MINISTRY OF EDUCATION AND TRAINING

VIETNAM ACADEMY OF SCIENCE AND TECHNOLOGY

GRADUATE UNIVERSITY OF SCIENCE AND TECHNOLOGY

Hoang Duc Anh

MODULARITY AND RANDOM WALKS IN COMMUNITY DETECTION

Major : Applied Mathematics
Code: 8 46 01 12

MASTER'S THESIS IN MATHEMATICS

ADVISOR:
Assoc. Prof. Dr. Sc. Phan Thi Ha Duong

Hanoi – 2022

Declaration

I hereby declare that this thesis and the work presented in it are the result of my own study. Whenever the works of others are involved, every effort has been made to give credit clearly, with due references to the literature. I confirm that this thesis has not been previously included in a thesis or dissertation submitted for a degree or any other qualification at this graduate university or any other institution. I take full responsibility for the above declaration.

Student

Hoang Duc Anh

Acknowledgements

I would like to express a deep gratitude to my advisor, Assoc. Prof. Dr. Sc. Phan Thi Ha Duong, who introduced me to network science and has provided me with support and guidance throughout my study. Her encouragement and enthusiasm for research have been a constant source of inspiration for me throughout the project.

I would like to thank the researchers at the Institute of Mathematics and the group Mathematical Foundation for Computer Science for having created a wonderful environment for young students like me. I also thank the lecturers and administrative staff at the Institute as well as the Graduate University of Science and Technology for their valuable lessons and dedicated help during my degree.

I would like to acknowledge the generous support of Vingroup JSC, who has funded my study for the last two years. I was supported by the Master, PhD Scholarship Programme of Vingroup Innovation Foundation (VINIF), Institute of Big Data, codes VINIF.2020.ThS.VTH.02 and VINIF.2021.ThS.VTH.08.

Contents

List of Figures

Introduction

Due to the rise of Big Data phenomenon and interdisciplinary research, network science emerged and has drawn enormous interest from both academia and industry. Dividing a network into smaller groups of similar nodes - a task called community detection - is one direction that has yielded valuable insights about complex network data. In this master's thesis, we study two topics in the field of community detection: a quality function called modularity, and clustering properties of the random walk eigenvectors of a graph.

This thesis contains four chapters and one appendix. The main content is in Chapter 2 and Chapter 3.

- Chapter 1 briefly discusses some notable features of network science and community structure in order to situate the main topics of the thesis.
- Chapter 2 is a detailed exposition of modularity - a popular clustering quality function. Section 2.1 defines modularity and gives the standard interpretation based on a random graph model. Section 2.2 presents basic properties of modularity, including modularity of some special graphs (cycles, complete multipartite graphs, ...). Section 2.3 explains several shortcomings of modularity when used in the practical context of community detection.
- Chapter 3 studies the spectral properties of the random walk matrix and a clustering algorithm based on those properties. Section 3.1 introduces the random walk matrix and its spectrum. Section 3.2 explains why the top eigenvectors of that matrix inherit the clustering structure of the graph and illustrates the phenomenon visually. Section 3.3 presents the Walktrap algorithm and performs experiments on some random graphs to investigate the effect of step size and linkage method in the algorithm.
- Chapter 4 summarizes the main content of the thesis and introduces some further directions.
- Appendix A provides a simple Python implementation of the Walktrap algorithm

introduced in Chapter 3.

This is an expository thesis. Our main contribution lies in collecting and organizing several results scattered in the literature; we try to provide more detail in theoretical explanations and proofs, and illustrate various ideas using our own experiments implemented in the Python programming language (more detail can be found in Chapter 4). We hope this document could be a useful starting point for people studying the two main topics mentioned above.

Notations and conventions

In this thesis, ‘graph’ and ‘network’ are used interchangeably.

Unless stated otherwise, we work with simple undirected graphs, i.e. undirected graphs with no parallel edges and no self-loops. For a graph G, let $V(G)$ and $E(G)$ be the vertex set and edge set of G; sometimes we simply use V and E if the underlying graph G is clear from context. For a vertex subset $P \subseteq V(G)$, let $E(P)$ be the set of edges lying inside P and let $e(P) := |E(P)|$. We also define the *volume* of P to be the sum of the degrees of the vertices inside P:

$$\mathrm{vol}(P) := \sum_{v \in P} \deg(v).$$

In case there are many graphs under consideration, we put G in the subscripts, like $e_G(P)$, $\mathrm{vol}_G(P)$, ...

A partition $\mathcal{P} = \{P_1, \ldots, P_k\}$ of a set V is a collection of disjoint non-empty subsets whose union is V, that is $P_i \cap P_j = \emptyset$ for all $i \neq j$ and $\sqcup_{i=1}^{k} P_i = V$.

All vectors are column vectors. The transpose of matrix M is denoted by $M^\top$, and similarly the transpose of vector x is $x^\top$ (which is a row vector). We use $\mathbf{1}$ to denote a vector with all entries equal to 1, whose dimension should be clear from context.

In many places we use subscripts to index vectors, so round brackets are used for vector entries: $x_i(u)$ is the u-th entry of vector x_i.

Chapter 1

NETWORKS AND COMMUNITIES

This short chapter introduces some notable features of network science and community structure in order to set the background for the main topics of the thesis.

1.1 On network science

Network science has grown to an enormous discipline, and it is certainly outside of this chapter's scope to even attempt a small survey. Instead, we only explain a few features that can be confusing for beginners. There are currently several good textbooks on network science; among them, we mention [1] with a broad coverage, and [2] with a unique focus on modeling, interpretation, and data quality.

One attempt at defining network science can be found in the editorial [3]: *network science is the study of network models.* A *network model* is a network representation of something, comprising two main components: *abstraction* from real phenomena to network concepts, and *representation* of those concepts by network data. What distinguishes network data from traditional tabular data is that there is some *dependency* (or *relationship*) built in, most easily visualized as links (or edges) in a graph. Whether a relationship should be represented by a network, and then how it can be represented, depend a lot on the problem being studied; see Chapters 5 and 11 of [2] for more detailed introduction.

There are several reasons, both commercial and scientific, for the increased interest in network science in recent decades. A popular reason, which is also the one most easily capturing the public imagination, is the rise of the Internet and big social media networks, whose links are given concrete names like 'tag', 'friend', 'follower', ... Another big spur to the study of networks is how they can be used to tackle *complexity* in various scientific disciplines. This approach introduced a new paradigm in science, called *topological explanations* by philosophers [4], complementing existing kinds of explanations like mechanistic, causal, probabilistic, ... See the surveys [5, 6] for more details on how networks can be used to model complexity.

One notable feature of network science is how scattered the literature is (as can be shown by a brief look at the bibliography of this thesis). Outside from a few recent network-specific journals, network science articles appear in journals and conferences of physics, computer science, mathematics, statistics, as well as sociology. Inevitably, there are different cultures and methods. The traditional divide is between social scientists coming from social network analysis, and natural scientists coming from physics. Social scientists study small, carefully curated networks in very specific contexts. They have very rich notions of links and care about the motivation of actors in the networks. In contrast, physicists are inspired by statistical physics and complexity, hence they search for 'universal laws' in large collections of large networks, abstracted from those networks' context. This divide is discussed in [7, 8] [2, Chapter 2]. A slightly different but related contrast is between those searching for universality independent of particular objects, and statisticians who focus on testable properties in real data. The division leads to the controversy of power-law degree distribution, carefully recounted in [9]. Finally, there are also computer scientists and mathematicians, each with their own approaches [10]. All of this make network science a 'trading zone' [9], where cross-fertilization of ideas as well as cultural clashes happen.

1.2 Community structure

Given a network, it is natural to find groups of similar nodes, and we say those groups form a community structure. That description is certainly vague, because we do not (and probably should not) have precise conditions for when nodes form a community. The task of discerning those groups in a network is called *community detection* or *graph clustering*; those two terms are used interchangeably in this thesis.

Graph clustering is closely related to tabular data clustering. Indeed, one popular way of clustering tabular data is *spectral clustering*: we create a graph where nodes represent data points, connect two nodes if they are 'close' enough, then use spectral properties of the graph to cluster data (see the surveys mentioned in Section 3.2 of this thesis). Conversely, *graph embedding* is a method of handling very large graphs by embedding vertices in low dimensional euclidean spaces before applying standard techniques of tabular data (see [11] for a recent survey of this big field).

Defining communities

There is no single, unified concept of a community; see [12, III.B] and [13, II] for many definitions. Some define communities using numerical characteristics like edge density or a quality function. Other take a procedural approach and define communities as results of community detection algorithms; in other words, the algorithms become implicit models of communities. There are also the issues of whether communities can be overlapped, and difference between global (discovering all communities) and local (finding communities in a small region only) methods.

For the purposes of Chapter 2 and Chapter 3 in this thesis, a community is a group of vertices which has higher internal density than external density, and a community structure is a partition of the vertex set (in particular, we do not consider overlapping communities). Figure 1.1 shows a graph with two clear groups together with its adjacency matrix, generated using the stochastic block model. Graph drawing is computationally intensive and not particularly useful if the edge density is high, so we mostly use adjacency matrices to represent graphs.

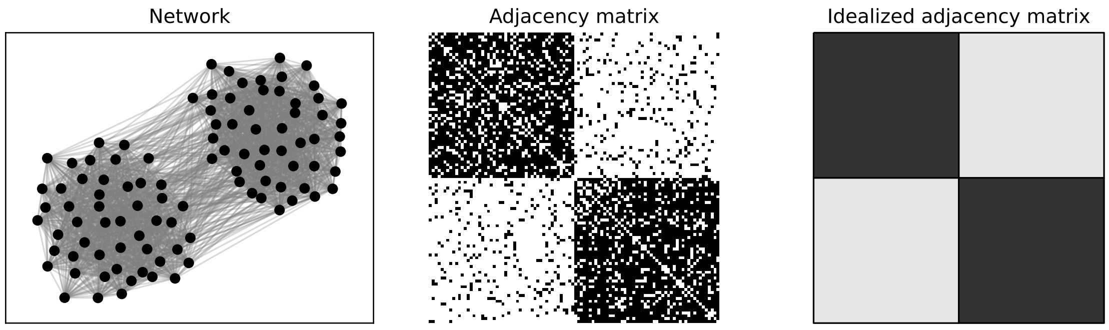


Figure 1.1: A graph with two communities and its representation by adjacency matrix

It may seem reasonable to define community using metadata on nodes as 'ground truth'. For example, we expect that the links (connections) in a social networks reveal some underlying social groups based on preferences, occupations, ... However, this kind of definition needs to be approached with care, probably requiring extensive domain knowledge. There are many different kinds of possible metadata, with no necessary relationship to the edges of the network. Data quality is also an issue, especially for networks mined from large databases [14]. Node metadata is best considered as additional data to be modeled together with the network [15, 16], or incorporated into the clustering algorithm [17]. See [18] for a more general survey considering information on both edges and nodes.

On a related note, clustering algorithms usually optimize (or at least favor higher values of) some objective functions, like quality metrics or likelihood functions. However, several empirical studies on real networks with metadata [19, 20] show that ground truth communities almost never give the best values for those objective functions. This means that our designed objectives can lead to overfit, or (more optimistically) the algorithms have found some hidden structure not revealed by given node data.

The goals

Following [21], we broadly identify three main reasons for finding community structures in a network:

- to find a coarse-level description of the network;
- to understand how dynamic and stochastic processes evolve on the network; and
- to reveal functional properties.

The goals can also be divided into two main groups: whether we analyze the network for descriptive purposes or inferential purposes [22]. For example, if we want to divide the network into small parts for efficient information processing, we can take a descriptive approach and analyze the network as is, using precise objective functions to quantify the results. On the other hand, sociologists trying to understand how social groups are formed need to take an inferential approach, accounting for uncertainty using statistical methods.

Choosing the most suitable approach (or approaches) requires evaluating many factors: computational resource, data quality, domain-specific goals ... Some surveys mentioned below can help in the process.

Community detection algorithms

There are currently many community detection methods available, as well as countless variants and improvements. We mention a few works that collect and compare a large number of methods.

