 & -0.825 & -0.025 & 0.875 \end{pmatrix}. \tag{186}$$

The eigenvalues of this matrix are 0, 0.1, 0.7, and 1.7. Even though the matrix M is symmetric and has all its row sums and column sums being 0, it is still not a cohesion matrix as $m_{1,1} - m_{1,2} - m_{1,4} + m_{2,4} = -0.4 < 0$.

8.7 Constructing a cohesion measure from a similarity measure

A similarity measure is in general defined as a bivariate function of two *distinct* data points and it is often characterized by a square matrix without specifying the diagonal elements. In the following, we show how one can construct a cohesion measure from a symmetric bivariate function by further specifying the diagonal elements. Its proof is given in Section 9.16.

Proposition 51. Suppose a bivariate function $\beta_0 : \Omega \times \Omega \mapsto \mathcal{R}$ is symmetric, i.e., $\beta_0(x, y) = \beta_0(y, x)$. Let $\beta_1(x, y) = \beta_0(x, y)$ for all $x \neq y$ and specify $\beta_1(x, x)$ such that

$$\beta_1(x, x) \geq \max_{x \neq y \neq z} [\beta_1(x, z) + \beta_1(x, y) - \beta_1(y, z)]. \tag{187}$$

Also, let

$$\begin{aligned}\beta(x, y) = \beta_1(x, y) - \frac{1}{n} \sum_{z_1 \in \Omega} \beta_1(z_1, y) \\ -\frac{1}{n} \sum_{z_2 \in \Omega} \beta_1(x, z_2) + \frac{1}{n^2} \sum_{z_1 \in \Omega} \sum_{z_2 \in \Omega} \beta_1(z_1, z_2).\end{aligned} \tag{188}$$

Then $\beta(x, y)$ is a cohesion measure for Ω.

We note that one simple choice for specifying $\beta_1(x, x)$ in (187) is to set

$$\beta_1(x, x) = 2\beta_{\max} - \beta_{\min}, \tag{189}$$

where

$$\beta_{\max} = \max_{x \neq y} \beta(x, y), \tag{190}$$

and

$$\beta_{\min} = \min_{x \neq y} \beta(x, y). \tag{191}$$

In particular, if the similarity measure $\beta(x, y)$ only has values 0 and 1 as in the adjacency matrix of a simple undirected graph, then one can simply choose $\beta_1(x, x) = 2$ for all x.

Example 52. **(A cohesion measure for a graph)** *As an illustrating example, suppose $A = (a_{i,j})$ is the $n \times n$ adjacency matrix of a simple undirected graph with $a_{i,j} = 1$ if there is an edge between node i and node j and 0 otherwise. Let $k_i = \sum_{j=1}^{n} a_{i,j}$ be the degree of node i and $m = \frac{1}{2} \sum_{i=1}^{n} k_i$ be the total number of edges in the graph. Then one can simply let $\beta_1(i, j) = 2\delta_{i,j} + a_{i,j}$, where $\delta_{i,j} = 1$ if $i = j$ and 0 otherwise. By doing so, we then have the following cohesion measure*

$$\beta(i, j) = 2\delta_{i,j} + a_{i,j} - \frac{2 + k_i}{n} - \frac{2 + k_j}{n} + \frac{2m + 2n}{n^2}. \tag{192}$$

We note that such a cohesion measure is known as the deviation to indetermination null model in [120].

8.8 The K-sets$^+$ algorithm: clustering data points in a semi-metric space

Though the extensions of the duality result and the equivalent statements for clusters to semi-metric spaces are basically the same as those in [67], one problem arises when extending the K-sets algorithm to a semi-metric space. The key problem is that the *triangular distance (Δ-distance)* defined in the K-sets algorithm (see Definition 53 below) might not be nonnegative in a semi-metric space.

Definition 53. **(Δ-distance [67])** For a symmetric bivariate function $\gamma(\cdot,\cdot)$ on a set of data points $\Omega = \{x_1, x_2, \ldots, x_n\}$, the *$\Delta$-distance* from a point x to a set S, denoted by $\Delta(x,S)$, is defined as follows:

$$\Delta(x,S) = \gamma(x,x) - \frac{2}{|S|}\gamma(x,S) + \frac{1}{|S|^2}\gamma(S,S), \tag{193}$$

where $\gamma(S_1,S_2)$ is defined (162).

Note from (173) that the Δ-distance from a point x to a set S in a semi-metric space can also be written as follows:

$$\Delta(x,S) = \frac{1}{|S|^2}\sum_{z_1\in S}\sum_{z_2\in S}\Big(d(x,z_1)+d(x,z_2)-d(z_1,z_2)\Big). \tag{194}$$

Now consider the data set of three points $\Omega = \{x,y,z\}$ with the semi-metric $d(\cdot,\cdot)$ in Table 7. For $S=\{y,z\}$, one can easily compute from (194) that $\Delta(x,S) = -1 < 0$.

TABLE 7
A data set of three points $\Omega = \{x,y,z\}$ with a semi-metric $d(\cdot,\cdot)$.

$d(\cdot,\cdot)$	x	y	z
x	0	1	1
y	1	0	6
z	1	6	0

Since the Δ-distance might not be nonnegative in a semi-metric space, the proofs for the convergence and the performance guarantee of the K-sets algorithm in [67] are no longer valid. Fortunately, the Δ-distance in a semi-metric space has the following (weaker) nonnegative property that will enable us to prove the performance guarantee of the K-sets$^+$ algorithm (defined in Algorithm 9 later) for clustering data points in a semi-metric space.

Proposition 54. Consider a data set $\Omega = \{x_1, x_2, \ldots, x_n\}$ with a semi-metric $d(\cdot,\cdot)$. For any subset S of Ω,

$$\sum_{x\in S}\Delta(x,S) = |S|\bar{d}(S,S) \geq 0. \tag{195}$$

The proof of (195) in Proposition 54 follows directly from (194) and (164). To introduce the K-sets$^+$ algorithm, we first define the adjusted Δ-distance in Definition 55 below.

Definition 55. **(Adjusted Δ-distance)** The *adjusted Δ-distance* from a point x to a set S, denoted by $\Delta_{\text{a}}(x,S)$, is defined as follows:

$$\Delta_{\text{a}}(x,S) = \begin{cases} \frac{|S|}{|S|+1}\Delta(x,S), & \text{if } x \notin S, \\ \frac{|S|}{|S|-1}\Delta(x,S), & \text{if } x \in S \text{ and } |S|>1, \\ -\infty, & \text{if } x \in S \text{ and } |S|=1. \end{cases} \tag{196}$$

Instead of using the Δ-distance for the assignment of a data point in the K-sets algorithm, we use the adjusted Δ-distance for the assignment in the K-sets$^+$ algorithm. We outline the K-sets$^+$

ALGORITHM 9: The K-sets$^+$ Algorithm

```
Input: A data set $\Omega = \{x_1, x_2, \ldots, x_n\}$, a symmetric matrix $\Gamma = (\gamma(\cdot,\cdot))$ and the number of
    sets $K$.
Output: A partition of sets $\{S_1, S_2, \ldots, S_K\}$.
(0) Initially, choose arbitrarily $K$ disjoint nonempty sets $S_1, \ldots, S_K$ as a partition of $\Omega$.
(1) for $i = 1, 2, \ldots, n$ do
    Compute the adjusted $\Delta$-distance $\Delta_{\mathsf{a}}(x_i, S_k)$ for each set $S_k$ by using (193) and (196).
    Find the set to which the point $x_i$ is closest in terms of the adjusted $\Delta$-distance. Assign
    that point $x_i$ to that set.
end
(2) Repeat from (1) until there is no further change.
```

algorithm in Algorithm 9. Note that in Algorithm 9, the bivariate function $\gamma(\cdot,\cdot)$ is required to be symmetric, i.e., $\gamma(x,y) = \gamma(y,x)$. If $\gamma(\cdot,\cdot)$ is not symmetric, one may consider using $\hat{\gamma}(x,y) = (\gamma(x,y) + \gamma(y,x))/2$.

In the following theorem, we show the convergence and the performance guarantee of the K-sets$^+$ algorithm. The proof of Theorem 56 is given in Section 9.17.

Theorem 56. For a data set $\Omega = \{x_1, x_2, \ldots, x_n\}$ with a *symmetric* matrix $\Gamma = (\gamma(\cdot,\cdot))$, consider the clustering problem that finds a partition $\{S_1, S_2, \ldots, S_K\}$ of Ω with a fixed K that maximizes the objective function $\sum_{k=1}^{K} \frac{1}{|S_k|}\gamma(S_k, S_k)$.

(i) The K-sets$^+$ algorithm in Algorithm 9 converges monotonically to a local optimum of the optimization problem in a finite number of iterations.

(ii) Suppose that $\gamma(\cdot,\cdot)$ is a semi-cohesion measure. Let $S_1, S_2, \ldots, S_K$ be the K sets when the algorithm converges. Then for all $i \neq j$, the two sets S_i and S_j are two clusters if these two sets are viewed in isolation (by removing the data points not in $S_i \cup S_j$ from Ω).

In particular, if $K = 2$, it then follows from Theorem 56(ii) that the K-sets$^+$ algorithm yields two clusters for data points in a semi-metric space.

8.9 Clustering with a symmetric similarity measure

In this section, we further extend the applicability of the K-sets$^+$ algorithm to the clustering problem with a symmetric similarity measure. A similarity measure is generally referred to as a bivariate function that measures how similar two data points are. The clustering problem with a similarity measure is to cluster data points so that similar data points are clustered together. For a symmetric similarity measure $\gamma(\cdot,\cdot)$, we have shown in Theorem 56(i) that the K-sets$^+$ algorithm in Algorithm 9 converges monotonically to a local optimum of the optimization problem $\sum_{k=1}^{K} \frac{1}{|S_k|}\gamma(S_k, S_k)$ within a finite number of iterations. Thus, the K-sets$^+$ algorithm can be applied for clustering with a symmetric similarity measure. But what is the physical meaning of the sets returned by the K-sets$^+$ algorithm for such a symmetric similarity measure? In order to answer this question, we show there is a natural semi-cohesion measure from a symmetric similarity measure and the K-sets$^+$ algorithm that uses this semi-cohesion measure converges to the same partition as that using the original symmetric similarity measure (if they both use the same initial partition). As a direct consequence of Theorem 56(ii), any two sets returned by the K-sets$^+$ algorithm for such a symmetric similarity measure are clusters with respect to the semi-cohesion measure when they are viewed in isolation.

In Lemma 57 below, we first show how one can map a symmetric similarity measure to a semi-cohesion measure. The proof is given in Section 9.18.

Lemma 57. For a symmetric similarity measure $\gamma(\cdot,\cdot)$, let

$$\begin{aligned}\tilde{\gamma}(x,y) &= \gamma(x,y) - \frac{1}{n}\gamma(x,\Omega) - \frac{1}{n}\gamma(y,\Omega) \\ &+ \frac{1}{n^2}\gamma(\Omega,\Omega) + \sigma\delta(x,y) - \frac{\sigma}{n}, \end{aligned} \tag{197}$$

where $\delta(x,y)$ is the usual δ function (that has value 1 if $x=y$ and 0 otherwise), and σ is a constant that satisfies

$$\sigma \geq \max_{x\neq y}[\gamma(x,y) - (\gamma(x,x)+\gamma(y,y))/2]. \tag{198}$$

Then the bivariate function $\tilde{\gamma}(\cdot,\cdot)$ in (198) is a *semi-cohesion measure* for Ω, i.e., it satisfies (C1), (C2) and (C3).