Many surveys group methods according to their intrinsic theoretical/conceptual foundation. For example, Rosvall et al. [23] group community detection methods under four perspectives:

- the cut-based perspective, which aims to minimize the number of edges between nodes;
- the clustering perspective, which finds dense, coherent groups of nodes;
- the stochastic equivalence perspective, which infers groups using statistical models (like stochastic block models); and
- the dynamical perspective, which relies on how modular structure impacts evolution of processes on networks.

Various other classifications are available, see [24, 12, 13, 25].

On the other hand, some studies compare community detection algorithms by running them on a large number of networks and analyzing the results. Dao et al. [26] classify methods into five main groups: edge removal based, modularity optimization, dynamic process based, statistical inference based, and a final group of miscellaneous methods not belonging to the other four. Those methods are run on more than 100 networks from various domains, then compared based on running time, number of communities found and community sizes, quality of communities, and similarity between the partitions produced. Detailed results are given, which have implications for choosing a suitable method in practice. Other empirical studies, with many different approaches, include [27, 28, 20].

Exploring community structure

We survey some papers that study community structure in real networks. Some authors use communities defined by node metadata, while others use clustering returned by algorithms.

Leskovec et al. [29] study the structure of networks using *network community profile* plots, which are plots of best conductance with respect to the number of nodes in one side of a cut. Since computing minimum conductance is intractable, the authors use several approximating algorithms. They found that in very large networks, the profile plots have u-shape, with the best cuts falling around 100 - 150 nodes; this differs from small networks and random networks. More detailed examination shows a common core-periphery structure (see [30] for further discussion of this particular structure). Jeub et al. [31] provide a more complete picture by identifying graphs with downward profile (low-dimensional structure) and flat profile (expanders). The authors of latter work also introduce *conductance ratio profile*, which measures quality by the ratio between global conductance and internal conductance. Figure 1.2 shows idealized representations of some basic structures found in real networks. They can be nested or combined to produce complicated topologies.

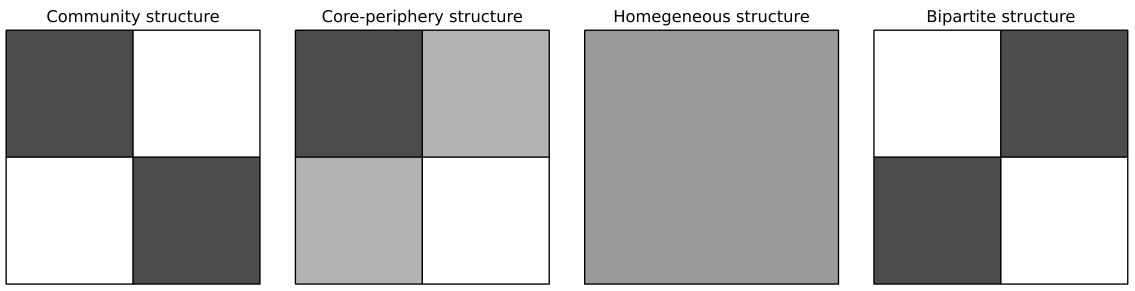


Figure 1.2: Some common network structures, represented by idealized adjacency matrices

Lancichinetti et al. [21] study networks from five domains (communication, internet, information, biological, and social). The authors use several algorithms to find communities, then calculate various statistics on the found groups: scaled link density, average shortest path length, maximum internal degree, and fraction of internal degree. Those statistics show similarity between networks from the same domain and indicate typical domain structures: star-like hubs, tree-like structures, homogeneous groups.

Dao et al. [19] also study various statistics of communities produced by several algorithms, but they compare the distribution of those statistics with that of metadata communities. There are some correlation, but still notable differences between two kinds of communities.

Dao et al. [32] use ground truth communities in several large networks. The authors use two statistics: the mean and standard deviation of the *out degree fraction*, and based on those identify six types of communities. The networks studied possess different composition of those types, so we have a simple method to identify structural differences between networks.

Dao et al. [33] use a similar approach to two previous works, but study a comprehensive set of statistics on communities discovered by several algorithms. The authors identify transitivity and hub dominance as the key measures characterizing four kinds of community topology: string-based, grid-based, star-based, and clique-based. The community profiles of various real networks as well as random graphs are described in detail.

Overall, these studies showcase the rich structures of networks, with many kinds of building blocks interacting with each other.

1.3 The topics of this thesis

This thesis is an exposition of two main topics: modularity as a clustering quality function, and spectral clustering properties of the random walk matrix.

Modularity was first introduced in [34] to select the number of communities in a dendrogram. Since then, it has become one of the most popular and well-studied quality functions. Chapter 2 introduces its basic properties and several shortcomings.

The random walk matrix is one of the basic matrices associated to a graph. If the graph has reasonably clear community structure, the top eigenvectors of that matrix can help us identify the groups. Chapter 3 explains clustering properties of the random walk matrix and introduces Walktrap - a clustering algorithm based on those properties.

Due to limited computational resources, all the experiments are carried out on small random graphs generated from stochastic block models. However, those small networks already suffice to illustrate some main points of our experiments.

Chapter 2

MODULARITY

This chapter is a detailed exposition of modularity - a popular clustering quality function. Section 2.1 defines modularity and gives the standard interpretation based on the configuration model. Section 2.2 presents basic properties of modularity, including modularity of some special graphs (cycles, complete multipartite graphs, ...). Section 2.3 explains several shortcomings of modularity when used in community detection.

Note that unless specified otherwise, we only consider simple undirected graphs, i.e. undirected graphs with no parallel edges and no self-loops.

2.1 Definition of modularity

Modularity, first introduced in [34], is now one of the most popular quality functions in community detection. Its original use is to choose between partitions of a graph: a partition with better modularity is considered to have better community structure. Since then, modularity has acquired a life of its own and some algorithms try to optimize it directly, giving modularity an additional role of being an objective function.

Definition

Following some authors, we separate two components in the modularity formula, which makes later theoretical discussion more convenient.

Definition 2.1. *Let G be a graph with $m \geq 1$ edges and $\mathcal{P}$ a vertex partition of G. Corresponding to the partition $\mathcal{P}$, we define the* edge contribution

$$q^E_{\mathcal{P}}(G) := \frac{1}{m} \sum_{P \in \mathcal{P}} e(P),$$

the degree tax

$$q_{\mathcal{P}}^{D}(G) := \frac{1}{4m^2} \sum_{P \in \mathcal{P}} \mathrm{vol}(P)^2,$$

and the modularity

$$q_{\mathcal{P}}(G) := q_{\mathcal{P}}^{E}(G) - q_{\mathcal{P}}^{D}(G). \tag{2.1}$$

The modularity $q^*(G)$ *of* G *is the maximum modularity over all partitions:*

$$q^*(G) := \max_{\mathcal{P}} q_{\mathcal{P}}(G).$$

By convention, we define the modularity of a graph with no edges to be 0.

The edge contribution is also called the *coverage*.

Interpretation

We can rewrite the modularity formula in a more illuminating way. Let G be a graph with n vertices and $m \geq 1$ edges, and let $\mathcal{P}$ be a vertex partition of G. For a vertex u we define $\sigma_{\mathcal{P}}(u)$ to be the unique set $P \in \mathcal{P}$ such that $u \in P$; we can consider $\sigma_{\mathcal{P}}$ to be the labeling defined by $\mathcal{P}$. Let A be the adjacency matrix of G: A is an $n \times n$ matrix, $A_{uv} = 1$ if $uv \in E(G)$ and 0 otherwise. Then

$$\begin{aligned}
q_{\mathcal{P}}(G) &= \frac{1}{2m} \sum_{P \in \mathcal{P}} \left(2e(P) - \frac{\mathrm{vol}(P)^2}{2m} \right) \\
&= \sum_{P \in \mathcal{P}} \left[\frac{e(P)}{m} - \left(\frac{\mathrm{vol}(P)}{2m} \right)^2 \right] \qquad (2.2) \\
&= \frac{1}{2m} \sum_{P \in \mathcal{P}} \sum_{u,v \in P} \left(A_{uv} - \frac{\deg(u)\deg(v)}{2m} \right) \\
&= \frac{1}{2m} \sum_{u,v \in V(G)} \left(A_{uv} - \frac{\deg(u)\deg(v)}{2m} \right) \cdot 1_{\sigma_{\mathcal{P}}(u) = \sigma_{\mathcal{P}}(v)}. \qquad (2.3)
\end{aligned}$$

Consider a random graph obtained from G by rewiring edges randomly such that all vertex degrees are preserved. We can visualize the process as following: cut off each edge in half to form two ‘stubs’, then join the $2m$ stubs randomly to form new edges. For any two vertices u and v, each ‘stub’ at u will be joined to v with probability $\frac{\deg(v)}{2m-1}$ (a ‘stub’ cannot be joined to itself), so by linearity of expectation, the expected number of new edges joining u and v is

$$\frac{\deg(u)\deg(v)}{2m-1}.$$

If m is large, $2m \approx 2m-1$, so (2.2) and (2.3) shows that modularity is large when the partition sets have more inside edges, i.e. are denser, than a random model. This model is considered to have no community structure because any two vertices can be connected regardless of their neighborhoods. Therefore a high modularity can be considered to be indicative of community structure.

To be more precise, let $\mathcal{P} = \{P_1, \dots, P_k\}$ be a partition of G with maximum modularity. If we merge, say, P_1 and P_2, the change in modularity (new value minus old value) is

$$\frac{e(P_1, P_2)}{m} - \frac{\mathrm{vol}(P_1)\mathrm{vol}(P_2)}{2m^2}.$$

Since $\mathcal{P}$ has maximum modularity, this change is non-positive, i.e.

$$\frac{e(P_1, P_2)}{m} \le \frac{\mathrm{vol}(P_1)\mathrm{vol}(P_2)}{2m^2}. \tag{2.4}$$

The LHS is the (global) edge density between P_1 and P_2, and the RHS is the (approximate) expected density in the configuration model. So we see that edge density between parts of $\mathcal{P}$ is lower than expected.

On the other hand, (2.4) also holds with P_2 replaced by $P_3, \dots, P_k$. Summing up all those inequalities, we obtain

$$\frac{e(P_1, \overline{P_1})}{m} \le \frac{\mathrm{vol}(P_1)\mathrm{vol}(\overline{P_1})}{2m^2},$$

where $\overline{P_1} := V \setminus P_1$. This implies

$$\frac{\mathrm{vol}(P_1) - 2e(P_1)}{2m} \le \frac{\mathrm{vol}(P_1)}{2m}\left(1 - \frac{\mathrm{vol}(P_1)}{2m}\right),$$

and so

$$\frac{e(P_1)}{m} \ge \left(\frac{\mathrm{vol}(P_1)}{2m}\right)^2.$$

Therefore the density of edges inside each community is greater than the expected density from the configuration model. Interestingly, we also have that each term in (2.2) is nonnegative.

Note that the configuration model above allows self-loops and parallel edges, and to obtain a formula resembling modularity we need an approximation. Therefore the model should only be considered as an heuristic to motivate the definition of modularity. We mathematically define modularity as in Definition 2.1, and work directly with that definition only.

Some simple bounds

To find out the range of modularity, formula (2.1) is more useful. The edge contribution is easy to bound:

$$0 \leq q^E_{\mathcal{P}}(G) \leq 1.$$

The lower bound is achieved when there are no edges inside members of $\mathcal{P}$, and the upper bound is achieved when there are no edges in-between members of $\mathcal{P}$. Generally speaking, merging members of $\mathcal{P}$ will increase the number of inside edges (and decrease the number of in-between edges), so edge contribution rewards partitions with few communities.

To bound the degree tax, let $\mathcal{P} = \{P_1, \ldots, P_k\}$. We have

$$q^D_{\mathcal{P}}(G) = \frac{1}{4m^2}\sum_{i=1}^{k} \mathrm{vol}(P_i)^2 \leq \frac{1}{4m^2}\left(\sum_{i=1}^{k} \mathrm{vol}(P_i)\right)^2 = 1,$$

and by a simple application of the Cauchy-Schwarz inequality,

$$q^D_{\mathcal{P}}(G) = \frac{1}{4m^2}\sum_{i=1}^{k} \mathrm{vol}(P_i)^2 \geq \frac{1}{4m^2 k}\left(\sum_{i=1}^{k} \mathrm{vol}(P_i)\right)^2 = \frac{1}{k}. \tag{2.5}$$

The upper bound is achieved when there is only a single set P_i with positive volumes, and the lower bound is achieved when all sets P_i's have the same volumes. In general, to lower the degree tax, we need many communities of approximately the same volume.