In the following lemma, we further establish the connections for the Δ-distance and the adjusted Δ-distance between the original symmetric similarity measure $\gamma(\cdot,\cdot)$ and the semi-cohesion measure $\tilde{\gamma}(\cdot,\cdot)$ in (197). The proof is given in Section 9.19.

Lemma 58. Let $\Delta(x,S)$ (resp. $\tilde{\Delta}(x,S)$) be the Δ-distance from a point x to a set S with respect to $\gamma(\cdot,\cdot)$ (resp. $\tilde{\gamma}(\cdot,\cdot)$). Also, let $\Delta_{\mathsf{a}}(x,S)$ (resp. $\tilde{\Delta}_{\mathsf{a}}(x,S)$) be the adjusted Δ-distance from a point x to a set S with respect to $\gamma(\cdot,\cdot)$ (resp. $\tilde{\gamma}(\cdot,\cdot)$). Then

$$\tilde{\Delta}(x,S) = \begin{cases} \Delta(x,S) + \sigma(1-\frac{1}{|S|}), & \text{if } x \notin S, \\ \Delta(x,S) + \sigma(1+\frac{1}{|S|}), & \text{if } x \in S. \end{cases} \tag{199}$$

Moreover,

$$\tilde{\Delta}_{\mathsf{a}}(x,S) = \Delta_{\mathsf{a}}(x,S) + \sigma. \tag{200}$$

It is easy to see that for any partition $S_1, S_2, \dots, , S_K$

$$\begin{aligned}&\sum_{k=1}^{K} \frac{1}{|S_k|}\tilde{\gamma}(S_k,S_k) \\ &= \sum_{k=1}^{K} \frac{1}{|S_k|}\gamma(S_k,S_k) - \frac{1}{n}\gamma(\Omega,\Omega) + (K-1)\sigma. \end{aligned} \tag{201}$$

Thus, optimizing $\sum_{k=1}^{K}\frac{1}{|S_k|}\gamma(S_k,S_k)$ with respect to the symmetric similarity measure $\gamma(\cdot,\cdot)$ is equivalent to optimizing $\sum_{k=1}^{K}\frac{1}{|S_k|}\tilde{\gamma}(S_k,S_k)$ with respect to the semi-cohesion measure $\tilde{\gamma}(\cdot,\cdot)$. Since

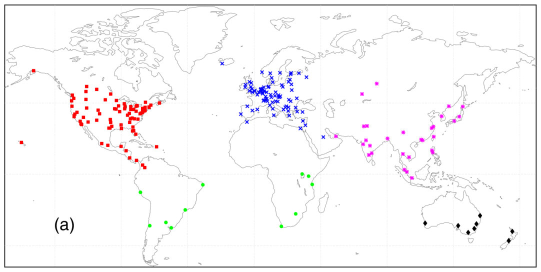
$$\tilde{\Delta}_{\mathsf{a}}(x,S) = \Delta_{\mathsf{a}}(x,S) + \sigma, \tag{202}$$

we conclude that for these two optimization problems, the K-sets$^+$ algorithm converges to the same partition if they both use the same initial partition.

Note that the K-means algorithm needs the data points to be in a Euclidean space, the kernel K-means algorithm needs the data points to be mapped into some Euclidean space, and the K-sets algorithm needs the data points to be in a metric space. The result in Lemma 58 shows that the K-sets$^+$ algorithm lifts all the constraints on the data points and it can be operated merely by a symmetric similarity measure.

8.10 Clustering of a real network

In this section, we test the K-sets$^+$ algorithm on the real network from the WonderNetwork website [121]. In this dataset, there are 216 servers in different locations and the latency (measured by the round trip time) between any two servers of these 216 servers are recorded in real time. The dataset in our experiment is a snapshot on Sept. 24, 2016. For this dataset, the triangular inequality is not always satisfied. For example, we notice that latency(Adelaide, Athens)=250, latency(Athens, Albany)=138, latency(Adelaide, Albany)=400, and $250 + 138 \leq 400$. In addition to the latency data, the WonderNetwork website also provides the geographic location of each server. We then use the Haversine formula to compute the distance between any two servers. In the WonderNetwork dataset, the latency measure from location L_1 to location L_2 is slightly different from that from location L_2 to location L_1. To ensure that the latency measure is a semi-metric, we simply symmetrize the latency matrix by taking the average of the latency measures from both directions. In our experiments, the number of clusters K is set to 5. We run 20 times of the K-sets$^+$ algorithm by using the distance matrix and the latency matrix, respectively. In each of the 20 trials, the initial partition is randomly selected. The output partition that has the best objective value from these 20 trials is selected. The results for the distance matrix and the latency matrix are shown in Figure 25(a) and (b), respectively. In Figure 25(a) and (b), the servers that are in the same cluster are marked with the same colored marker. In view of Figure 25(a), we can observe that almost all the servers are partitioned into densely packed clusters except for the servers in South America and Africa. On the other hand, as shown in Figure 25(b), the servers in South America and Africa are merged into other clusters. To shed more light on these interesting differences, we compare the findings in Figure 25(b) to the Submarine Cable Map [122] (which records the currently active submarine cables). We notice that there are many cables placed around South America, connecting to the Caribbean and then to the East Coast of the United States. These cables greatly reduce the latency from South America to North America and thus cause the servers in South America to be clustered with the servers in North America. Similarly, there are a few connected cables between Africa and Europe. Therefore, the servers in Africa and Europe are clustered together. Due to many directly connected cables from Dubai to the Singapore Strait, servers around India are not clustered with the other servers in Asia. In particular, there are two servers marked with green dots, located in Jakarta and Singapore, that are clustered with servers in India even though they are geographically closer to the East Asia. These outliers have low latency to communicate with those servers in India. Finally, there are three servers, two in Russia and one in Lahore, that have low latency to the servers in Europe and they are clustered with the servers in Europe.



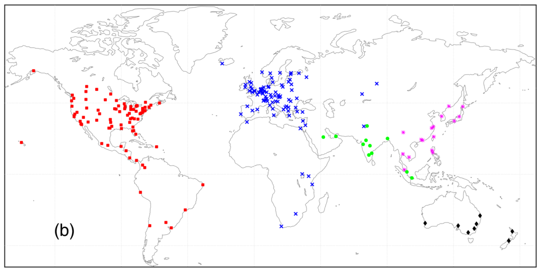


Fig. 25. Clustering for the WonderNetwork dataset: (a) the (geographic) distance matrix and (b) the latency matrix.

9 Proofs

9.1 Proof of Proposition 7

Since the relative centrality is a conditional probability and the centrality is a probability, the properties in (i),(ii) and (iii) follow trivially from the property of probability measures.

(iv) From the symmetric property of $p(\cdot,\cdot)$, the definitions of relative centrality and centrality in (18) and (16), it follows that

$$\begin{aligned} &C(S_1)C(S_2|S_1) = \mathsf{P}(W \in S_1)\mathsf{P}(W \in S_2|V \in S_1) \\ &= \mathsf{P}(V \in S_1)\mathsf{P}(W \in S_2|V \in S_1) = \mathsf{P}(V \in S_1, W \in S_2) \\ &= \mathsf{P}(V \in S_2, W \in S_1). \end{aligned} \tag{203}$$

Similarly, we also have

$$C(S_2)C(S_1|S_2) = \mathsf{P}(V \in S_1, W \in S_2) = \mathsf{P}(V \in S_2, W \in S_1). \tag{204}$$

Thus,

$$C(S_1)C(S_2|S_1) = C(S_2)C(S_1|S_2).$$

9.2 Proof of Theorem 12

We first prove that the first four statements are equivalent by showing (i)$\Rightarrow$ (ii)$\Rightarrow$ (iii)$\Rightarrow$ (iv)$\Rightarrow$ (i).

(i) $\Rightarrow$ (ii): Note from Proposition 7 (i) and (ii) that $C(S|S) + C(S^c|S) = C(V_g|S) = 1$ and $C(S) + C(S^c) = C(V_g) = 1$. It then follows from the reciprocal property in Proposition 7(iv) that

$$\begin{aligned} &C(S^c)(C(S|S) - C(S|S^c)) \\ &= C(S^c)C(S|S) - C(S^c)C(S|S^c) \\ &= (1 - C(S))C(S|S) - C(S)C(S^c|S) \\ &= (1 - C(S))C(S|S) - C(S)(1 - C(S|S)) \\ &= C(S|S) - C(S) = Str(S) \geq 0. \end{aligned}$$

As we assume that $0 < C(S) < 1$, we also have $0 < C(S^c) < 1$. Thus,

$$C(S|S) - C(S|S^c) \geq 0.$$

(ii) $\Rightarrow$ (iii): Since we assume that $C(S|S) \geq C(S|S^c)$, we have from $C(S|S)+C(S^c|S) = C(V_g|S) = 1$ that

$$1 = C(S|S) + C(S^c|S) \geq C(S|S^c) + C(S^c|S).$$

Multiplying both sides by $C(S^c)$ yields

$$C(S^c) \geq C(S^c)C(S|S^c) + C(S^c)C(S^c|S).$$

From the reciprocal property in Proposition 7(iv) and $C(S) + C(S^c) = C(V_g) = 1$, it follows that

$$\begin{aligned} C(S^c) &\geq C(S)C(S^c|S) + C(S^c)C(S^c|S) \\ &= (C(S) + C(S^c))C(S^c|S) \\ &= C(S^c|S). \end{aligned}$$

(iii) $\Rightarrow$ (iv): Note from the reciprocal property in Proposition 7(iv) that

$$C(S)C(S^c|S) = C(S^c)C(S|S^c). \tag{205}$$

It then follows from $C(S^c|S) \leq C(S^c)$ that $C(S|S^c) \leq C(S)$.

(iv) $\Rightarrow$ (i): Since we assume that $C(S|S^c) \leq C(S)$, it follows from (205) that $C(S^c|S) \leq C(S^c)$. In conjunction with $C(S|S) + C(S^c|S) = C(V_g|S) = 1$ and $C(S) + C(S^c) = C(V_g) = 1$, we have

$$C(S|S) - C(S) = C(S^c) - C(S^c|S) \geq 0.$$

Now we show that and (iv) and (v) are equivalent. Since $C(S|S^c) + C(S^c|S^c) = C(V_g|S^c) = 1$ and $C(S) + C(S^c) = C(V_g) = 1$, we have

$$C(S|S^c) - C(S) = C(S^c) - C(S^c|S^c) = -Str(S^c).$$

Thus, $C(S|S^c) \leq C(S)$ if and only if $Str(S^c) \geq 0$.

Replacing S by S^c, we see that (v) and (vi) are also equivalent because (i) and (ii) are equivalent.