Combining the above bounds, we see that

$$-1 \leq q_{\mathcal{P}}(G) < 1$$

for all partition $\mathcal{P}$. The lower bound can actually be improved to $-1/2$, see Proposition 2.5 below. Modularity is maximized when we can balance between maximizing edge contribution, which requires few communities, and minimizing degree tax, which requires many communities.

Maximum modularity can be bounded by

$$0 \leq q^*(G) < 1.$$

The upper bound follows from the upper bounds for all partitions. To obtain the lower bound, notice that the trivial partition where all vertices belong to the same community gives us modularity 0.

Modularity for weighted graphs

Occasionally we need to use modularity for weighted graphs with self-loops. A *weighted graph* (with self-loops) G is a set of vertices V together with a symmetric weight function $w : V^2 \to [0, \infty)$. There is no need for an edge set, since non-existent links are just edges with weight 0. The number of vertices is $n := |V|$, and the total edge weight is $m := \sum_{u,v \in V} w(u, v)$. A technical issue is how self-loops contribute to vertex degrees. In the configuration model, each split edge creates two stubs, so self-loops should contribute twice to degree:

$$\deg(u) = 2w(u, u) + \sum_{v \neq u} w(u, v).$$

For a subset P, volume is the sum of all degrees:

$$\operatorname{vol}(P) := \sum_{v \in P} \deg(v),$$

and $e(P)$ is the total edge weight inside P (we only count distinct edges):

$$e(P) := \sum_{u \in P} w(u, u) + \sum_{(u,v) \in \binom{P}{2}} w(u, v).$$

Modularity is then defined exactly as before:

$$q_{\mathcal{P}}(G) := \sum_{P \in \mathcal{P}} \left[\frac{e(P)}{m} - \left(\frac{\operatorname{vol}(P)}{2m} \right)^2 \right].$$

This generalization is not just a theoretical exercise, but also practically useful. A partition $\mathcal{P} = \{P_1, \ldots, P_k\}$ of a (non-weighted) graph G can be considered as a summary of G by a weighted graph with k vertices $v_1, \ldots, v_k$, where edge weight $w(v_i, v_j)$ is the number of edges between P_i and P_j, and self-loop weight $w(v_i, v_i)$ is the number of edges inside P_i. This weighted summary can be used to keep track of modularity in a merging algorithm (see Appendix A for a concrete application).

2.2 Basic properties

This section is theoretical, consisting only of theorems and proofs. Unless stated otherwise, the proof comes from or is based on ideas in the same source as the statement.

We begin with some intuitive (and desirable) behaviors of modularity. To avoid repetition, instead of writing the full phrase ‘partition with maximum modularity’, we sometimes say ‘maximum modular partition’, or just ‘optimal partition’.

Lemma 2.2 ([35, Lemma 3.4]). *Let G be a graph. Then there is a vertex partition $\mathcal{P}$ of maximum modularity such that for each member $P \in \mathcal{P}$, the restriction of G to P is a connected graph.*

Proof. Assume that $P \in \mathcal{P}$ can be split into A and B such that there are no edges between A and B. Let $\mathcal{P}' = \mathcal{P} \setminus \{P\} \cup \{A, B\}$. The edge contribution remains the same because there are no edges between A and B, but the degree tax decreases because

$$\mathrm{vol}(P)^2 = (\mathrm{vol}(A) + \mathrm{vol}(B))^2 \geq \mathrm{vol}(A)^2 + \mathrm{vol}(B)^2,$$

where the inequality is strict if $\mathrm{vol}(A) \cdot \mathrm{vol}(B) > 0$. So $q_{\mathcal{P}'} \geq q_{\mathcal{P}}$. We can continue this process to obtain a refinement of $\mathcal{P}$ containing no disconnected members. □

Isolated vertices have no impact on modularity.

Lemma 2.3 ([35, Corollary 3.2]). *Let G be a graph with $m \geq 1$ edges and v an isolated vertex (i.e. $\deg(v) = 0$). Let $\mathcal{P} = \{P_1, \ldots, P_k\}$ be a partition of $V(G) \setminus \{v\}$. For each $i = 1, \ldots, k$ define*

$$\mathcal{P}_i = \{P_1, \ldots, P_i \cup \{v\}, \ldots, P_k\},$$

and define

$$\mathcal{P}_0 = \{\{v\}, P_1, \ldots, P_k\}.$$

Then for each $i = 0, 1, \ldots, k$,

$$q_{\mathcal{P}_j}(G) = q_{\mathcal{P}}(G \setminus \{v\}).$$

Proof. Easily seen from the modularity formula, because v contributes nothing to edge counts or degree sum of each subset. □

There are no dangling vertices in a maximum modular partition.

Lemma 2.4 ([36, Lemma 1.6.5]). *Let G be a graph and $\mathcal{P}$ an optimal vertex partition of G. Then $P = \{u\}$ for some $P \in \mathcal{P}$ implies $\deg(u) = 0$, i.e. u is an isolated vertex.*

Proof. Assume that $\deg(u) = d > 0$ and that $\mathcal{P} = \{\{u\}, P_1, \ldots, P_k\}$ is an optimal partition of G. For each $i = 1, \ldots, k$, define a new partition $\mathcal{P}_i = \{P_1, \ldots, P_i \cup \{u\}, \ldots, P_k\}$. After some easy calculation we obtain

$$q_{\mathcal{P}_i} - q_{\mathcal{P}} = \frac{1}{m} \cdot e(u, P_i) - \frac{1}{2m^2} \cdot \deg(u)\mathrm{vol}(P_i).$$

Since $\mathcal{P}$ is an optimal partition, $q_{\mathcal{P}_i} \leq q_{\mathcal{P}}$ and therefore $2m \cdot e(u, P_i) \leq \deg(u)\mathrm{vol}(P_i)$ for each i. Summing over $i = 1, \ldots, k$ we obtain

$$2md \leq d(2m - d) < 2md,$$

a contradiction. □

Next are some general bounds on modularity.

Proposition 2.5 ([35, Lemma 3.1])**.** *Let $G = (V, E)$ be a graph with $m \geq 1$ edges and let $\mathcal{P}$ be a partition of G. Then*

$$-\frac{1}{2} \leq q_{\mathcal{P}}(G) < 1.$$

Proof. The upper bound is obvious. To prove the lower bound, we use a new representation of modularity. Let $\mathcal{P} = \{P_1, \ldots, P_k\}$. For each $i = 1, \ldots, k$, set

$$e_i = E_G(P_i), \quad \bar{e}_i = E_G(P_i, V \setminus P_i).$$

Then we can rewrite the modularity formula as

$$q_{\mathcal{P}}(G) = \sum_{i=1}^{k} \left[\frac{e_i}{m} - \left(\frac{e_i}{m} + \frac{\bar{e}_i}{2m} \right)^2 \right].$$

Note that $0 \leq e \leq m - \bar{e}_i$. The function

$$f(x) = x - \left(x + \frac{\bar{e}_i}{2m} \right)^2, \quad 0 \leq x \leq 1 - \frac{\bar{e}_i}{m},$$

is a concave quadratic function with maximum at the point $x_0 = \frac{m - \bar{e}_i}{2m}$, so

$$f(x) \geq f(0) = f\left(1 - \frac{\bar{e}_i}{m}\right) = -\left(\frac{\bar{e}_i}{2m}\right)^2.$$

If there are no edges between parts of $\mathcal{P}$ (i.e. $\bar{e}_i = 0$ for all i), then

$$q_{\mathcal{P}}(G) = \sum_{i=1}^{k} \left[\frac{e_i}{m} - \left(\frac{e_i}{m} \right)^2 \right] \geq 0,$$

since $0 \leq e_i/m \leq 1$. Now assume that there are some edges between parts of $\mathcal{P}$; in particular, $k \geq 2$. We delete all edges within parts of $\mathcal{P}$ to obtain a new graph $G_0 = (V, E_0)$ with m_0 edges, $1 \leq m_0 \leq m$. The edges between partition members are kept intact. We then have

$$q_{\mathcal{P}}(G) \geq -\sum_{i=1}^{k} \left(\frac{\bar{e}_i}{2m} \right)^2 \geq -\sum_{i=1}^{k} \left(\frac{\bar{e}_i}{2m_0} \right)^2 = q_{\mathcal{P}}(G_0).$$

It suffices to find a lower bound for $q_{\mathcal{P}}(G_0)$. The algebra is simple but there are quite a lot of indices, so for convenience we introduce a visual picture of all the variables. Let K_k be the complete graph on vertices $[k] := \{1, \ldots, k\}$; this graph is used for keeping track of variables only. For $(i, j) \in E(K_k)$ let $x_{ij} := e_{G_0}(P_i, P_j)$. We immediately have

$$m_0 = \sum_{e \in E(K_k)} x_e,$$

and

$$\mathrm{vol}_{G_0}(P_i) = \sum_{\substack{e\in E(K_k)\\ i\in e}} x_e.$$

Thus,

$$\begin{aligned}\sum_{i=1}^{k} \mathrm{vol}_{G_0}(P_i)^2 &= \sum_{i=1}^{k}\left(\sum_{\substack{e\in E(K_k)\\ i\in e}} x_e\right)^2\\ &= 2\cdot \sum_{e\in E(K_k)} x_e^2 + 2\cdot \sum_{\substack{e,f\in E(K_k)\\ e \text{ adjacent to } f}} x_e x_f,\end{aligned}$$

and

$$\begin{aligned}4m_0^2 &= 4\cdot \sum_{e\in E(K_k)} x_e^2 + 8\cdot \sum_{\substack{e,f\in E(K_k)\\ e\neq f}} x_e x_f\\ &\geq 4\cdot \sum_{e\in E(K_k)} x_e^2 + 8\cdot \sum_{\substack{e,f\in E(K_k)\\ e \text{ adjacent to } f}} x_e x_f\\ &\geq 2\cdot \sum_{i=1}^{k} \mathrm{vol}_{G_0}(P_i)^2.\end{aligned}$$

Therefore $-q_{\mathcal{P}}(G_0) \leq \frac{1}{2}$ and we are done. Tracing back the proof, we see that the equality holds only in the case of bipartite graphs with the natural partition. □

For the next result, we need the concept of a *detachment* of a graph. Let G be a graph with vertex set $V(G) = \{v_1, \ldots, v_n\}$. We say a graph H is a *detachment* of G if H admits a vertex partition $\mathcal{I} = \{I_1, \ldots, I_n\}$ such that each I_i is an independent set and

$$e_H(I_i, I_j) = \begin{cases} 1 & \text{if } v_iv_j \in E(G),\\ 0 & \text{otherwise.}\end{cases}$$

Lemma 2.6 ([36, Lemma 1.4.1])**.** *Let H be a detachment of G. Then $q^*(H) \geq q^*(G)$.*

Proof. Let G be a graph with m edges and vertex set $V(G) = \{v_1, \ldots, v_n\}$. Let $\mathcal{I}$ be a vertex partition of H compatible with G (from the definition of detachment).

Let $\mathcal{P} = \{P_1, \ldots, P_k\}$ be an optimal partition of G. For each $i = 1, \ldots, k$, define a corresponding vertex subset of H:

$$P_i' = \bigcup_{j:v_j\in P_i} I_j.$$

This gives us a partition $\mathcal{P}' = \{P_1', \dots, P_k,\}$ of H. Note that G and H have the same number of edges, and it is easy to check that for each i we have $e_G(P_i) = e_H(P_i')$ and $\mathrm{vol}_G(P_i) = \mathrm{vol}_H(P_i')$. Thus $q^*(H) \geq q_{\mathcal{P}'}(H) = q_{\mathcal{P}}(G) = q^*(G)$. □

Proposition 2.7 ([36, Corollary 1.4.2])**.** *Let G be a graph with $m \geq 1$ edges. Then $q^*(G) \leq 1 - 1/m$.*

Proof. If G has a vertex v with degree $d > 1$, we replace v by an independent set I of d new vertices, and connect each new vertex to one and only one neighbor of v. The new graph G' is a detachment of G, so by Lemma 2.6 $q^*(G') \geq q^*(G)$. We continue this operation until we arrive at a graph H in which no vertex has degree greater than 1, i.e. H has m disjoint edges and some isolated vertices, and $q^*(H) \geq q^*(G)$. Since isolated vertices do not affect modularity, we discard all of them, and consider H to be a graph of $2m$ vertices and m disjoint edges. By Lemma 2.2, each member of the optimal partition is either an edge or a single vertex. Since H has no isolated vertices, Lemma 2.4 implies that each partition member must be an edge. So the only optimal partition is to put each edge of H in a single subset, which gives us the modularity $1 - 1/m$. □

A natural follow-up question is: what is the maximum modularity of *connected* graphs with m edges?

Proposition 2.8 ([37, Proposition 10])**.** *Let G be a graph with $m \geq 1$ edges. If G is connected then*

$$q^*(G) \leq 1 - \frac{2}{\sqrt{m}} + \frac{1}{m},$$

and if G is 2-edge-connected then

$$q^*(G) \leq 1 - \frac{2}{\sqrt{m}}.$$

Proof. Let $\mathcal{P} = \{P_1, \dots, P_k\}$ be a partition of G. Since G is connected, there are at least $k - 1$ edges in-between members of $\mathcal{P}$, so

$$q_{\mathcal{P}}^E(G) \leq 1 - \frac{k-1}{m}.$$

Combining with the degree tax lower bound (2.5), we obtain

$$q_{\mathcal{P}}(G) \leq 1 - \frac{k-1}{m} - \frac{1}{k} = 1 + \frac{1}{m} - \left(\frac{k}{m} + \frac{1}{k}\right) \leq 1 + \frac{1}{m} - \frac{2}{\sqrt{m}}.$$

If G is 2-edge-connected instead, then there are at least k in-between edges, and

$$q_{\mathcal{P}}(G) \leq 1 - \frac{k}{m} - \frac{1}{k} \leq 1 - \frac{2}{\sqrt{m}}.$$

□

The upper bounds above can be tight, as in the cycle graph. We state and prove an asymptotic version only; precise results can be found in the cited reference.

Proposition 2.9 ([35, Theorem 6.7]). *Let C_n be the cycle on n vertices. Then*

$$q^*(C_n) = 1 - \frac{2}{\sqrt{n}} + o(1).$$

Proof. Let $\mathcal{P} = \{P_1, \dots, P_k\}$ be a partition of C_n. By Lemma 2.2, we should choose each P_i to be a connected segment of vertices. Set $x_i = |P_i|$, then $\mathrm{vol}(P_i) = 2x_i$. The modularity of the partition is

$$q_{\mathcal{P}}(C_n) = 1 - \frac{k}{n} - \frac{1}{n^2} \sum_{i=1}^{k} x_i^2.$$

Pick $k = \sqrt{n}$ and $x_i = n/k = \sqrt{n}$ (ignoring integer rounding), we have $q_{\mathcal{P}}(C_n) = 1 - 2/\sqrt{n}$. □

This result already hints at some issues of modularity as a quality function for community structure. All vertices on the cycle are absolutely equivalent, so there is no natural grouping, yet the modularity is still very high.

Now we calculate the modularity of some other familiar graphs.

Proposition 2.10 ([35, Corollary 6.2]). *Let K_n be the complete graphs on n vertices. Then $q^*(K_n) = 0$.*

Proof. Let $\mathcal{P} = \{P_1, \dots, P_k\}$ be a partition of K_n. Set $x_i = |P_i|$, then $\mathrm{vol}(P_i) = (n-1)x_i$. We have

$$\begin{aligned}
q_{\mathcal{P}}(K_n) &= \frac{1}{m} \sum_{i=1}^{k} \binom{x_i}{2} - \frac{1}{4m^2} \cdot (n-1)^2 \cdot \sum_{i=1}^{k} x_i^2 \\
&= \frac{1}{n(n-1)} \sum_{i=1}^{k} x_i(x_i - 1) - \frac{1}{n^2} \sum_{i=1}^{k} x_i^2 \\
&= \frac{1}{n^2(n-1)} \sum_{i=1}^{k} x_i^2 - \frac{1}{n-1} \\
&\leq \frac{1}{n^2(n-1)} \left(\sum_{i=1}^{k} x_i \right)^2 - \frac{1}{n-1} \\
&= \frac{1}{n-1} - \frac{1}{n-1} = 0.
\end{aligned}$$

Equality holds only when all vertices belong to the same group. □

Proposition 2.11. *Let G consist of disconnected complete cliques. Then the cliques form an optimal partition of G.*

Proof. By Lemma 2.3 we can ignore isolated vertices. Lemma 2.2 implies that each partition subset lies completely inside a clique. We repeat the argument in the proof of Proposition 2.10. Let $P_1, \dots, P_k$ be the partition members inside a clique of size a, and let $x_i = |P_i|$. Note that $m \geq \binom{a}{2}$. The contribution to modularity from the P_i's is

$$\begin{aligned} &\frac{1}{m}\sum_{i=1}^{k}\binom{x_i}{2} - \frac{(a-1)^2}{4m^2}\sum_{i=1}^{k} x_i^2 \\ &= \left(\frac{1}{2m} - \frac{(a-1)^2}{4m^2}\right)\sum_{i=1}^{k} x_i^2 - \frac{a}{2m}, \end{aligned}$$

which is maximized only when $k = 1$, i.e. the whole clique is a partition member. $\square$

The following result will later be generalized to multipartite graphs. However, the proof in the bipartite case is much simpler and already contains the key idea.

Theorem 2.12 ([36, Theorem 1.3.5])**.** *Let G be a complete bipartite graph on vertex sets U, V. Then $q^*(G) = 0$.*

Proof. Let $\mathcal{P} = \{P_1, \dots, P_k\}$ be any vertex partition. For each $1 \leq i \leq k$, set $U_i = P_i \cap U$, $V_i = P_i \cap V$, $u_i = |U_i|$, and $v_i = |V_i|$. G has $|U||V|$ edges, each edge in U has degree $|V|$ and each edge in V has degree $|U|$. We have the edge contribution:

$$q^E_{\mathcal{P}}(G) = \frac{1}{|U||V|}\sum_{i=1}^{k} u_i v_i,$$

and the degree tax:

$$q^D_{\mathcal{P}}(G) = \frac{1}{4|U|^2|V|^2}\sum_{i=1}^{k}\left(u_i|V| + v_i|U|\right)^2.$$

Therefore the modularity is

$$\begin{aligned} q_{\mathcal{P}}(G) &= \frac{1}{4|U|^2|V|^2}\sum_{i=1}^{k}\left(4|U||V|u_i v_i - (u_i|V| + v_i|U|)^2\right) \\ &= \frac{1}{4|U|^2|V|^2}\sum_{i=1}^{k}\left(-(u_i|V| - v_i|U|)^2\right) \\ &\leq 0. \end{aligned}$$

We also see that the partition $\mathcal{P}$ has modularity 0 if and only if $\frac{u_i}{|U|} = \frac{v_i}{|V|}$ for all i, i.e. each partite set contributes the same proportion of vertices to each partition member. $\square$

Theorem 2.13. *Let G be a complete multipartite graph. Then $q^*(G) = 0$.*

Proof. In this proof, we will use superscripts to index multipartite sets and subscripts to index partition members. A mix of both should have obvious meaning.

Let the partite sets of G be $U^{(1)}, \ldots, U^{(d)}$, and let $x^{(j)} = |U^{(j)}|$. So G has

$$x := \sum_{j=1}^{d} x^{(j)}$$

vertices and

$$m := \sum_{1 \le j_1 < j_2 \le d} x^{(j_1)} x^{(j_2)}$$

edges. Let $\mathcal{P} = \{P_1, \ldots, P_k\}$ be an arbitrary vertex partition of G. For each $i = 1, \ldots, k$ and $j = 1, \ldots, d$, let $x_i^{(j)} = |P_i \cap U^{(j)}|$, and $x_i = |P_i|$. For convenience, we collect the variables in the following table:

x	$x^{(1)}$	$\ldots$	$x^{(d)}$
x_1	$x_1^{(1)}$	$\ldots$	$x_1^{(d)}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$
x_k	$x_k^{(1)}$	$\ldots$	$x_k^{(d)}$

The edge contribution is

$$q^E_{\mathcal{A}} = \frac{1}{m} \sum_{i=1}^{k} \sum_{1 \le j_1 < j_2 \le d} x_i^{(j_1)} x_i^{(j_2)}.$$

The degree tax is

$$\begin{aligned} q^D_{\mathcal{P}} &= \frac{1}{4m^2} \sum_{i=1}^{k} \left[\sum_{j=1}^{d} x_i^{(j)} (x - x^{(j)}) \right]^2 \\ &= \frac{1}{4m^2} \sum_{i=1}^{k} \left[\sum_{j=1}^{d} \left[x_i^{(j)} (x - x^{(j)}) \right]^2 + 2 \sum_{1 \le j_1 < j_2 \le d} x_i^{(j_1)} x_i^{(j_2)} (x - x^{(j_1)})(x - x^{(j_2)}) \right]. \end{aligned}$$

So the modularity is

$$\begin{aligned} q_{\mathcal{P}} = \frac{1}{4m^2} \sum_{i=1}^{k} \Bigg[&- \sum_{j=1}^{d} \left[x_i^{(j)} (x - x^{(j)}) \right]^2 \\ &+ 2 \sum_{1 \le j_1 < j_2 \le d} x_i^{(j_1)} x_i^{(j_2)} \left(2m - (x - x^{(j_1)})(x - x^{(j_2)}) \right) \Bigg]. \end{aligned}$$

For fixed i and $j_1 < j_2$, we have the inequality (a guess based on the equality in the bipartite case):

$$\begin{aligned}
&2x_i^{(j_1)}x_i^{(j_2)}\left(2m - (x - x^{(j_1)})(x - x^{(j_2)})\right)\\
&= 2 \cdot \frac{x_i^{(j_1)}}{x^{(j_1)}} \cdot \frac{x_i^{(j_2)}}{x^{(j_2)}} \cdot x^{(j_1)}x^{(j_2)}\left(2m - (x - x^{(j_1)})(x - x^{(j_2)})\right)\\
&\le \left(\left[\frac{x_i^{(j_1)}}{x^{(j_1)}}\right]^2 + \left[\frac{x_i^{(j_2)}}{x^{(j_2)}}\right]^2\right) \cdot x^{(j_1)}x^{(j_2)}\left(2m - (x - x^{(j_1)})(x - x^{(j_2)})\right).
\end{aligned}$$

We sum this over $1 \le j_1 < j_2 \le d$, then plug back into the square brackets in the expression for $q_{\mathcal{P}}$. For each i and j, we collect the terms containing $x_i^{(j)}$:

$$\begin{aligned}
&\frac{\left[x_i^{(j)}\right]^2}{4m^2} \cdot \left[-(x - x^{(j)})^2 + \sum_{j_1 : j_1 \ne j} \frac{x^{(j_1)}}{x^{(j)}} \cdot \left(2m - (x - x^{(j)})(x - x^{(j_1)})\right)\right]\\
&= \frac{\left[x_i^{(j)}\right]^2}{4m^2} \cdot \frac{x - x^{(j)}}{x^{(j)}} \cdot \left[2m - \sum_{j_1=1}^{d} x^{(j_1)}(x - x^{(j_1)})\right]\\
&= \frac{\left[x_i^{(j)}\right]^2}{4m^2} \cdot \frac{x - x^{(j)}}{x^{(j)}} \cdot 0 = 0.
\end{aligned}$$

Therefore $q_{\mathcal{P}} \le 0$ and we are done. Equality holds if and only if

$$\frac{x_i^{(j_1)}}{x^{(j_1)}} = \frac{x_i^{(j_2)}}{x^{(j_2)}}, \quad \text{for all } 1 \le i \le k,\ 1 \le j_1 < j_2 \le d.$$

In other words, we obtain modularity 0 only when each partition member contains the same proportion of vertices form each partite set. □

For alternative proofs using matrix analysis, see [38, 39].