9.3 Proof of Theorem 18

(i) Since we choose two sets that have a nonnegative correlation measure, i.e., $q(S_i, S_j) \geq 0$, to merge, it is easy to see from (67) and (61) that the modularity is non-decreasing in every iteration.

(ii) Suppose that there is only one set left. Then this set is V_g and it is the trivial community. On the other hand, suppose that there are $C \geq 2$ sets $\{S_1, S_2, \ldots, S_C\}$ left when the algorithm converges. Then we know that $q(S_i, S_j) < 0$ for $i \neq j$.

Note from (58) and (59) that for any node v,

$$q(v, V_g) = \sum_{w \in V_g} q(v, w) = 0. \tag{206}$$

Thus,

$$q(S_i, V_g) = \sum_{v \in S_i} q(v, V_g) = 0. \tag{207}$$

Since $\{S_1, S_2, \ldots, S_C\}$ is a partition of V_g, it then follows that

$$0 = q(S_i, V_g) = q(S_i, S_i) + \sum_{j \neq i} q(S_i, S_j). \tag{208}$$

Since $q(S_i, S_j) < 0$ for $i \neq j$, we conclude that $q(S_i, S_i) > 0$ and thus S_i is a community.

9.4 Proof of Corollary 20

Suppose that S_i and S_j are merged into the new set S_k. According to the update rules in (68) and the symmetric property of $q(\cdot,\cdot)$, we know that

$$q(S_k, S_\ell) = q(S_\ell, S_k) = q(S_i, S_\ell) + q(S_j, S_\ell) = q(S_i, S_\ell) + q(S_\ell, S_j),$$

for all $\ell \neq k$. Thus,

$$\bar{q}(S_k, S_\ell) = \frac{|S_i|}{|S_i| + |S_j|}\bar{q}(S_i, S_\ell) + \frac{|S_j|}{|S_i| + |S_j|}\bar{q}(S_\ell, S_j).$$

Since we select the two sets with the *largest* average correlation measure in each merge operation, we have $\bar{q}(S_i, S_\ell) \le \bar{q}(S_i, S_j)$ and $\bar{q}(S_\ell, S_j) \le \bar{q}(S_i, S_j)$. These then lead to

$$\bar{q}(S_k, S_\ell) \le \bar{q}(S_i, S_j).$$

Thus, $\bar{q}(S_i, S_j)$ is not less than the average correlation measure between any two sets after the merge operation. As such, the average correlation measure at each merge is non-increasing.

9.5 Proof of Theorem 21

It suffices to show that if node v is in a set S_1 and $q_0(v, S_2) > q_0(v, S_1)$, then move node v from S_1 to S_2 increases the value of the modularity. Also let $\mathcal{P}$ (resp. $\mathcal{P}'$) be the partition before (resp. after) the change. Since $q_0(\cdot,\cdot)$ is symmetric and $q_0(v,v) = 0$, it follows from (71) that

$$\begin{aligned}
&Q(\mathcal{P}') - Q(\mathcal{P}) \\
&= q_0(S_1\backslash\{v\}, S_1\backslash\{v\}) + q_0(S_2 \cup \{v\}, S_2 \cup \{v\}) \\
&\quad - q_0(S_1, S_1) - q_0(S_2, S_2) \\
&= 2\Big(q_0(v, S_2) - q_0(v, S_1)\Big) > 0.
\end{aligned}$$

As the modularity is non-decreasing after a change of the partition, there is no loop in the algorithm. Since the number of partitions is finite, the partitional algorithm thus converges in a finite number of steps.

9.6 Proof of Lemma 23

Note from (53) that

$$\hat{Q}(\mathcal{P}^a) = \sum_{k=1}^{K} \sum_{c_1 \in S_k} \sum_{c_2 \in S_k} (\hat{p}(c_1, c_2) - \hat{p}_V(c_1)\hat{p}_W(c_2)), \tag{209}$$

where

$$\hat{p}_V(c_1) = \hat{p}_W(c_1) = \sum_{c_2=1}^{C} \hat{p}(c_1, c_2) \tag{210}$$

is the marginal distribution of the bivariate distribution $\hat{p}(\cdot,\cdot)$. In view of (75) and (76), it is easy to verify that

$$\sum_{c_1 \in S_k} \sum_{c_2 \in S_k} \hat{p}(c_1, c_2) = \sum_{u \in S_k^o} \sum_{v \in S_k^o} p(u, v). \tag{211}$$

Similarly,

$$\begin{aligned}
\sum_{c_1 \in S_k} \hat{p}_V(c_1) &= \sum_{c_1 \in S_k} \sum_{c_2=1}^{C} \hat{p}(c_1, c_2) \\
&= \sum_{u \in S_k^o} \sum_{v \in V_g} p(u, v) = \sum_{u \in S_k^o} p_V(u).
\end{aligned} \tag{212}$$

From (209), (211) and (212), it then follows that $\hat{Q}(\mathcal{P}^a) = Q(\mathcal{P}^o)$.

9.7 Proof of Theorem 24

(i) This is a direct consequence of the modularity increasing result in Theorem 21 and the modularity preserving result in Lemma 23.

(ii) Suppose that Algorithm 3 returns a partition $\{S_1, S_2, \ldots, S_C\}$. Observe from (F2) of Algorithm 3 that the algorithm terminates when the deepest level of the partitional algorithm returns the same partition as its input partition. Let $p^*(u,v)$, $\tilde{p}^*(u,v)$, and $q^*(u,v)$ be the bivariate distribution that has the same marginal distributions, the symmetric bivariate distribution and the correlation measure (see (58)) at the deepest level, respectively. Also, let $q_0^*(u,v) = q^*(u,v)$ for $u \neq v$ and 0 otherwise. As the initial partition in (F0) is the partition that contains a single element in each set, the condition that the output partition is the same as the input partition (see (P1) in Algorithm 2) implies that

$$q^*(u,v) = q_0^*(u,v) \leq q_0^*(v,v) = 0$$

for all $u \neq v$ and $\tilde{p}^*(u,v) > 0$. On the other hand, if $\tilde{p}^*(u,v) = 0$, then $q^*(u,v) \leq 0$. Thus, we conclude that in the deepest level $q^*(u,v) \leq 0$ for all $u \neq v$. Let S_u be the set recursively expanded from the (super)giant node u in the deepest level. Analogous to the modularity preserving proof in Lemma 23, one can easily show that $q(S_u, S_v) = q^*(u,v)$ and thus $q(S_u, S_v) \leq 0$ for all $S_u \neq S_v$. For a correlation measure, we know that $\sum_{v=1}^{C} q(S_u, S_v) = 0$. Thus, $q(S_u, S_u) \geq 0$ and S_u is indeed a community (cf. the proof of Theorem 18(ii)).

9.8 Proof of Lemma 33

Since $p_V(v) = p_W(v)$ for all v, it then follows from (93) that

$$\begin{aligned} Q(\mathcal{P}) &= \sum_{c=1}^{C} \sum_{v \in S_c} \sum_{w \in S_c} (p(v,w) - p_V(v) p_V(w)) && (213) \\ &= \sum_{c=1}^{C} \sum_{v \in S_c} \sum_{w \in S_c} (p(w,v) - p_V(w) p_V(v)). && (214) \end{aligned}$$

Adding (213) and (214) yields

$$2Q(\mathcal{P}) = \sum_{c=1}^{C} \sum_{v \in S_c} \sum_{w \in S_c} (p(v,w) + p(w,v) - 2 p_V(v) p_V(w)). \quad (215)$$

As $\tilde{p}(v,w) = (p(v,w) + p(w,v))/2$, we have

$$\tilde{p}_V(v) = \sum_{w \in V_g} \tilde{p}(v,w) = p_V(v).$$

Thus,

$$Q(\mathcal{P}) = \sum_{c=1}^{C} \sum_{v \in S_c} \sum_{w \in S_c} (\tilde{p}(v,w) - \tilde{p}_V(v) \tilde{p}_V(w)) = \tilde{Q}(\mathcal{P}).$$

9.9 Proof of Proposition 37

(i) Since the distance measure $d(\cdot,\cdot)$ is symmetric, we have from (160) that $\gamma(x,y) = \gamma(y,x)$. Thus, the cohesion measure between two points is symmetric.

(ii) We note from (161) that

$$\gamma(x,x) = \frac{1}{n^2} \sum_{z_2 \in \Omega} \sum_{z_1 \in \Omega} \Big(d(x,z_1) + d(z_2,x) - d(z_1,z_2) - d(x,x) \Big). \quad (216)$$

Since $d(x,x)=0$, we have from the triangular inequality that $\gamma(x,x)\geq 0$.

(iii) Note from (216) and (161) that

$$\begin{aligned}&\gamma(x,x)-\gamma(x,y)\\&=\frac{1}{n}\sum_{z_2\in\Omega}\Big(d(z_2,x)-d(x,x)+d(x,y)-d(z_2,y)\Big).\end{aligned}$$

Since $d(x,x)=0$, we have from the triangular inequality that

$$\gamma(x,x)\geq\gamma(x,y).$$

(iv) This can be easily verified by summing y in (160).

9.10 Proof of Theorem 39

We first show several properties that will be used in the proof of Theorem 39.

Proposition 59.

(i) Suppose that S_2 and S_3 are two disjoint subsets of Ω. Then

$$\begin{aligned}&\bar{d}(S_1,S_2\cup S_3)\\&=\frac{|S_2|}{|S_2|+|S_3|}\bar{d}(S_1,S_2)+\frac{|S_3|}{|S_2|+|S_3|}\bar{d}(S_1,S_3).\end{aligned} \tag{217}$$

(ii) The cohesion measure between two sets can be represented in terms of the average distance as follows:

$$\begin{aligned}\gamma(S_1,S_2) &= |S_1|\cdot|S_2|\cdot\Big(\bar{d}(\Omega,S_2)+\bar{d}(S_1,\Omega)\\&\quad-\bar{d}(\Omega,\Omega)-\bar{d}(S_1,S_2)\Big).\end{aligned} \tag{218}$$

(iii) Suppose that S is a nonempty set and it is not equal to Ω. Let $S^c=\Omega\backslash S$ be the set of points that are not in S. Then

$$\gamma(S,S)=-\gamma(S^c,S)=\gamma(S^c,S^c). \tag{219}$$

Proof. (i) This is trivial from the definition of the average distance in (164).

(ii) From (160), one can represent the cohesion measure between two points in terms of the average distance as follows:

$$\gamma(x,y)=\bar{d}(\Omega,y)+\bar{d}(x,\Omega)-\bar{d}(\Omega,\Omega)-d(x,y). \tag{220}$$

That (218) holds then follows from (162) and (220).

(iii) From (218), we know that $\gamma(\Omega,S)=\gamma(S\cup S^c,S)=0$. Thus, $\gamma(S,S)+\gamma(S^c,S)=0$. We then have

$$\gamma(S,S)=-\gamma(S^c,S). \tag{221}$$

From (221), we also have

$$\gamma(S^c,S^c)=-\gamma(S,S^c). \tag{222}$$

From the symmetric property of the cohesion measure, we have $\gamma(S^c,S)=\gamma(S,S^c)$. As a result of (221) and (222), we then have

$$\gamma(S,S)=\gamma(S^c,S^c).$$

■

Proof. (Theorem 39) (i) $\Rightarrow$ (ii): If $\gamma(S,S) \geq 0$, we then have from (219) that

$$\gamma(S^c, S^c) = \gamma(S,S) \geq 0. \tag{223}$$

(ii) $\Rightarrow$ (iii): If $\gamma(S^c, S^c) \geq 0$, then it follows from (219) that

$$\gamma(S^c, S) = -\gamma(S^c, S^c) \leq 0. \tag{224}$$

(iii) $\Rightarrow$ (iv): If $\gamma(S^c, S) \leq 0$, then we have from (219) that $\gamma(S,S) = -\gamma(S^c, S) \geq 0$. Thus, $\gamma(S,S) \geq \gamma(S, S^c)$.

(iv) $\Rightarrow$ (v): If $\gamma(S,S) \geq \gamma(S,S^c)$, then it follows from (224) that

$$\gamma(S,S) \geq \gamma(S,S^c) = -\gamma(S,S).$$

This then leads to $\gamma(S,S) \geq 0$. From (218), we know that

$$\gamma(S,S) = |S|^2 \cdot \Big(2\bar{d}(S,\Omega) - \bar{d}(\Omega,\Omega) - \bar{d}(S,S)\Big) \geq 0. \tag{225}$$

Thus, $2\bar{d}(S,\Omega) - \bar{d}(\Omega,\Omega) - \bar{d}(S,S) \geq 0$.

(v) $\Rightarrow$ (vi): Observe from (217) that

$$\bar{d}(S,\Omega) = \bar{d}(S, S \cup S^c) = \frac{|S|}{n}\bar{d}(S,S) + \frac{|S^c|}{n}\bar{d}(S,S^c),$$

and that

$$\bar{d}(\Omega,\Omega) = \bar{d}(S \cup S^c, S \cup S^c) = \frac{|S|^2}{n^2}\bar{d}(S,S) + \frac{2|S||S^c|}{n^2}\bar{d}(S,S^c) + \frac{|S^c|^2}{n^2}\bar{d}(S^c,S^c).$$

Thus,

$$2\bar{d}(S,\Omega) - \bar{d}(\Omega,\Omega) - \bar{d}(S,S) = (\frac{(|S^c|)}{n})^2\Big(2\bar{d}(S,S^c) - \bar{d}(S,S) - \bar{d}(S^c,S^c)\Big). \tag{226}$$

Clearly, if $2\bar{d}(S,\Omega) - \bar{d}(\Omega,\Omega) - \bar{d}(S,S) \geq 0$, then $2\bar{d}(S,S^c) - \bar{d}(S,S) - \bar{d}(S^c,S^c) \geq 0$.

(vi) $\Rightarrow$ (i): From (225) and (226), we know that $\gamma(S,S) \geq 0$ if $2\bar{d}(S,S^c) - \bar{d}(S,S) - \bar{d}(S^c,S^c) \geq 0$. ■

9.11 Proof of Lemma 42

(i) From the triangular inequality, it is easy to see from the definition of the triangular distance in (167) that $\Delta(x,S) \geq 0$. Note that

$$\frac{1}{|S|^2}\sum_{z_1 \in S}\sum_{z_2 \in S} d(x,z_1) = \frac{1}{|S|}\sum_{z_1 \in S} d(x,z_1) = \bar{d}(\{x\},S).$$

Similarly,

$$\frac{1}{|S|^2}\sum_{z_1 \in S}\sum_{z_2 \in S} d(x,z_2) = \bar{d}(\{x\},S).$$

Thus, the triangular distance in (167) can also be written as $\Delta(x,S) = 2\bar{d}(\{x\},S) - \bar{d}(S,S)$.

(ii) Recall from (218) that

$$\begin{aligned}\gamma(S_1,S_2) &= |S_1| \cdot |S_2| \cdot \Big(\bar{d}(\Omega,S_2) + \bar{d}(S_1,\Omega) \\ &\qquad -\bar{d}(\Omega,\Omega) - \bar{d}(S_1,S_2)\Big).\end{aligned}$$

Thus,

$$\begin{aligned}
&\gamma(x,x) - \frac{2}{|S|}\gamma(\{x\},S) + \frac{1}{|S|^2}\gamma(S,S) \\
&= 2\bar{d}(\{x\},\Omega) - \bar{d}(\Omega,\Omega) - 2\Big(\bar{d}(\{x\},\Omega) + \bar{d}(S,\Omega) \\
&\qquad -\bar{d}(\Omega,\Omega) - \bar{d}(\{x\},S)\Big) \\
&\qquad +\Big(2\bar{d}(\Omega,S) - \bar{d}(\Omega,\Omega) - \bar{d}(S,S)\Big) \\
&= 2\bar{d}(\{x\},S) - \bar{d}(S,S) = \Delta(x,S),
\end{aligned}$$

where we use (i) of the lemma in the last equality.

(iii) Note from (ii) that

$$\begin{aligned}
&\sum_{k=1}^{K}\sum_{x\in S_k}\Delta(x,S_k) \\
&= \sum_{k=1}^{K}\sum_{x\in S_k}\Big(\gamma(x,x) - \frac{2}{|S_k|}\gamma(\{x\},S_k) \\
&\qquad + \frac{1}{|S_k|^2}\gamma(S_k,S_k)\Big) \\
&= \sum_{k=1}^{K}\sum_{x\in S_k}\gamma(x,x) - \sum_{k=1}^{K}\frac{1}{|S_k|}\gamma(S_k,S_k) \\
&= \sum_{x\in\Omega}\gamma(x,x) - Q_{\text{nor}}(\mathcal{P}). \qquad (227)
\end{aligned}$$

(iv) From (i) of this lemma, we have

$$\sum_{k=1}^{K}\sum_{x\in S_k}\Delta(x,S_k) = \sum_{k=1}^{K}\sum_{x\in S_k}(2\bar{d}(\{x\},S_k) - \bar{d}(S_k,S_k)).$$

Observe that

$$\sum_{x\in S_k}\bar{d}(\{x\},S_k) = |S_k|\bar{d}(S_k,S_k).$$

Thus,

$$\begin{aligned}
&\sum_{k=1}^{K}\sum_{x\in S_k}\Delta(x,S_k) = \sum_{k=1}^{K}|S_k|\cdot(2\bar{d}(S_k,S_k) - \bar{d}(S_k,S_k)) \\
&= \sum_{k=1}^{K}|S_k|\cdot\bar{d}(S_k,S_k) = \sum_{k=1}^{K}\sum_{x\in S_k}\bar{d}(\{x\},S_k) \\
&= \sum_{x\in\Omega}\bar{d}(x,S_{c(x)}).
\end{aligned}$$

9.12 Proof of Theorem 43

In this section, we prove Theorem 43. For this, we need to prove the following two inequalities.

Lemma 60. For any set S and any point x that is not in S,

$$\sum_{y\in S\cup\{x\}} \Delta(y,S\cup\{x\}) \le \sum_{y\in S\cup\{x\}} \Delta(y,S), \tag{228}$$

and

$$\sum_{y\in S} \Delta(y,S) \le \sum_{y\in S} \Delta(y,S\cup\{x\}). \tag{229}$$

Proof. We first show that for any set S and any point x that is not in S,

$$\bar{d}(S\cup\{x\},S\cup\{x\}) - 2\bar{d}(S\cup\{x\},S) + \bar{d}(S,S) \le 0. \tag{230}$$

From the symmetric property and the weighted average property in Proposition 59(i), we have

$$\begin{aligned}\bar{d}(S\cup\{x\},S\cup\{x\}) &= \frac{|S|^2}{(|S|+1)^2}\bar{d}(S,S)\\ &+\frac{2|S|}{(|S|+1)^2}\bar{d}(\{x\},S) + \frac{1}{(|S|+1)^2}\bar{d}(\{x\},\{x\}),\end{aligned}$$

and

$$\bar{d}(S\cup\{x\},S) = \frac{|S|}{|S|+1}\bar{d}(S,S) + \frac{1}{|S|+1}\bar{d}(\{x\},S).$$

Note that

$$\bar{d}(\{x\},\{x\}) = d(x,x) = 0.$$

Thus,

$$\begin{aligned}&\bar{d}(S\cup\{x\},S\cup\{x\}) - 2\bar{d}(S\cup\{x\},S) + \bar{d}(S,S)\\ &= \frac{1}{(|S|+1)^2}\Big(\bar{d}(S,S) - 2\bar{d}(\{x\},S)\Big) \le 0,\end{aligned}$$

where we use (168) in the last inequality.

Note from (168) that

$$\begin{aligned}&\sum_{y\in S_2} \Delta(y,S_1) = \sum_{y\in S_2} (2\bar{d}(\{y\},S_1) - \bar{d}(S_1,S_1))\\ &= |S_2|\cdot\Big(2\bar{d}(S_1,S_2) - \bar{d}(S_1,S_1)\Big).\end{aligned} \tag{231}$$

Using (231) yields

$$\begin{aligned}&\sum_{y\in S\cup\{x\}} \Delta(y,S\cup\{x\}) - \sum_{y\in S\cup\{x\}} \Delta(y,S)\\ &= |S\cup\{x\}|\cdot\Big(2\bar{d}(S\cup\{x\},S\cup\{x\})\\ &\qquad -\bar{d}(S\cup\{x\},S\cup\{x\})\Big)\\ &\qquad -|S\cup\{x\}|\cdot\Big(2\bar{d}(S,S\cup\{x\}) - \bar{d}(S,S)\Big)\\ &= |S\cup\{x\}|\cdot\Big(\bar{d}(S\cup\{x\},S\cup\{x\})\\ &\qquad -2\bar{d}(S,S\cup\{x\}) + \bar{d}(S,S)\Big).\end{aligned} \tag{232}$$

As a result of (230), we then have

$$\sum_{y \in S \cup \{x\}} \Delta(y, S \cup \{x\}) - \sum_{y \in S \cup \{x\}} \Delta(y, S) \le 0.$$

Similarly, using (231) and (230) yields

$$\begin{aligned}
&\sum_{y \in S} \Delta(y, S) - \sum_{y \in S} \Delta(y, S \cup \{x\}) \\
&= |S| \cdot \Big(2\bar{d}(S,S) - \bar{d}(S,S)\Big) \\
&\quad -|S| \cdot \Big(2\bar{d}(S \cup \{x\}, S) - \bar{d}(S \cup \{x\}, S \cup \{x\})\Big) \\
&= |S| \cdot \Big(\bar{d}(S \cup \{x\}, S \cup \{x\}) \\
&\quad -2\bar{d}(S \cup \{x\}, S) + \bar{d}(S,S)\Big) \\
&\le 0 \qquad (233)
\end{aligned}$$