For a comprehensive list of modularity of many graph classes, see the table at the end of [40].

Next are some results concerning the robustness of modularity. First of all, since searching over all partitions of a set is prohibitively expensive, we would like to know how good a limited search over partitions with few members can be.

Proposition 2.14 ([41, Lemma 1])**.** *Let G be a graph with $m \ge 1$ edges, and let t be a positive integers. Then*

$$\max_{\mathcal{P} : |\mathcal{P}| \le t} q_{\mathcal{P}}(G) \ge \left(1 - \frac{1}{t}\right) q^*(G).$$

Proof. We use the probabilistic method. Let $\mathcal{P}$ be an optimal partition of G. If $|\mathcal{P}| \leq t$ then we are done. Otherwise, we construct a new partition $\mathcal{P}'$ by randomly assign each member in $\mathcal{P}$ to one of t labeled 'buckets', then merge all members in each bucket. By construction, $\mathcal{P}'$ is a random partition and $|\mathcal{P}'| \leq t$. For $u, v \in G$, set

$$M_{uv} = 1_{uv \in E(G)} - \frac{\deg(u)\deg(v)}{2m}.$$

Then

$$\sum_{u,v \in V(G)} M_{uv} = 0,$$

and

$$q_{\mathcal{P}}(G) = \frac{1}{2m} \sum_{u,v \in V(G)} M_{uv} 1_{\sigma_{\mathcal{P}}(u) = \sigma_{\mathcal{P}}(v)}.$$

If $\sigma_{\mathcal{P}}(u) = \sigma_{\mathcal{P}}(v)$, then $\sigma_{\mathcal{P}'}(u) = \sigma_{\mathcal{P}'}(v)$. If $\sigma_{\mathcal{P}}(u) \neq \sigma_{\mathcal{P}}(v)$, then $\sigma_{\mathcal{P}'}(u) = \sigma_{\mathcal{P}'}(v)$ with probability $1/t$. Therefore the expectation of the new modularity is

$$\begin{aligned} \mathbb{E}\, q_{\mathcal{P}'}(G) &= \frac{1}{2m}\left[\sum_{u,v \in V(G)} M_{uv} 1_{\sigma_{\mathcal{P}}(u) = \sigma_{\mathcal{P}}(v)} + \frac{1}{t} \sum_{u,v \in V(G)} M_{uv} 1_{\sigma_{\mathcal{P}}(u) \neq \sigma_{\mathcal{P}}(v)}\right] \\ &= \left(1 - \frac{1}{t}\right) \cdot \frac{1}{2m} \sum_{u,v \in V(G)} M_{uv} 1_{\sigma_{\mathcal{P}}(u) = \sigma_{\mathcal{P}}(v)} \\ &= \left(1 - \frac{1}{t}\right) q^*(G). \end{aligned}$$

This implies that there is at least one partition $\mathcal{P}'$ satisfying the conclusion of the lemma. □

Network data are often noisy, with missing edges as well as redundant ones. The next several results bound the change in modularity when we perturb the edge sets.

Proposition 2.15 ([40, Lemma 5.1])**.** *Let $G = (V, E)$ be a graph, let E_0 be a non-empty subset of E, let $E' = E \setminus E_0$ and $G' = (V, E')$. Then*

$$|q^*(G) - q^*(G')| < \frac{2|E_0|}{|E|}. \tag{2.6}$$

Proof. We may assume that G' has at least one edge. Let $\mathcal{P} = \{P_1, \ldots, P_k\}$ be any partition of V, E_1 the set of edges in E_0 lying withins parts of $\mathcal{P}$, and E_2 the set of edges in E_0 lying between parts of $\mathcal{P}$. Set

$$\alpha = \alpha(\mathcal{P}) = \frac{|E_1|}{|E|}, \quad \beta = \beta(\mathcal{P}) = \frac{|E_2|}{|E|}.$$

Note that $\alpha + \beta = |E_0|/|E|$. We will prove the following two strict inequalities:

$$q_{\mathcal{P}}(G') - q_{\mathcal{P}}(G) < 2\alpha + 2\beta, \tag{2.7}$$

$$q^*(G) - q^*(G') < 2\alpha + \beta. \tag{2.8}$$

These two suffice to establish (2.6). Indeed, suppose first that $q^*(G') \geq q^*(G)$. Taking $\mathcal{P}$ to be an optimal partition for G', (2.7) gives us

$$|q^*(G) - q^*(G')| = q_{\mathcal{P}}(G') - q^*(G) \leq q_{\mathcal{P}}(G') - q_{\mathcal{P}}(G) < 2\alpha + 2\beta = \frac{2|E_0|}{|E|}.$$

On the other hand, if $q^*(G) > q^*(G')$, then (2.8) obviously implies (2.6).

Now we focus on proving (2.7) and (2.8). We calculate the change in edge contribution:

$$q^E_{\mathcal{P}}(G) = \frac{1}{|E|} \sum_{P \in \mathcal{P}} e_G(C) = \frac{1}{|E|} \left(|E_1| + \sum_{P \in \mathcal{P}} e_{G'}(P) \right) = \alpha + (1 - \alpha - \beta) q^E_{\mathcal{P}}(G').$$

So

$$q^E_{\mathcal{P}}(G) - q^E_{\mathcal{P}}(G') = \alpha - (\alpha + \beta) q^E_{\mathcal{P}}(G'). \tag{2.9}$$

In particular, since edge contribution is at most 1, we have

$$q^E_{\mathcal{P}}(G') - q^E_{\mathcal{P}}(G) \leq \beta. \tag{2.10}$$

Now we bound the change in degree tax. For each $i = 1, \ldots, k$, let

$$\alpha_i = \frac{|E_1 \cap E(C_i)|}{|E|}, \quad \beta_i = \frac{|E_2 \cap E(P_i, V \setminus P_i)|}{|E|}.$$

Note that $\sum_i \alpha_i = \alpha$, $\sum_i \beta_i = 2\beta$, and $\beta_i \leq \beta$. We have

$$\mathrm{vol}_G(P_i) - \mathrm{vol}_{G'}(P_i) = (2\alpha_i + \beta_i)|E|.$$

So

$$\begin{aligned}
&\sum_i \mathrm{vol}_G(P_i)^2 - \sum_i \mathrm{vol}_{G'}(P_i)^2 \\
&= \sum_i (\mathrm{vol}_G(P_i) + \mathrm{vol}_{G'}(P_i)) \cdot (\mathrm{vol}_G(P_i) - \mathrm{vol}_{G'}(P_i)) \\
&< 2|E| \sum_i \mathrm{vol}_G(P_i)(2\alpha_i + \beta_i) \\
&\leq 4|E| \left(\max_i \mathrm{vol}_G(P_i) \right) \sum_i \alpha_i + 2|E|\beta \sum_i \mathrm{vol}_G(P_i) \\
&\leq 4|E|^2 (2\alpha + \beta).
\end{aligned}$$

Combining with $|E'| < |E|$ we have

$$q^D_{\mathcal{P}}(G) - q^D_{\mathcal{P}}(G') < 2\alpha + \beta. \tag{2.11}$$

Two inequalities (2.10) and (2.11) then give us (2.7).

Moving on to proving (2.8), we have

$$q^D_{\mathcal{P}}(G) > \frac{1}{4|E|^2} \sum_{P \in \mathcal{P}} \mathrm{vol}_{G'}(P)^2 = (1-\alpha-\beta)^2 q^D_{\mathcal{P}}(G') > (1-2\alpha-2\beta) q^D_{\mathcal{P}}(G').$$

So

$$q^D_{\mathcal{P}}(G') - q^D_{\mathcal{P}}(G) < 2(\alpha+\beta) q^D_{\mathcal{P}}(G'),$$

which together with (2.9) imply

$$q_{\mathcal{P}}(G) - q_{\mathcal{P}}(G') < \alpha - (\alpha+\beta) q_{\mathcal{P}}(G') + (\alpha+\beta) q^D_{\mathcal{P}}(G'). \tag{2.12}$$

Fix $\mathcal{P}$ to be an optimal partition for G. If $q_{\mathcal{P}}(G') \geq 0$ then

$$q_{\mathcal{P}}(G) - q_{\mathcal{P}}(G') < \alpha + (\alpha+\beta) q^D_{\mathcal{P}}(G'),$$

and

$$\begin{aligned} q^*(G) - q^*(G') &= q_{\mathcal{P}}(G) - q^*(G') \\ &\leq q_{\mathcal{P}}(G) - q_{\mathcal{P}}(G') \\ &< \alpha + (\alpha+\beta) q^D_{\mathcal{P}}(G') \\ &\leq \alpha + (\alpha+\beta) = 2\alpha + \beta. \end{aligned}$$

On the other hand, if $q_{\mathcal{P}}(G') < 0$, then

$$\begin{aligned} &q^*(G) - q^*(G') \leq q^*(G) = q_{\mathcal{P}}(G) \\ &= q_{\mathcal{P}}(G) - q_{\mathcal{P}}(G') + q_{\mathcal{P}}(G') \\ &< \alpha + (1-\alpha-\beta) q_{\mathcal{P}}(G') + (\alpha+\beta) q^D_{\mathcal{P}}(G') \qquad \text{(using (2.12))} \\ &\leq \alpha + (\alpha+\beta) q^D_{\mathcal{P}}(G') \qquad \text{(since } \alpha+\beta \leq 1 \text{ and } q_{\mathcal{P}}(G') < 0) \\ &\leq 2\alpha + \beta. \end{aligned}$$

This concludes our proof. □

There is a similar bound when two graphs have the same number of edges.

Proposition 2.16 ([40, Lemma 5.2])**.** *Let $G = (V, E)$ and $G' = (V, E')$ be two distinct graphs on the same vertex set V, each with $m \geq 1$ edges. Then*

$$|q^*(G) - q^*(G')| < \frac{|E \triangle E'|}{m}.$$

Proof. Since $|E| = |E'|$, $|E \triangle E'|$ is an even number. By the triangle inequality, it suffices to consider the case $|E \triangle E'| = 2$. Let $E \triangle E' = \{e, e'\}$, where $e \in E \setminus E'$ and $e' \in E' \setminus E$. Assume without loss of generality that $q^*(G) \leq q^*(G')$. Let $\mathcal{P}$ be an optimal partition for G', it suffices to prove that

$$q_{\mathcal{P}}(G') < q_{\mathcal{P}}(G) + \frac{2}{m}. \tag{2.13}$$

We consider two cases, according to whether e lie within or between parts of $\mathcal{P}$. First, assume that e lie within a part $P \in \mathcal{P}$, which implies $q^E_{\mathcal{P}}(G') \leq q^E_{\mathcal{P}}(G)$. Setting $x = \mathrm{vol}_G(P)$, we have

$$q^D_{\mathcal{P}}(G) - q^D_{\mathcal{P}}(G') \leq \frac{x^2 - (x-2)^2}{4m^2} = \frac{x-1}{m^2} < \frac{x}{m^2} \leq \frac{2}{m},$$

and (2.13) holds in this case. Now suppose e connects two parts P_1 and P_2. Then

$$q^E_{\mathcal{P}}(G') - q^E_{\mathcal{P}}(G) \leq \frac{1}{m}.$$

Setting $x_1 = \mathrm{vol}_G(P_1)$ and $x_2 = \mathrm{vol}_G(P_2)$, we have (after considering several possible positions of e')

$$q^D_{\mathcal{P}}(G) - q^D_{\mathcal{P}}(G') \leq \frac{x_1^2 - (x_1-1)^2 + x_2^2 - (x_2-1)^2}{4m^2} = \frac{x_1 + x_2 - 1}{2m^2} < \frac{x_1 + x_2}{2m^2} \leq \frac{1}{m}.$$

Adding two inequalities above gives (2.13) and concludes our proof. □

The following extends two results above.