■

Proof. (Theorem 43) (i) Let S_k (resp. S'_k), $k = 1, 2, \ldots, K$, be the partition before (resp. after) the change. Also let $c(x)$ be the index of the set to which x belongs. Suppose that $\Delta(x_i, S_{k^*}) < \Delta(x_i, S_{c(x_i)})$ and x_i is moved from $S_{c(x_i)}$ to S_{k^*} for some point x_i and some k^*. In this case, we have $S'_{k^*} = S_{k^*} \cup \{x_i\}$, $S'_{c(x_i)} = S'_{c(x_i)} \backslash \{x\}$ and $S'_k = S_k$ for all $k \neq c(x_i), k^*$. It then follows from $\Delta(x_i, S_{k^*}) < \Delta(x_i, S_{c(x_i)})$ that

$$\begin{aligned}
&\sum_{k=1}^{K} \sum_{x \in S_k} \Delta(x, S_k) \\
&= \sum_{k \neq c(x_i), k^*} \sum_{x \in S_k} \Delta(x, S_k) \\
&\quad + \sum_{x \in S_{c(x_i)}} \Delta(x, S_{c(x_i)}) + \sum_{x \in S_{k^*}} \Delta(x, S_{k^*}) \\
&= \sum_{k \neq c(x_i), k^*} \sum_{x \in S_k} \Delta(x, S_k) + \sum_{x \in S_{c(x_i)} \backslash \{x_i\}} \Delta(x, S_{c(x_i)}) \\
&\quad + \Delta(x_i, S_{c(x_i)}) + \sum_{x \in S_{k^*}} \Delta(x, S_{k^*}) \\
&> \sum_{k \neq c(x_i), k^*} \sum_{x \in S_k} \Delta(x, S_k) + \sum_{x \in S_{c(x_i)} \backslash \{x_i\}} \Delta(x, S_{c(x_i)}) \\
&\quad + \Delta(x_i, S_{k^*}) + \sum_{x \in S_{k^*}} \Delta(x, S_{k^*}) \\
&= \sum_{k \neq c(x_i), k^*} \sum_{x \in S'_k} \Delta(x, S'_k) + \sum_{x \in S_{c(x_i)} \backslash \{x_i\}} \Delta(x, S_{c(x_i)}) \\
&\quad + \sum_{x \in S_{k^*} \cup \{x_i\}} \Delta(x, S_{k^*}), \qquad (234)
\end{aligned}$$

where we use the fact that $S'_k = S_k$ for all $k \neq c(x_i), k^*$, in the last equality. From (228) and $S'_{k^*} = S_{k^*} \cup \{x_i\}$, we know that

$$\begin{aligned}\sum_{x \in S_{k^*} \cup \{x_i\}} \Delta(x, S_{k^*}) &\geq \sum_{x \in S_{k^*} \cup \{x_i\}} \Delta(x, S_{k^*} \cup \{x_i\}) \\ &= \sum_{x \in S'_{k^*}} \Delta(x, S'_{k^*}). \end{aligned} \tag{235}$$

Also, it follows from (229) and $S'_{c(x_i)} = S'_{c(x_i)} \backslash \{x_i\}$ that

$$\begin{aligned}\sum_{x \in S_{c(x_i)} \backslash \{x_i\}} \Delta(x, S_{c(x_i)}) &\geq \sum_{x \in S_{c(x_i)} \backslash \{x_i\}} \Delta(x, S_{c(x_i)} \backslash \{x_i\}) \\ &= \sum_{x \in S'_{c(x_i)}} \Delta(x, S'_{c(x_i)}). \end{aligned} \tag{236}$$

Using (235) and (236) in (234) yields

$$\begin{aligned}&\sum_{k=1}^{K} \sum_{x \in S_k} \Delta(x, S_k) \\ &> \sum_{k \neq c(x_i), k^*} \sum_{x \in S'_k} \Delta(x, S'_k) + \sum_{x \in S'_{k^*}} \Delta(x, S'_{k^*}) \\ &\quad + \sum_{x \in S'_{c(x_i)}} \Delta(x, S'_{c(x_i)}) \\ &= \sum_{k=1}^{K} \sum_{x \in S'_k} \Delta(x, S'_k). \end{aligned} \tag{237}$$

In view of (170) and (237), we then conclude that the normalized modularity is increasing when there is a change. Since there is only a finite number of partitions for Ω, the algorithm thus converges to a local optimum of the normalized modularity.

(ii) The algorithm converges when there are no further changes. As such, we know for any $x \in S_i$ and $j \neq i$, $\Delta(x, S_i) \leq \Delta(x, S_j)$. Summing up all the points $x \in S_i$ and using (168) yields

$$\begin{aligned}0 &\geq \sum_{x \in S_i} \Big(\Delta(x, S_i) - \Delta(x, S_j)\Big) \\ &= \sum_{x \in S_i} \Big(2\bar{d}(\{x\}, S_i) - \bar{d}(S_i, S_i)\Big) \\ &\quad - \sum_{x \in S_i} \Big(2\bar{d}(\{x\}, S_j) - \bar{d}(S_j, S_j)\Big) \\ &= |S_i| \cdot \Big(\bar{d}(S_i, S_i) - 2\bar{d}(S_i, S_j) + \bar{d}(S_j, S_j)\Big). \end{aligned} \tag{238}$$

When the two sets S_i and S_j are viewed in isolation (by removing the data points not in $S_i \cup S_j$ from Ω), we have $S_j = S_i^c$. Thus,

$$\bar{d}(S_i, S_i) - 2\bar{d}(S_i, S_i^c) + \bar{d}(S_i^c, S_i^c) \leq 0.$$

As a result of Theorem 39(viii), we conclude that S_i is a cluster when the two sets S_i and S_j are viewed in isolation. Also, Theorem 39(ii) implies that $S_j = S_i^c$ is also a cluster when the two sets S_i and S_j are viewed in isolation. ■

9.13 Proof of Lemma 45

We first show (C1). Since $d(x,y) = d(y,x)$ for all $x \neq y$, we have from (173) that $\beta(x,y) = \beta(y,x)$ for all $x \neq y$.

To verify (C2), note from (173) that

$$\begin{aligned}\sum_{y\in\Omega}\beta(x,y) &= \frac{1}{n}\sum_{y\in\Omega}\sum_{z_2\in\Omega} d(z_2,y) + \sum_{z_1\in\Omega} d(x,z_1)\\ &- \frac{1}{n}\sum_{z_2\in\Omega}\sum_{z_1\in\Omega} d(z_2,z_1) - \sum_{y\in\Omega} d(x,y)\\ &= 0. \end{aligned} \tag{239}$$

Now we show (C3). Note from (173) that

$$\begin{aligned}&\beta(x,x) + \beta(y,z) - \beta(x,z) - \beta(x,y)\\ &= -d(x,x) - d(y,z) + d(x,z) + d(x,y). \end{aligned} \tag{240}$$

Since $d(x,x) = 0$, it then follows from the triangular inequality for $d(\cdot,\cdot)$ that

$$\beta(x,x) + \beta(y,z) - \beta(x,z) - \beta(x,y) \geq 0.$$

9.14 Proof of Lemma 46

Clearly, $d(x,x) = 0$ from (174) and thus (D2) holds trivially. That (D3) holds follows from the symmetric property in (C1) of Definition 44.

To see (D1), choosing $z = y$ in (172) yields

$$0 \leq \beta(x,x) + \beta(y,y) - \beta(x,y) - \beta(x,y) = 2d(x,y).$$

For the triangular inequality in (D4), note from (174) and (172) in (C3) that

$$\begin{aligned}&d(x,z) + d(z,y) - d(x,y)\\ &= \frac{(\beta(x,x)+\beta(z,z))}{2} - \beta(x,z) + \frac{(\beta(z,z)+\beta(y,y))}{2}\\ &\qquad -\beta(z,y) - \frac{(\beta(x,x)+\beta(y,y))}{2} + \beta(x,y)\\ &= \beta(z,z) + \beta(x,y) - \beta(z,x) - \beta(z,y) \geq 0. \end{aligned}$$

9.15 Proof of Theorem 47

We first show that $d^{**}(x,y) = d(x,y)$ for a distance measure $d(\cdot,\cdot)$. Note from (175) and $d(x,x) = 0$ that

$$\begin{aligned}d^*(x,x) &= \frac{1}{n}\sum_{z_2\in\Omega} d(z_2,x) + \frac{1}{n}\sum_{z_1\in\Omega} d(x,z_1)\\ &\quad -\frac{1}{n^2}\sum_{z_2\in\Omega}\sum_{z_1\in\Omega} d(z_2,z_1). \end{aligned} \tag{241}$$

From the symmetric property of $d(\cdot,\cdot)$, it then follows that

$$d^*(x,x) = \frac{2}{n}\sum_{z_1\in\Omega} d(x,z_1) - \frac{1}{n^2}\sum_{z_2\in\Omega}\sum_{z_1\in\Omega} d(z_2,z_1). \tag{242}$$

Similarly,

$$d^*(y,y) = \frac{2}{n}\sum_{z_2\in\Omega} d(z_2,y) - \frac{1}{n^2}\sum_{z_2\in\Omega}\sum_{z_1\in\Omega} d(z_2,z_1). \tag{243}$$

Using (175), (242) and (243) in (176) yields

$$d^{**}(x,y) = (d^*(x,x) + d^*(y,y))/2 - d^*(x,y) = d(x,y). \tag{244}$$

Now we show that $\beta^{**}(x,y) = \beta(x,y)$ for a cohesion measure $\beta(\cdot,\cdot)$. Note from (176) that

$$\begin{aligned}&\beta^*(z_2,y) + \beta^*(x,z_1) - \beta^*(z_1,z_2) - \beta^*(x,y)\\ &= -\beta(z_2,y) - \beta(x,z_1) + \beta(z_1,z_2) + \beta(x,y).\end{aligned} \tag{245}$$

Also, we have from (175) that

$$\begin{aligned}&\beta^{**}(x,y)\\ &= \frac{1}{n}\sum_{z_2\in\Omega}\beta^*(z_2,y) + \frac{1}{n}\sum_{z_1\in\Omega}\beta^*(x,z_1)\\ &\quad -\frac{1}{n^2}\sum_{z_2\in\Omega}\sum_{z_1\in\Omega}\beta^*(z_2,z_1) - \beta^*(x,y)\\ &= \frac{1}{n^2}\sum_{z_2\in\Omega}\sum_{z_1\in\Omega}\Big(\beta^*(z_2,y) + \beta^*(x,z_1)\\ &\quad -\beta^*(z_1,z_2) - \beta^*(x,y)\Big).\end{aligned} \tag{246}$$

Using (245) in (246) yields

$$\begin{aligned}&\beta^{**}(x,y)\\ &= \frac{1}{n^2}\sum_{z_2\in\Omega}\sum_{z_1\in\Omega}\Big(\beta(x,y) + \beta(z_1,z_2)\\ &\quad -\beta(x,z_1) - \beta(z_2,y)\Big)\\ &= \beta(x,y) + \frac{1}{n^2}\sum_{z_2\in\Omega}\sum_{z_1\in\Omega}\beta(z_1,z_2) - \frac{1}{n}\sum_{z_1\in\Omega}\beta(x,z_1)\\ &\quad -\frac{1}{n}\sum_{z_1\in\Omega}\beta(z_2,y).\end{aligned} \tag{247}$$

Since $\beta(\cdot,\cdot)$ is a cohesion measure that satisfies (C1)–(C3), we have from (C1) and (C2) that the last three terms in (247) are all equal to 0. Thus, $\beta^{**}(x,y) = \beta(x,y)$.