Proposition 2.17 ([40, Lemma 5.3])**.** *Let $G = (V, E)$ and $G = (V, E')$ be two distinct graphs on the same vertex set V with $|E| \geq |E'|, |E| > 0$. Then*

$$|q^*(G) - q^*(G')| < \frac{2|E \setminus E'|}{|E|}.$$

Proof. Let $E'' = E \cap E'$, and let F be a subset of $E \setminus E''$ containing $|E| - |E'|$ elements. Let H be the graph $(V, E' \cup F)$, a graph on the same vertex set V with $|E|$ edges. By Proposition 2.15,

$$|q^*(H) - q^*(G')| \leq \frac{2|F|}{|E|} = \frac{2(|E| - |E'|)}{|E|}.$$

By Proposition 2.16,

$$|q^*(H) - q^*(G)| \leq \frac{2(|E'| - |E''|)}{|E|}.$$

It is easy to check that either $H \neq G$ or $H \neq G'$, so at least one of the above inequality is strict. Therefore

$$\begin{aligned} |q^*(G) - q^*(G')| &\leq |q^*(H) - q^*(G')| + |q^*(H) - q^*(G)| \\ &< \frac{2(|E| - |E'|)}{|E|} + \frac{2(|E'| - |E''|)}{|E|} \\ &= \frac{2(|E| - |E''|)}{|E|} = \frac{2|E \setminus E'|}{|E|}. \end{aligned}$$

□

Note that the above results only show the stability of the *value* of modularity, not of the optimal partition.

2.3 Modularity in community detection

To be clear, modularity is perfectly fine as a mathematical concept. When we say 'shortcomings' or 'issues', we say that with regards to using modularity to understand community structures in networks. Even though the concept of community is ill-defined, there are some common, qualitative properties that most people agree it should have. 'Issues' arise when the behaviors of modularity clash with those intuitive qualities. Strictly speaking, that could also mean that modularity has unveiled something non-intuitive but still valuable. We sidestep those semantic discussions because they require subject matter context for each particular network, and focus instead on simple notions of community.

Optimizing modularity

Modularity was originally introduced in [34] to cut off the dendrogram in a divisive clustering method. Gradually, several methods focus on optimizing modularity itself, turning it into an objective function over the space of all partitions.

Finding a partition to optimize modularity, or even just to approximate it within a constant factor, is an NP-hard problem [35, 42]. Therefore, most optimization methods use heuristic or randomization. In [43], the authors provide evidence that modularity of real world networks have many local maxima close to the global maximum, and the structures at those maxima are quite different from each other. This means heuristic methods often succeed at finding a good modularity, but the partitions they produce are structurally fragile (i.e. unstable) and therefore hard to interpret.

The rugged landscape of modularity is generally considered to be an undesirable property. This is discussed in detail in [22], where modularity is compared with statistical (inferential) methods based on stochastic block models. The likelihood landscapes of block models also have several local maxima, but the models obtained at those points can be interpreted as competing hypotheses for the available data. On the other hand, modularity, a purely descriptive function, cannot provide such interpretation. See [22, Section 4B] for more detailed discussion.

Resolution limit

The modularity formula has a global parameter m - the number of edges in G. Adding more edges can change the optimal partition in some small corner that has no relationship with those edges at all. The most famous manifestation of this phenomenon is the *resolution limit* of modularity (see [44], [43, Section 2] for further discussion). Let $\mathcal{P} = \{P_1, P_2, \ldots\}$ be a partition of G. The change in modularity (new value minus old value) when we merge P_1 and P_2 into one community is

$$\Delta q = \frac{e(P_1, P_2)}{m} - \frac{\mathrm{vol}(P_1)\mathrm{vol}(P_2)}{2m^2}.$$

This change is positive when

$$e(P_1, P_2) > \frac{\mathrm{vol}(P_1)\mathrm{vol}(P_2)}{2m}. \tag{2.14}$$

The quantities $e(P_1, P_2)$, $\mathrm{vol}(P_1)$, and $\mathrm{vol}(P_2)$ depend only on the neighborhood structure surrounding P_1 and P_2, but m is a global quantity. If we add more edges in any far-flung corner of G, at some point the condition (2.14) will be satisfied as long as there are some edges joining P_1 and P_2. In an extreme case, if we embed G in some other enormous graph, then even without connecting G to the new graph, the new maximum modular partition will only split G into connected components. To stretch it a bit more, any connected graph collapses into a single community in a large enough context. This shows that modularity can underfit if there are communities at very different scales.

One way to understand this behavior is to look at the configuration model that motivates modularity. In that model, all vertices can connect with each other randomly, so if our community structure has some kind of locality, the model is not suitable.

There are several ways to address the resolution limit. One simple method is to add a *resolution parameter* γ to obtain a family of modularity function:

$$q_{\mathcal{P}}(G, \gamma) := q^E_{\mathcal{P}}(G) - \gamma q^D_{\mathcal{P}}(G) = \sum_{P \in \mathcal{P}} \left[\frac{e(P)}{m} - \gamma \left(\frac{\mathrm{vol}(P)}{2m} \right)^2 \right].$$

The parameter γ adjusts the relative influence of the edge contribution and the degree tax. Recalling the discussion in Section 2.1, maximizing edge contribution tends to produce a few large communities, while minimizing degree tax tends to produce many small, balanced groups. Therefore, a large γ is like a magnifying glass, allowing us to see smaller communities; of course a potential side-effect is that some large communities may be broken up into smaller ones.

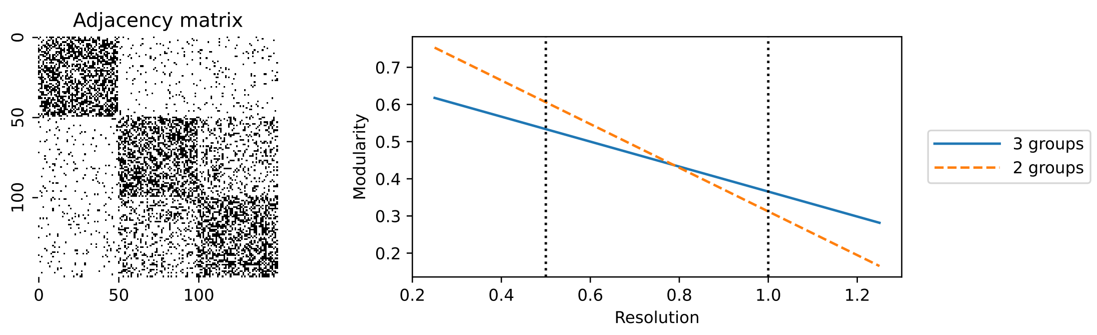


Figure 2.1: Effect of the resolution parameter. Group 1 is connected to the rest with density 0.05, while groups 2 and 3 are connected with density 0.20. The internal density of all three groups is 0.50.

Figure 2.1 illustrates the effect of the resolution parameter. The graph has two natural partitions, one consisting of three balanced groups, the other two unbalanced groups. If we set the resolution at 1.0 (i.e. standard modularity), the finer partition with 3 groups gives better value. On the other hand, setting resolution at 0.5 favors the coarse partition with just 2 groups.

This leaves the question of which resolution to choose. A single parameter may not be enough to detect communities at both ends of the scale (see [45]). We can choose suitable resolution using stability: if a partition gives good modularity over a long range of resolution, it is likely meaningful. The following result is a basis for this claim.

Proposition 2.18 ([46, Theorem 1])**.** *If the partition $\mathcal{P}$ is optimal for both $q(\gamma_1)$ and $q(\gamma_2)$ ($\gamma_1 \leq \gamma_2$), then it is also optimal for all $q(\gamma)$ with $\gamma_1 \leq \gamma \leq \gamma_2$.*

Proof. This follows easily from the fact that $q(\gamma)$ is a convex combination of $q(\gamma_1)$ and $q(\gamma_2)$. Since $\gamma_1 \leq \gamma \leq \gamma_2$, there is $0 \leq a \leq 1$ such that $\gamma = (1-a)\gamma_1 + a\gamma_2$. For any partition $\mathcal{Q}$,

$$\begin{aligned} q_{\mathcal{Q}}(\gamma) &= (1-a)q_{\mathcal{Q}}(\gamma_1) + aq_{\mathcal{Q}}(\gamma_2) \\ &\leq (1-a)q_{\mathcal{P}}(\gamma_1) + aq_{\mathcal{P}}(\gamma_2) = q_{\mathcal{P}}(\gamma). \end{aligned} \qquad \square$$

See [46] for additional results.

Interpreting high modularity

Graphs with higher modularity should have clearer community structures, but we do not know how high should modularity be for the structure to be meaningful. We have seen that cycles have modularity near 1, and there are many other graphs which are very symmetric and still have high modularity (see [37] and the citations therein).

Since graphs with no community structure can still have modularity near 1, we need principled methods to determine if the obtained modularity is indeed high and therefore indicative of the graph having community structure. One simple way to produce a baseline value is by randomization, similar to statistical testing. We choose a random graph model that preserves some aspects of the network at hand but randomizes other, then produce a lot of random samples to obtain a good approximation to the distribution of modularity in that model. The modularity of the given network can then be placed in the distribution to produce a 'p-value'.

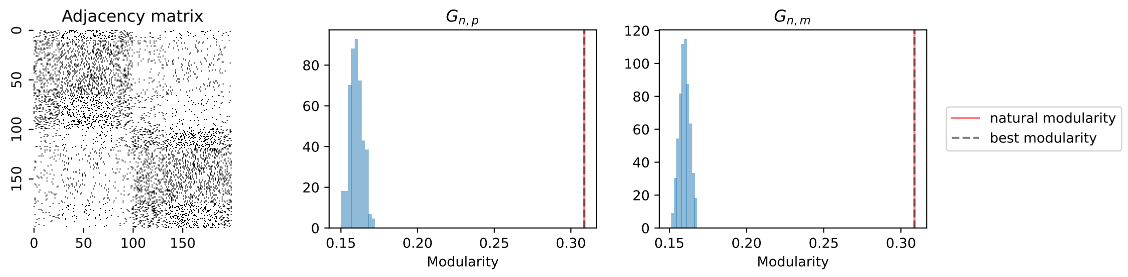


Figure 2.2: Significance of modularity on a graph with two balanced groups. The internal density is 0.20, and the external density is 0.05. Each histograms is generated from 200 samples.

Figure 2.2 shows a simple example. We produce a graph G with two balanced groups; set n to be the number of vertices, m the number of edges, and p the edge density (i.e. $p = m/\binom{n}{2}$). Consider two familiar random graph ensembles: $\mathcal{G}_{n,p}$ where each edge appears with probability p, and $\mathcal{G}_{n,m}$ where m edges are placed randomly. For each model we generate 200 samples. The maximum modularity of each graph is approximated using 3 runs of the Louvain method ([47], as implemented in NetworkX library). (We choose such a low number to ensure reasonable running time; more accurate experiments requires the number of runs at least in the dozens.) In this case the planted partition is likely the best one, and the histograms show that the modularity is indeed very high, with p-value near zero.

However, as pointed out in [22, Section 4C], there are two very different interpretations of the p-value, with the first being much stronger than the second:

- it is the probability that the graph does not have community structure, or
- it is the probability that the graph is not generated by the model we are considering.

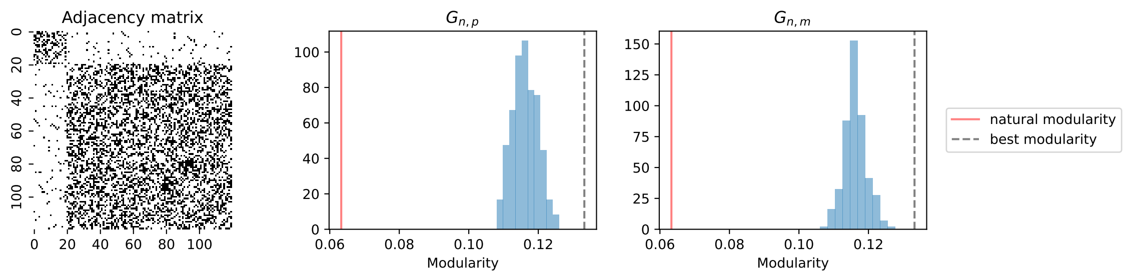


Figure 2.3: Significance of modularity on a graph with two unbalanced groups. The small group has size 20, while the big group has size 100. The internal density is 0.40, and the external density is 0.05. Each histogram is generated from 200 samples.