9.16 Proof of Proposition 51

We first show (C1). Since $\beta_1(x,y) = \beta_0(x,y)$ for all $x \neq y$, we have from the symmetric property of $\beta_0(\cdot,\cdot)$ that $\beta_1(x,y) = \beta_1(y,x)$ for all $x \neq y$. In view of (188), we then also have $\beta(x,y) = \beta(y,x)$ for all $x \neq y$.

To verify (C2), note from (188) that

$$\begin{aligned}\sum_{y\in\Omega}\beta(x,y) &= \sum_{y\in\Omega}\beta_1(x,y) - \frac{1}{n}\sum_{y\in\Omega}\sum_{z_1\in\Omega}\beta_1(z_1,y)\\ &- \sum_{z_2\in\Omega}\beta_1(x,z_2) + \frac{1}{n}\sum_{z_1\in\Omega}\sum_{z_2\in\Omega}\beta_1(z_1,z_2)\\ &= 0.\end{aligned} \tag{248}$$

Now we show (C3). Note from (188) that

$$\begin{aligned}&\beta(x,x)+\beta(y,z)-\beta(x,z)-\beta(x,y)\\&=\beta_1(x,x)+\beta_1(y,z)-\beta_1(x,z)-\beta_1(x,y).\end{aligned} \tag{249}$$

It then follows from (187) that for all $x \neq y \neq z$ that

$$\beta(x,x)+\beta(y,z)-\beta(x,z)-\beta(x,y) \geq 0. \tag{250}$$

If either $x=y$ or $x=z$, we also have

$$\beta(x,x)+\beta(y,z)-\beta(x,z)-\beta(x,y) = 0.$$

Thus, it remains to show the case that $y=z$ and $x \neq y$. For this case, we need to show that

$$\beta(x,x)+\beta(y,y)-\beta(x,y)-\beta(x,y) \geq 0. \tag{251}$$

Note from (250) that

$$\beta(y,y)+\beta(x,z)-\beta(y,z)-\beta(y,x) \geq 0. \tag{252}$$

Summing the two inequalities in (250) and (252) and using the symmetric property of $\beta(\cdot,\cdot)$ yields the desired inequality in (251).

9.17 Proof of Theorem 56

(i) It suffices to show that if x is in a set S_1 with $|S_1| > 1$ and $\Delta_{\mathsf{a}}(x,S_2) < \Delta_{\mathsf{a}}(x,S_1)$, then move point x from S_1 to S_2 increases the value of the objective function. Let S_k (resp. S_k'), $k=1,2,\ldots,K$, be the partition before (resp. after) the change. Also let R (resp. R') be the value of the objective function before (resp. after) the change. Then

$$\begin{aligned}&R'-R\\&=\frac{\gamma(S_1\backslash\{x\},S_1\backslash\{x\})}{|S_1|-1}+\frac{\gamma(S_2\cup\{x\},S_2\cup\{x\})}{|S_2|+1}\\&-\frac{\gamma(S_1,S_1)}{|S_1|}-\frac{\gamma(S_2,S_2)}{|S_2|}.\end{aligned}$$

Since

$$\begin{aligned}&\gamma(S_2\cup\{x\},S_2\cup\{x\})\\&=\gamma(S_2,S_2)+2\gamma(x,S_2)+\gamma(x,x),\end{aligned}$$

we have from (193) and (196) that

$$\begin{aligned}&\frac{\gamma(S_2\cup\{x\},S_2\cup\{x\})}{|S_2|+1}-\frac{\gamma(S_2,S_2)}{|S_2|}\\&=\frac{2|S_2|\gamma(x,S_2)+|S_2|\gamma(x,x)-\gamma(S_2,S_2)}{|S_2|\cdot(|S_2|+1)}\\&=\gamma(x,x)-\frac{|S_2|}{|S_2|+1}\Delta(x,S_2)\\&=\gamma(x,x)-\Delta_{\mathsf{a}}(x,S_2).\end{aligned}$$

On the other hand, we note that

$$\gamma(S_1\backslash\{x\},S_1\backslash\{x\})=\gamma(S_1,S_1)-2\gamma(x,S_1)+\gamma(x,x). \tag{253}$$

Using (253), (193) and (196) yields

$$\begin{aligned}
&\frac{\gamma(S_1\backslash\{x\}, S_1\backslash\{x\})}{|S_1|-1} - \frac{\gamma(S_1,S_1)}{|S_1|} \\
&= \frac{-2|S_1|\gamma(x,S_1) + |S_1|\gamma(x,x) + \gamma(S_1,S_1)}{|S_1|\cdot(|S_1|-1)} \\
&= -\gamma(x,x) + \frac{|S_1|}{|S_1|-1}\Delta(x,S_1) \\
&= -\gamma(x,x) + \Delta_{\mathsf{a}}(x,S_1).
\end{aligned}$$

Thus,

$$R' - R = \Delta_{\mathsf{a}}(x,S_1) - \Delta_{\mathsf{a}}(x,S_2) > 0.$$

As the objective value is non-increasing after a change of the partition, there is no loop in the algorithm. Since the number of partitions is finite, the algorithm thus converges in a finite number of steps (iterations).

(ii) Let $d(\cdot,\cdot)$ be the induced semi-metric. In view of Theorem 39(vi), it suffices to show that for all $i \neq j$

$$2\bar{d}(S_i,S_j) - \bar{d}(S_i,S_i) - \bar{d}(S_j,S_j) \geq 0. \tag{254}$$

If the set S_i contains a single element x, then

$$\bar{d}(S_i,S_i) = d(x,x) = 0.$$

Thus, the inequality in (254) holds trivially if $|S_i| = |S_j| = 1$.

Now suppose that $\min(|S_i|,|S_j|) \geq 2$. Without loss of generality, we assume that $|S_i| \geq 2$. When the K-sets$^+$ algorithm converges, we know that for any $x \in S_i$,

$$\Delta_{\mathsf{a}}(x,S_i) \leq \Delta_{\mathsf{a}}(x,S_j).$$

Summing over $x \in S_i$ yields

$$\sum_{x\in S_i} \Delta_{\mathsf{a}}(x,S_i) \leq \sum_{x\in S_i} \Delta_{\mathsf{a}}(x,S_j). \tag{255}$$

Note from (196) that for any $x \in S_i$,

$$\Delta_{\mathsf{a}}(x,S_i) = \frac{|S_i|}{|S_i|-1}\Delta(x,S_i),$$

and

$$\Delta_{\mathsf{a}}(x,S_j) = \frac{|S_j|}{|S_j|+1}\Delta(x,S_j).$$

Thus, it follows from (255) that

$$\frac{|S_i|}{|S_i|-1}\sum_{x\in S_i}\Delta(x,S_i) \leq \frac{|S_j|}{|S_j|+1}\sum_{x\in S_i}\Delta(x,S_j). \tag{256}$$

Note from (194) that

$$\sum_{x\in S_i}\Delta(x,S_i) = |S_i|\bar{d}(S_i,S_i), \tag{257}$$

and that

$$\sum_{x\in S_i}\Delta(x,S_j) = |S_i|(2\bar{d}(S_i,S_j) - \bar{d}(S_j,S_j)). \tag{258}$$

Since $d(\cdot,\cdot)$ is the induced semi-metric, we know from Proposition 54 that

$$\sum_{x \in S_i} \Delta(x, S_i) \ge 0.$$

Using this in (256) yields

$$\sum_{x \in S_i} \Delta(x, S_j) \ge 0.$$

Thus, we have from (256) that

$$\begin{aligned}
&\sum_{x \in S_i} \Delta(x, S_i) \le \frac{|S_i|}{|S_i|-1} \sum_{x \in S_i} \Delta(x, S_i) \\
&\le \frac{|S_j|}{|S_j|+1} \sum_{x \in S_i} \Delta(x, S_j) \le \sum_{x \in S_i} \Delta(x, S_j).
\end{aligned} \tag{259}$$

That the inequality in (254) holds follows directly from (257), (258) and (259).

9.18 Proof of Lemma 57

Since $\gamma(\cdot,\cdot)$ is symmetric, clearly $\tilde{\gamma}(\cdot,\cdot)$ is also symmetric. Thus, (C1) is satisfied trivially. To see that (C2) is satisfied, observe from (197) that

$$\begin{aligned}
\sum_{y \in \Omega} \tilde{\gamma}(x, y) &= \gamma(x, \Omega) - \gamma(x, \Omega) - \frac{1}{n}\gamma(\Omega, \Omega) \\
&+ \frac{1}{n}\gamma(\Omega, \Omega) + \sigma - \sigma = 0.
\end{aligned} \tag{260}$$

To see that (C3) holds, we note that

$$\tilde{\gamma}(x, x) = \gamma(x, x) - \frac{2}{n}\gamma(x, \Omega) + \frac{1}{n^2}\gamma(\Omega, \Omega) + \frac{(n-1)}{n}\sigma, \tag{261}$$

and that

$$\tilde{\gamma}(y, y) = \gamma(y, y) - \frac{2}{n}\gamma(y, \Omega) + \frac{1}{n^2}\gamma(\Omega, \Omega) + \frac{(n-1)}{n}\sigma,$$

Thus, for $x \ne y$, we have from (198) that

$$\begin{aligned}
&\tilde{\gamma}(x, x) + \tilde{\gamma}(y, y) - 2\tilde{\gamma}(x, y) \\
&= \gamma(x, x) + \gamma(y, y) - 2\gamma(x, y) + 2\sigma \ge 0.
\end{aligned}$$

9.19 Proof of Lemma 58

Note from Definition 53 and Definition 55 that

$$\tilde{\Delta}(x, S) = \tilde{\gamma}(x, x) - \frac{2}{|S|}\tilde{\gamma}(x, S) + \frac{1}{|S|^2}\tilde{\gamma}(S, S), \tag{262}$$

and that

$$\tilde{\Delta}_{\mathsf{a}}(x, S) = \begin{cases} \frac{|S|}{|S|+1}\tilde{\Delta}(x, S), & \text{if } x \notin S, \\ \frac{|S|}{|S|-1}\tilde{\Delta}(x, S), & \text{if } x \in S \text{ and } |S| > 1, \\ -\infty, & \text{if } x \in S \text{ and } |S| = 1. \end{cases} \tag{263}$$

To show (199), we need to consider two cases: (i) $x \in S$ and (ii) $x \notin S$. For both cases, we have from (197) that

$$\tilde{\gamma}(x,x) = \gamma(x,x) - \frac{2}{n}\gamma(x,\Omega) + \frac{1}{n^2}\gamma(\Omega,\Omega) + \frac{(n-1)}{n}\sigma, \tag{264}$$

and

$$\begin{aligned} \tilde{\gamma}(S,S) &= \gamma(S,S) - \frac{|S|}{n}\gamma(S,\Omega) - \frac{|S|}{n}\gamma(S,\Omega) \\ &+ \frac{|S|^2}{n^2}\gamma(\Omega,\Omega) + \sigma|S| - \sigma\frac{|S|^2}{n}. \end{aligned} \tag{265}$$

Now we consider the first case that $x \in S$. In this case, note that for $x \in S$

$$\begin{aligned} \tilde{\gamma}(x,S) &= \gamma(x,S) - \frac{|S|}{n}\gamma(x,\Omega) - \frac{1}{n}\gamma(S,\Omega) \\ &+ \frac{|S|}{n^2}\gamma(\Omega,\Omega) + \sigma - \sigma\frac{|S|}{n}. \end{aligned} \tag{266}$$

Using (264), (266) and (265) in (262) yields

$$\begin{aligned} \tilde{\Delta}(x,S) &= \tilde{\gamma}(x,x) - \frac{2}{|S|}\tilde{\gamma}(x,S) + \frac{1}{|S|^2}\tilde{\gamma}(S,S) \\ &= \gamma(x,x) - \frac{2}{|S|}\gamma(x,S) + \frac{1}{|S|^2}\gamma(S,S) \\ &\quad +\sigma(1 - \frac{1}{|S|}) \\ &= \Delta(x,S) + \sigma(1 - \frac{1}{|S|}). \end{aligned} \tag{267}$$

From (263), it then follows that

$$\begin{aligned} &\tilde{\Delta}_{\mathsf{a}}(x,S) = \frac{|S|}{|S|-1}\tilde{\Delta}(x,S) \\ &= \frac{|S|}{|S|-1}\Big(\Delta(x,S) + \sigma(1 - \frac{1}{|S|})\Big) \\ &= \Delta_{\mathsf{a}}(x,S) + \sigma. \end{aligned} \tag{268}$$

Now we consider the second case that $x \notin S$. In this case, note that for $x \notin S$

$$\begin{aligned} \tilde{\gamma}(x,S) &= \gamma(x,S) - \frac{|S|}{n}\gamma(x,\Omega) - \frac{1}{n}\gamma(S,\Omega) \\ &+ \frac{|S|}{n^2}\gamma(\Omega,\Omega) - \sigma\frac{|S|}{n}. \end{aligned} \tag{269}$$

Using (264), (269) and (265) in (262) yields

$$\begin{aligned} \tilde{\Delta}(x,S) &= \tilde{\gamma}(x,x) - \frac{2}{|S|}\tilde{\gamma}(x,S) + \frac{1}{|S|^2}\tilde{\gamma}(S,S) \\ &= \gamma(x,x) - \frac{2}{|S|}\gamma(x,S) + \frac{1}{|S|^2}\gamma(S,S) \\ &\quad +\sigma(1 + \frac{1}{|S|}) \\ &= \Delta(x,S) + \sigma(1 + \frac{1}{|S|}). \end{aligned} \tag{270}$$

From (263), it then follows that

$$\begin{aligned}\tilde{\Delta}_{\mathsf{a}}(x,S) &= \frac{|S|}{|S|+1}\tilde{\Delta}(x,S) \\ &= \frac{|S|}{|S|+1}\Big(\Delta(x,S)+\sigma(1+\frac{1}{|S|})\Big) \\ &= \Delta_{\mathsf{a}}(x,S)+\sigma. \end{aligned} \tag{271}$$

REFERENCES

[1] U. Brandes, G. Robins, A. Mccranie, and S. Wasserman, "What is network science?" *Network Science*, vol. 1, no. 1, 2013.
[2] L. C. Freeman, "A set of measures of centrality based on betweenness," *Sociometry*, pp. 35–41, 1977.
[3] ——, "Centrality in social networks conceptual clarification," *Social networks*, vol. 1, no. 3, pp. 215–239, 1979.
[4] M. Newman, *Networks: an introduction*. OUP Oxford, 2009.
[5] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," *Computer networks and ISDN systems*, vol. 30, no. 1, pp. 107–117, 1998.
[6] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi, "Defining and identifying communities in networks," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 101, no. 9, pp. 2658–2663, 2004.
[7] Y. Hu, H. Chen, P. Zhang, M. Li, Z. Di, and Y. Fan, "Comparative definition of community and corresponding identifying algorithm," *Physical Review E*, vol. 78, no. 2, p. 026121, 2008.
[8] R. Andersen, F. Chung, and K. Lang, "Local graph partitioning using pagerank vectors," in *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*. IEEE, 2006, pp. 475–486.
[9] R. Andersen and K. J. Lang, "Communities from seed sets," in *Proceedings of the 15th international conference on World Wide Web*. ACM, 2006, pp. 223–232.
[10] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney, "Statistical properties of community structure in large social and information networks," in *Proceedings of the 17th international conference on World Wide Web*. ACM, 2008, pp. 695–704.
[11] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3, pp. 75–174, 2010.
[12] D. J. Watts and S. H. Strogatz, "Collective dynamics of small-worldnetworks," *Nature*, vol. 393, no. 6684, pp. 440–442, 1998.
[13] 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, pp. 814–818, 2005.
[14] M. E. Newman, "Fast algorithm for detecting community structure in networks," *Physical review E*, vol. 69, no. 6, p. 066133, 2004.
[15] P. J. Mucha, T. Richardson, K. Macon, M. A. Porter, and J.-P. Onnela, "Community structure in time-dependent, multiscale, and multiplex networks," *Science*, vol. 328, no. 5980, pp. 876–878, 2010.
[16] B. Karrer, E. Levina, and M. E. Newman, "Robustness of community structure in networks," *Physical Review E*, vol. 77, no. 4, p. 046119, 2008.
[17] J. Leskovec, K. J. Lang, and M. Mahoney, "Empirical comparison of algorithms for network community detection," in *Proceedings of the 19th international conference on World wide web*. ACM, 2010, pp. 631–640.
[18] J. Yang and J. Leskovec, "Defining and evaluating network communities based on ground-truth," in *Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics*. ACM, 2012, p. 3.
[19] M. Rosvall and C. T. Bergstrom, "An information-theoretic framework for resolving community structure in complex networks," *Proceedings of the National Academy of Sciences*, vol. 104, no. 18, pp. 7327–7331, 2007.
[20] ——, "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.
[21] R. Lambiotte, "Multi-scale modularity in complex networks," in *Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt), 2010 Proceedings of the 8th International Symposium on*. IEEE, 2010, pp. 546–553.
[22] J.-C. Delvenne, S. N. Yaliraki, and M. Barahona, "Stability of graph communities across time scales," *Proceedings of the National Academy of Sciences*, vol. 107, no. 29, pp. 12 755–12 760, 2010.
[23] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Gossip algorithms: Design, analysis and applications," in *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, vol. 3. IEEE, 2005, pp. 1653–1664.
[24] M. A. Porter, J.-P. Onnela, and P. J. Mucha, "Communities in networks," *Notices of the AMS*, vol. 56, no. 9, pp. 1082–1097, 2009.
[25] M. E. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical review E*, vol. 69, no. 2, p. 026113, 2004.
[26] F. Wu and B. A. Huberman, "Finding communities in linear time: a physics approach," *The European Physical Journal B-Condensed Matter and Complex Systems*, vol. 38, no. 2, pp. 331–338, 2004.
[27] J. Duch and A. Arenas, "Community detection in complex networks using extremal optimization," *Physical review E*, vol. 72, no. 2, p. 027104, 2005.
[28] U. N. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," *Physical Review E*, vol. 76, no. 3, p. 036106, 2007.
[29] A. Clauset, M. E. Newman, and C. Moore, "Finding community structure in very large networks," *Physical review E*, vol. 70, no. 6, p. 066111, 2004.

[30] A. Lancichinetti, S. Fortunato, and J. Kertész, "Detecting the overlapping and hierarchical community structure in complex networks," *New Journal of Physics*, vol. 11, no. 3, p. 033015, 2009.

[31] K. Kamvar, S. Sepandar, K. Klein, D. Dan, M. Manning, and C. Christopher, "Spectral learning," in *International Joint Conference of Artificial Intelligence*. Stanford InfoLab, 2003.

[32] I. S. Dhillon, Y. Guan, and B. Kulis, "Kernel k-means: spectral clustering and normalized cuts," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2004, pp. 551–556.

[33] B. Kulis, S. Basu, I. Dhillon, and R. Mooney, "Semi-supervised graph clustering: a kernel approach," *Machine Learning*, vol. 74, no. 1, pp. 1–22, 2009.

[34] B. Long, Z. M. Zhang, and P. S. Yu, "A probabilistic framework for relational clustering," in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2007, pp. 470–479.

[35] B. Karrer and M. E. Newman, "Stochastic block models and community structure in networks," *Physical Review E*, vol. 83, no. 1, p. 016107, 2011.

[36] L. Danon, A. Diaz-Guilera, J. Duch, and A. Arenas, "Comparing community structure identification," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2005, no. 09, p. P09008, 2005.

[37] A. Lancichinetti and S. Fortunato, "Community detection algorithms: A comparative analysis," *Physical review E*, vol. 80, no. 5, p. 056117, 2009.

[38] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*. Elsevier Academic press, USA, 2006.

[39] A. Rajaraman, J. Leskovec, and J. D. Ullman, *Mining of massive datasets*. Cambridge University Press, 2012.

[40] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM computing surveys (CSUR)*, vol. 31, no. 3, pp. 264–323, 1999.

[41] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651–666, 2010.

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

[43] M. Van der Laan, K. Pollard, and J. Bryan, "A new partitioning around medoids algorithm," *Journal of Statistical Computation and Simulation*, vol. 73, no. 8, pp. 575–584, 2003.

[44] H.-S. Park and C.-H. Jun, "A simple and fast algorithm for k-medoids clustering," *Expert Systems with Applications*, vol. 36, no. 2, pp. 3336–3341, 2009.

[45] S. Har-Peled and B. Sadri, "How fast is the k-means method?" *Algorithmica*, vol. 41, no. 3, pp. 185–202, 2005.

[46] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035.

[47] B. Bahmani, B. Moseley, A. Vattani, R. Kumar, and S. Vassilvitskii, "Scalable k-means++," *Proceedings of the VLDB Endowment*, vol. 5, no. 7, pp. 622–633, 2012.

[48] R. Jaiswal and N. Garg, "Analysis of k-means++ for separable data," in *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. Springer, 2012, pp. 591–602.

[49] M. Agarwal, R. Jaiswal, and A. Pal, "k-means++ under approximation stability," in *Theory and Applications of Models of Computation*. Springer, 2013, pp. 84–95.

[50] J. Shi and J. Malik, "Normalized cuts and image segmentation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 8, pp. 888–905, 2000.

[51] A. Ben-Hur, D. Horn, H. T. Siegelmann, and V. Vapnik, "Support vector clustering," *The Journal of Machine Learning Research*, vol. 2, pp. 125–137, 2002.

[52] C. Ding and X. He, "K-means clustering via principal component analysis," in *Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004, p. 29.

[53] F. Camastra and A. Verri, "A novel kernel method for clustering," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, no. 5, pp. 801–805, 2005.

[54] W.-Y. Chen, Y. Song, H. Bai, C.-J. Lin, and E. Y. Chang, "Parallel spectral clustering in distributed systems," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 33, no. 3, pp. 568–586, 2011.

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

[56] M. Filippone, F. Camastra, F. Masulli, and S. Rovetta, "A survey of kernel and spectral methods for clustering," *Pattern recognition*, vol. 41, no. 1, pp. 176–190, 2008.