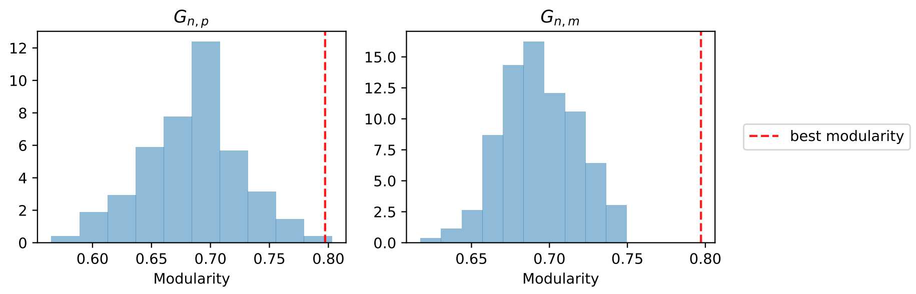


Figure 2.4: Significance of modularity on a cycle of size 100. Each histogram is generated from 200 samples.

Figure 2.3 shows a similar example, but this time the two groups have different sizes. The modularity is still very high (i.e. statistically significant), but this time the natural partition has *lower* modularity than the ensemble. In this case, the second, and weaker, conclusion is the more suitable one: our graph is not generated by $\mathcal{G}_{n,p}$ or $\mathcal{G}_{n,m}$. The same experiment, performed on cycles, are shown in Figure 2.4.

The models $\mathcal{G}_{n,p}$ and $\mathcal{G}_{n,m}$ are too simple baseline models for most real networks. A more realistic one, which is also the original motivation for modularity, is the configuration model. However, there are several configuration models depending on whether we allow self-loops and parallel edges and whether the stubs are labeled. Crucially, the choice has to be made in the context of the data, involving subject matter knowledge. See [48] for a detailed introduction to configuration models (modularity is discussed in Section 5 of that paper).

Chapter 3

RANDOM WALKS IN COMMUNITY DETECTION

This chapter studies the spectral properties of the random walk matrix and a clustering algorithm based on those properties. Section 3.1 introduces the random walk matrix and its spectrum. Section 3.2 explains why the top eigenvectors of that matrix inherit the clustering structure of the graph and illustrates the phenomenon visually. Section 3.3 presents the Walktrap algorithm and performs experiments to test the effect of step size and linkage method.

In this chapter we only work with connected and non-bipartite graphs; the reasons are given in Section 3.1 below.

3.1 Random walks and stochastic matrices

Let G be a connected simple graph (having no self-loops and no parallel edges) with vertex set $\{1, 2, \ldots, n\}$. A simple *random walk* on G starting from vertex v is a sequence of random variables $X_0, X_1, X_2, \ldots$, where $X_0 = v$, and X_{i+1} is picked uniformly randomly from the neighbors of X_i. More generally, X_0 can be picked from the vertices of G according to some distribution on the vertex set V. The sequence (X_i) forms a Markov chain with transition probability from vertex u to vertex v:

$$p_{uv} := \mathbb{P}\{X_{i+1} = v | X_i = u\} = \frac{1}{\deg(u)},$$

and we have the transition probability matrix

$$P = D^{-1}A,$$

where A is the adjacency matrix of G, and D is the degree diagonal matrix ($d_{uu} = \deg(u)$). We call P the *random walk matrix* of G. The matrix P is a *stochastic matrix*, i.e. a matrix with nonnegative entries and row sums equal to 1. For a positive integer t, the matrix power P^t contains the transition probabilities after performing t steps of the random walk.

In this chapter, we take the more concrete perspective of matrix analysis.

Recall some properties of stochastic matrices from [49, Chapter 8]. The matrix P has spectral radius 1, and 1 is also an eigenvalue of P with eigenvector $\mathbf{1}$. Since G is connected, P is *irreducible*, and the top eigenvalue 1 is algebraically simple.

The basic phenomenon we try to understand is the convergence of the random walk to a stationary distribution. More specifically, there are three main questions:

1. Does the random walk have a stationary distribution?
2. Does the random walk converge to a stationary distribution?
3. How fast does the random walk converge?

The first question is quite easy. Simple calculations show that the walk has a stationary distribution π, where

$$\pi(u) = \frac{\deg(u)}{2m}.$$

The row vector $\pi^\top$ is a left eigenvector of P with eigenvalue 1, so by the Perron-Frobenius theorem ([49, Theorem 8.4.4]) it is the unique stationary distribution of the walk.

For the second question, we need the concept *ergodicity* (also called *primitivity* in the context of matrices, see [49, Section 8.5]). For each vertex u, we list all the walks that start and end at u (those walks can visit u or any vertex multiple times), then let $\mathrm{period}(u)$ be the greatest common divisor of the lengths of those walks. The random walk on G is said to be ergodic if $\gcd\{\mathrm{period}(u) : u \in V\} = 1$. There are actually only two cases to consider. Each vertex u has a loop-walk of length 2 (just take $u - v - u$ for some neighbor v of u), so the period of u is either 1 or 2. If the walk on G is not ergodic, it means that each vertex has period 2, therefore G has no walks of odd length. By a standard result in graph theory, that implies G is a bipartite graph.

Bipartite graphs, also called two-mode networks or affiliation networks, form a special class of graphs. They usually contains nodes of two different types, like author-paper or actor-movie, and community structure is therefore very different from that in non-bipartite (or one-mode) networks (see [50] for an introduction). In this chapter, we assume *all graphs are connected and non-bipartite*, therefore all random walks are ergodic.

Since P is ergodic, 1 is the only eigenvalue on the spectral circle. In particular, all other eigenvalues have norm strictly less than 1. This shows that as $t \to \infty$, P^t converges to the rank-one matrix $\mathbf{1}\pi^\top$ with speed $O(|\lambda|^t)$, where λ is the eigenvalue with the second largest norm (see Theorem 8.5.1 and the decomposition in Theorem 1.4.7 in [49]).

Random walks on graphs have special properties that allow us to be more concrete in describing the spectrum of P. The random walk matrix P is similar to a symmetric matrix:

$$D^{\frac{1}{2}} P D^{-\frac{1}{2}} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}.$$

This shows that P actually has real spectrum. The matrix $L = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$ is called the *Laplacian* of G, which has nonnegative spectrum. For our purpose it suffices to work directly with P and A.

Proposition 3.1 ([51, Lemma 1]). *The matrix P has n real eigenvalues satisfying:*

$$1 = \lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n > -1.$$

Moreover, there is an orthonormal family of real vectors $(s_i)_{1 \leq i \leq n}$ such that for each i, the vectors $x_i := D^{-\frac{1}{2}} s_i$ and $y_i := D^{\frac{1}{2}} s_i$ are respectively the right and left eigenvectors of P corresponding with the eigenvalue λ_i. In particular,

$$P x_i = \lambda_i x_i, \quad y_i^\top P = \lambda_i y_i^\top,$$

and

$$y_i^\top x_j = s_i^\top s_j = \delta_{ij},$$

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

Proof. The spectrum has been established by the previous discussion. For the second part, take (s_i) to be an orthonormal set of eigenvectors of the symmetric matrix $D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$. A simple change of basis gives us the desired conclusions. $\square$

Remark 3.2. Let Λ be the diagonal matrix with diagonal entries $\lambda_1, \ldots, \lambda_n$, and let X and Y be square matrices containing (x_i) and (y_i) as columns, respectively. Orthogonality gives us $Y^\top X = I$, and so $XY^\top = I$ by uniqueness of matrix inversion. The columns of X are right eigenvectors of P, therefore

$$PX = \Lambda X.$$

Multiplying both side on the right with $Y^\top$, we obtain a useful decomposition:

$$P = \Lambda X Y^\top = \sum_{i=1}^{n} \lambda_i x_i y_i^\top. \tag{3.1}$$

From this decomposition, matrix powers of P are represented as

$$P^t = \sum_{i=1}^{n} \lambda_i^t x_i y_i^\top. \tag{3.2}$$

Remark 3.3. By the Perron-Frobenius theorem, x_1 is a scalar multiple of $\mathbf{1}$ and y_1 is a scalar multiple of π. Note that

$$1 = s_1^\top s_1 = x_1^\top D x_1 = y_1^\top D^{-1} y_1.$$

Therefore by suitable scaling, we can choose x_1 and y_1 such that

$$x_1(u) = \frac{1}{\sqrt{\sum_v \deg(v)}}, \quad y_1(u) = \frac{\deg(u)}{\sqrt{\sum_v \deg(v)}}.$$

3.2 Spectral clustering

Spectral clustering is a traditional subject with huge literature; see [52, 53, 54] for some comprehensive surveys. Here we only present the main idea and some numerical illustrations.

Consider an ideal case, where the graph consists of two disjoint connected communities (the arguments easily generalize to the case of k groups). We can rearrange the vertices so that the random walk matrix has, say, the following form:

$$P = \begin{bmatrix} * & * & * & * & 0 & 0 \\ * & * & * & * & 0 & 0 \\ * & * & * & * & 0 & 0 \\ * & * & * & * & 0 & 0 \\ 0 & 0 & 0 & 0 & * & * \\ 0 & 0 & 0 & 0 & * & * \end{bmatrix}.$$

Then the top eigenvalue 1 has dimension 2, and all other eigenvalues have norm strictly less than 1. We therefore have a gap between the top two eigenvalues and the rest.

The eigenvectors corresponding to eigenvalue 1 are

$$\begin{bmatrix}1\\1\\1\\1\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\0\\0\\1\\1\end{bmatrix}.$$

If we combine those two column vectors into a matrix and use the rows to represent the corresponding vertices, the two original communities are easily separated.

If we perturb the edge set a little, we also perturb the adjacency matrix A a little. By eigenvalue perturbation theorems [49, Section 6.5], the spectrum of the symmetric matrix $D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$, which is also the spectrum of P, remains close to that of the original matrix. So we still have a gap between the top two eigenvalues and the rest. We also expect that the top two eigenvectors remain close to the original form, so we can use them to separate the two communities in the graph.

Note that the above argument is only an heuristic. Whether the eigengap remains depends on the original gap as well as how large the perturbation is. Moreover, while the spectrum is stable, the eigenvectors are not (see the last exercise in [49, Section 6.3]). The 'correct' argument uses stability of eigenspaces instead, see [52, Section 7].

To sum up, if the graph G has k well-separated communities, we expect the random walk matrix P has the following properties:

- there is a noticeable gap between λ_k and λ_{k+1}, and
- the $k-1$ eigenvectors $x_2, \dots, x_k$ can separate the k groups apart.

We do not need x_1 because it is just a constant vector. Now we can formulate the generic spectral clustering algorithm (using the notations of Proposition 3.1).

Generic spectral clustering

Input: A graph G and the number of communities k.

Output: A clustering of G into k disjoint groups.

1. Calculate the random walk matrix P and the $k-1$ eigenvectors $x_2, \dots, x_{k-1}$ corresponding to the eigenvalues $\lambda_2, \dots, \lambda_k$.
2. Embed each vertex of G into $\mathbb{R}^{k-1}$ using the coordinates of the eigenvectors:
$$u \mapsto [x_2(u), \dots, x_k(u)]^\top.$$
3. Apply a clustering algorithm for data in euclidean space.

There are many variants of the algorithm above: which kind of matrix to use, how to normalize the eigenvectors, how to normalize the embedding, which clustering algorithm to use, ...

It could be instructive to look at the spectrum of near-bipartite graphs, even though they are not the focus of this thesis. A bipartite graph has eigenvalue -1, corresponding to an eigenvector with all entries 1 on a partite set and all entries -1 on the other partite set. Therefore a near-bipartite graph will have the bottom eigenvalue λ_n close to -1, and we also expect that the bottom eigenvector can separate the two partite sets. The bottom gap $|\lambda_n + 1|$ can be used to quantify how close our graph is to a bipartite graph, see [55].