[57] M.-F. Balcan, A. Blum, and A. Gupta, "Clustering under approximation stability," *Journal of the ACM (JACM)*, vol. 60, no. 2, p. 8, 2013.

[58] C.-S. Chang, C.-Y. Hsu, J. Cheng, and D.-S. Lee, "A general probabilistic framework for detecting community structure in networks," in *IEEE INFOCOM '11*, April 2011.

[59] C.-S. Chang, C.-J. Chang, W.-T. Hsieh, D.-S. Lee, L.-H. Liou, and W. Liao, "Relative centrality and local community detection," *Network Science*, vol. 3, no. 4, pp. 445–479, 2015.

[60] D. Liben-Nowell and J. Kleinberg, "The link prediction problem for social networks," in *Proceedings of the twelfth international conference on Information and knowledge management*. ACM, 2003, pp. 556–559.

[61] L. Katz, "A new status index derived from sociometric analysis," *Psychometrika*, vol. 18, no. 1, pp. 39–43, 1953.

[62] 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.

[63] A. Lancichinetti and S. Fortunato, "Limits of modularity maximization in community detection," *Physical Review E*, vol. 84, no. 6, p. 066122, 2011.

[64] C. Granell, S. Gomez, and A. Arenas, "Hierarchical multiresolution method to overcome the resolution limit in complex networks," *International Journal of Bifurcation and Chaos*, vol. 22, no. 07, 2012.

[65] U. Brandes, D. Delling, M. Gaertler, R. Görke, M. Hoefer, Z. Nikoloski, and D. Wagner, "Maximizing modularity is hard," *arXiv preprint physics/0608255*, 2006.

[66] M. E. Newman, "Finding community structure in networks using the eigenvectors of matrices," *Physical review E*, vol. 74, no. 3, p. 036104, 2006.

[67] C.-S. Chang, W. Liao, Y.-S. Chen, and L.-H. Liou, "A mathematical theory for clustering in metric spaces," *IEEE Transactions on Network Science and Engineering*, vol. 3, no. 1, pp. 2–16, 2016.

[68] 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, p. P10008, 2008.

[69] F. D. Malliaros and M. Vazirgiannis, "Clustering and community detection in directed networks: A survey," *Physics Reports*, vol. 533, no. 4, pp. 95–142, 2013.

[70] C.-S. Chang, D.-S. Lee, L.-H. Liou, S.-M. Lu, and M.-H. Wu, "A probabilistic framework for structural analysis in directed networks," in *ICC*, May 2016.

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

[72] R. Lambiotte and M. Rosvall, "Ranking and clustering of nodes in networks with smart teleportation," *Physical Review E*, vol. 85, no. 5, p. 056107, 2012.

[73] T. H. Haveliwala, "Topic-sensitive pagerank," in *Proceedings of the 11th international conference on World Wide Web*. ACM, 2002, pp. 517–526.

[74] G. Jeh and J. Widom, "Scaling personalized web search," in *Proceedings of the 12th international conference on World Wide Web*. ACM, 2003, pp. 271–279.

[75] S. Juneja and P. Shahabuddin, "Rare-event simulation techniques: an introduction and recent advances," *Handbooks in operations research and management science*, vol. 13, pp. 291–350, 2006.

[76] T. M. Cover and J. A. Thomas, *Elements of information theory*. John Wiley & Sons, 2012.

[77] C.-S. Chang, *Performance guarantees in communication networks*. Springer Science & Business Media, 2012.

[78] C.-S. Chang, C.-T. Chang, D.-S. Lee, and L.-H. Liou, "K-sets+: a linear-time clustering algorithm for data points with a sparse similarity measure," *arXiv preprint arXiv:1705.04249*, 2017.

[79] R. Nelson, *Probability, stochastic processes, and queueing theory: the mathematics of computer performance modeling*. Springer Verlag, 1995.

[80] 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.

[81] J. Reichardt and S. Bornholdt, "Statistical mechanics of community detection," *Physical Review E*, vol. 74, no. 1, p. 016110, 2006.

[82] A. Arenas, A. Fernandez, and S. Gomez, "Analysis of the structure of complex networks at different resolution levels," *New Journal of Physics*, vol. 10, no. 5, p. 053039, 2008.

[83] W. W. Zachary, "An information flow model for conflict and fission in small groups," *Journal of anthropological research*, pp. 452–473, 1977.

[84] A. Clauset, M. E. Newman, and C. Moore, "Finding community structure in very large networks," *Physical review E*, vol. 70, no. 6, p. 066111, 2004.

[85] F. Chung, "Laplacians and the cheeger inequality for directed graphs," *Annals of Combinatorics*, vol. 9, no. 1, pp. 1–19, 2005.

[86] Y. Kim, S.-W. Son, and H. Jeong, "Finding communities in directed networks," *Physical Review E*, vol. 81, no. 1, p. 016103, 2010.

[87] M. Rosvall and C. T. Bergstrom, "Maps of information flow reveal community structure in complex networks," *Proceedings of the National Academy of Sciences USA*, vol. 105, pp. 1118–1123, 2008.

[88] R. Lambiotte, J.-C. Delvenne, and M. Barahona, "Random walks, markov processes and the multiscale modular organization of complex networks," *IEEE Transactions on Network Science and Engineering*, vol. 1, no. 2, pp. 76–90, 2014.

[89] E. A. Leicht and M. E. Newman, "Community structure in directed networks," *Physical review letters*, vol. 100, no. 11, p. 118703, 2008.

[90] P. Erdös and A. Rényi, "On random graphs," *Publ. Math. Debrecen*, vol. 6, pp. 290–297, 1959.

[91] A. Saade, F. Krzakala, and L. Zdeborová, "Spectral clustering of graphs with the bethe hessian," in *Advances in Neural Information Processing Systems*, 2014, pp. 406–414.

[92] L. Z. Aurelien Decelle, Florent Krzakala and P. Zhang. (2012) Mode-net: Modules detection in networks. [Online]. Available: http://mode_net.krzakala.org/

[93] L. Massoulié, "Community detection thresholds and the weak ramanujan property," in *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*. ACM, 2014, pp. 694–703.

[94] E. Mossel, J. Neeman, and A. Sly, "A proof of the block model threshold conjecture," *arXiv preprint arXiv:1311.4115*, 2013.

[95] A. Lancichinetti, S. Fortunato, and F. Radicchi, "Benchmark graphs for testing community detection algorithms," *Physical review E*, vol. 78, no. 4, p. 046110, 2008.

[96] G. Csardi and T. Nepusz, "The igraph software package for complex network research," *InterJournal*, vol. Complex Systems, p. 1695, 2006. [Online]. Available: http://igraph.org

[97] S. Harenberg, G. Bello, L. Gjeltema, S. Ranshous, J. Harlalka, R. Seay, K. Padmanabhan, and N. Samatova, "Community detection in large-scale networks: a survey and empirical evaluation," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 6, no. 6, pp. 426–439, 2014.

[98] J. Jung, W. Jin, L. Sael, and U. Kang, "Personalized ranking in signed networks using signed random walk with restart," in *Data Mining (ICDM), 2016 IEEE 16th International Conference on*. IEEE, 2016, pp. 973–978.

[99] J. Weng, E.-P. Lim, J. Jiang, and Q. He, "Twitterrank: finding topic-sensitive influential twitterers," in *Proceedings of the third ACM international conference on Web search and data mining*. ACM, 2010, pp. 261–270.

[100] L. A. Adamic and N. Glance, "The political blogosphere and the 2004 us election: divided they blog," in *Proceedings of the 3rd international workshop on Link discovery*. ACM, 2005, pp. 36–43.

[101] J. Leskovec, L. Backstrom, and J. Kleinberg, “Memetracker data,” 2008.

[102] S. Osiński and D. Weiss, “Carrot2: Design of a flexible and efficient web information retrieval framework,” in *International atlantic web intelligence conference*. Springer, 2005, pp. 439–444.

[103] C.-H. Chang and C.-S. Chang, “Exponentially twisted sampling: a unified approach for centrality analysis in attributed networks,” *arXiv preprint arXiv:1708.00379*, 2017.

[104] S. Lloyd, “Least squares quantization in pcm,” *Information Theory, IEEE Transactions on*, vol. 28, no. 2, pp. 129–137, 1982.

[105] U. Von Luxburg and S. Ben-David, “Towards a statistical theory of clustering,” in *Pascal workshop on statistics and optimization of clustering*. Citeseer, 2005.

[106] N. Jardine and R. Sibson, “Mathematical taxonomy,” *London etc.: John Wiley*, 1971.

[107] W. E. Wright, “A formalization of cluster analysis,” *Pattern Recognition*, vol. 5, no. 3, pp. 273–282, 1973.

[108] J. Puzicha, T. Hofmann, and J. M. Buhmann, “A theory of proximity based clustering: Structure detection by optimization,” *Pattern Recognition*, vol. 33, no. 4, pp. 617–634, 2000.

[109] J. Kleinberg, “An impossibility theorem for clustering,” *Advances in neural information processing systems*, pp. 463–470, 2003.

[110] R. B. Zadeh and S. Ben-David, “A uniqueness theorem for clustering,” in *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*. AUAI Press, 2009, pp. 639–646.

[111] G. Carlsson, F. Mémoli, A. Ribeiro, and S. Segarra, “Hierarchical quasi-clustering methods for asymmetric networks,” *arXiv preprint arXiv:1404.4655*, 2014.

[112] S. Ben-David and M. Ackerman, “Measures of clustering quality: A working set of axioms for clustering,” in *Advances in neural information processing systems*, 2009, pp. 121–128.

[113] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise.” in *Kdd*, vol. 96, 1996, pp. 226–231.

[114] A. Cuevas, M. Febrero, and R. Fraiman, “Cluster analysis: a further approach based on density estimation,” *Computational Statistics & Data Analysis*, vol. 36, no. 4, pp. 441–459, 2001.

[115] M. Halkidi and M. Vazirgiannis, “A density-based cluster validity approach using multi-representatives,” *Pattern Recognition Letters*, vol. 29, no. 6, pp. 773–786, 2008.

[116] S. X. Yu and J. Shi, “Multiclass spectral clustering,” in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*. IEEE, 2003, pp. 313–319.

[117] I. Dhillon, Y. Guan, and B. Kulis, *A unified view of kernel k-means, spectral clustering and graph cuts*. Computer Science Department, University of Texas at Austin, 2004.

[118] A. Y. Ng, M. I. Jordan, Y. Weiss *et al.*, “On spectral clustering: Analysis and an algorithm,” *Advances in neural information processing systems*, vol. 2, pp. 849–856, 2002.

[119] I. S. Dhillon, Y. Guan, and B. Kulis, “Weighted graph cuts without eigenvectors a multilevel approach,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 29, no. 11, pp. 1944–1957, 2007.

[120] R. Campigotto, P. C. Céspedes, and J.-L. Guillaume, “A generalized and adaptive method for community detection,” *arXiv preprint arXiv:1406.2518*, 2014.

[121] “The wondernetwork dataset,” https://wondernetwork.com/.

[122] “The submarine cable map,” http://www.submarinecablemap.com/.