Numerical illustrations

We illustrate spectral clustering on small random graphs generated by stochastic block models, varying the group sizes, number of groups, and densities. Some notable results are recorded in Figures 3.1 to 3.5, with detailed parameter settings in the captions. Calculations are done using the random walk matrices, and all eigenvectors are normalized to have l_2-norm 1. Note that all indexing starts from zero.

Figure 3.1 (page 44) presents an ideal case with two groups of the same size and density. As expected, there is a noticeable gap between the second and third eigenvalues, and the second eigenvector shows a clear clustering into two groups. The other eigenvectors are not of much use. The same phenomena appear in Figure 3.2 (page 44), but this time we need both the second and third eigenvectors to separate groups 1 and 3.

Not all graphs have such clear spectral properties. Figure 3.3 (page 45) and Figure 3.4 (page 45) present two cases where the spectral gaps are harder to detect. They also show that smaller and sparser groups have more diffuse coordinates.

Figure 3.5 (page 52) shows that for near bipartite graphs, we have to use the bottom eigenvector instead.

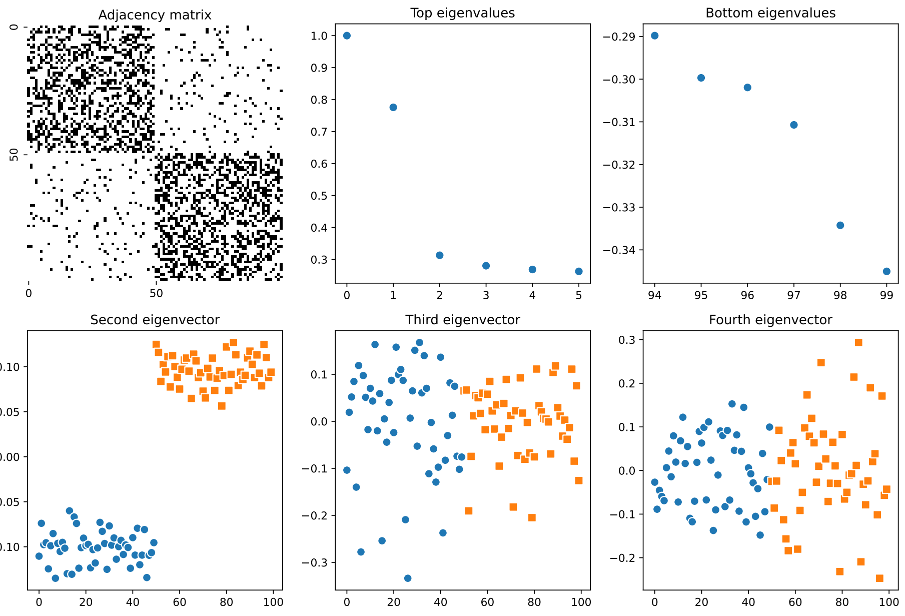


Figure 3.1: Spectral properties of a graph with two balanced groups. The internal density of both groups is 0.40, and the external density is 0.05.

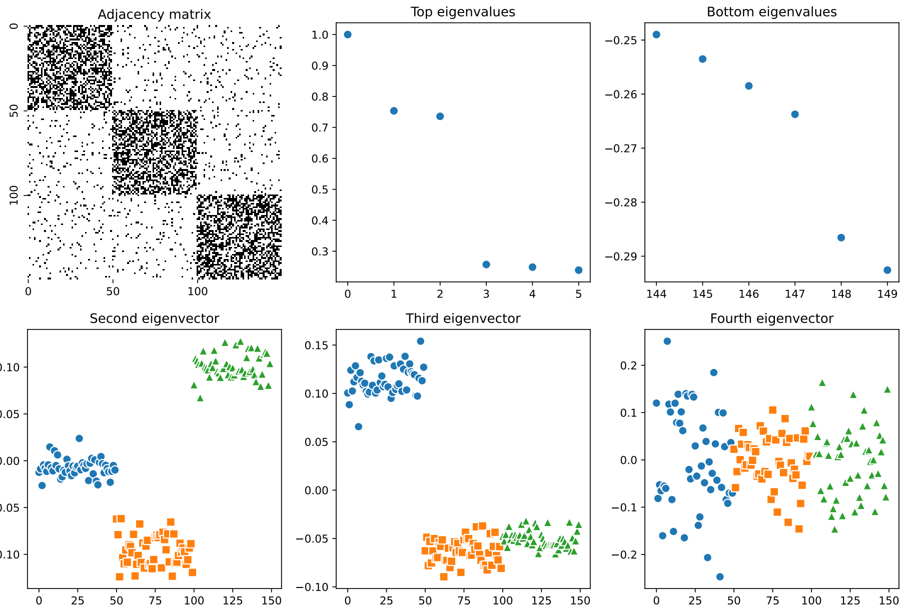


Figure 3.2: Spectral properties of a graph with three balanced groups. The internal density of all three groups is 0.50, and the external density is 0.05.

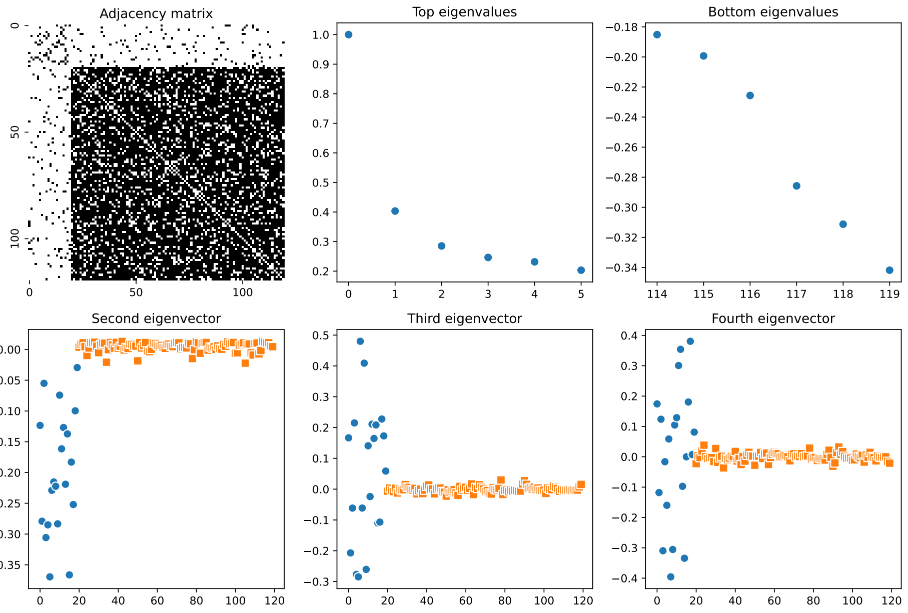


Figure 3.3: Spectral properties of a graph with two unbalanced groups. The small groups has size 20 and density 0.20, while the big group has size 100 and density 0.80. The external density is 0.05.

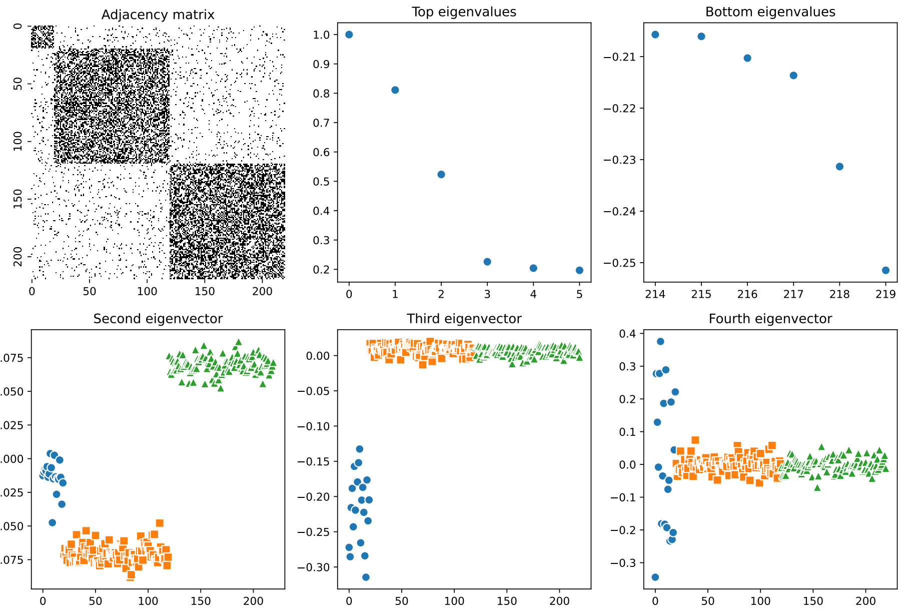


Figure 3.4: Spectral properties of a graph with three unbalanced groups of sizes 20, 100, and 100. All three groups have internal density 0.50, and external density 0.05.

3.3 The Walktrap algorithm

The Walktrap algorithm, proposed in [51], is one of the popular community detection algorithms. It has two basic components: the Walktrap metric, and an agglomerative clustering method. As the name suggests, the main motivation for the algorithm is that a random walker tends to spend more time in a community before moving to a different one. More concretely, random walkers starting at similar nodes will have similar 'views' of other vertices.

The Walktrap metric

Let t be a positive integer. Denote by P^t the t-power of the random walk matrix P, P^t_{ij} the (i,j)-entry of P^t, and $P^t_{u\bullet}$ row u of P^t (considered as a column vector). Recall that for a vector v, $v(i)$ denotes the i-th component of v.

Definition 3.4 ([51, Definition 1]). *For two vertices u and v and a positive integer t, we define the t-*step Walktrap metric *between u and v to be*

$$d^t_{\mathrm{WT}}(u,v) := \sqrt{\sum_{k=1}^{n} \frac{(P^t_{uk} - P^t_{vk})^2}{\deg(k)}} = \|D^{-\frac{1}{2}}P^t_{u\bullet} - D^{-\frac{1}{2}}P^t_{v\bullet}\|_2.$$

If t is already clear from context, we simply write $d_{\mathrm{WT}}(u,v)$, or even just $d(u,v)$.

Proposition 3.5 ([51, Theorem 1]). *Using the notations from Proposition 3.1 and Remark 3.3, we have the following formula for the t-step Walktrap metric:*

$$d(u,v)^2 = \sum_{i=2}^{n} \lambda_i^{2t}(x_i(u) - x_i(v))^2. \tag{3.3}$$

Proof. Using formula (3.2), we obtain the formula for rows of P^t:

$$P^t_{u\bullet} = \sum_{i=1}^{n} \lambda_i^t x_i(u) y_i.$$

Since $\{D^{-\frac{1}{2}}y_i\}$ form an orthonormal set of vectors, we easily obtain

$$\|D^{-\frac{1}{2}}P^t_{u\bullet} - D^{-\frac{1}{2}}P^t_{v\bullet}\|_2^2 = \sum_{i=1}^{n} \lambda_i^{2t}(x_i(u) - x_i(v))^2.$$

Recalling that x_1 is a constant vector, we can drop the index 1 in the sum. □

Remark 3.6. As noted in [51], this metric is the same as the *diffusion distance*, used in a data dimension reduction method [56, 57].

After fixing a step size t and computing the Walktrap matrix $W = P^t D^{-\frac{1}{2}}$, we have an embedding of G into $\mathbb{R}^n$, where each vertex is mapped to the corresponding row of W.

Formula (3.3) shows that the Walktrap distance can be considered as a smoothed version of spectral embedding, where eigenvectors with higher eigenvalues are prioritized. As demonstrated in Section 3.2, if our graph has clear community structure, those eigenvectors show clear clustering property, and the Walktrap distance - with a suitable step size - should inherit that.

Agglomerative clustering

An agglomerative clustering algorithm gradually collapses all data points into a single group, producing a hierarchy of nested partitions (visually represented by a dendrogram). The most crucial detail is which two groups should be merged at each step, and this is determined by the *linkage method*, which is the method to assign distance to two distinct clusters. See [58, Chapter 4] for a more thorough introduction.

Agglomerative clustering

Input: n data points in an euclidean space, and a linkage method.

Output: A sequence of n nested partitions (i.e. a dendrogram).

1. Form n clusters, each containing a single data point.
2. As long as there are more than one clusters, merge two clusters with the minimum distance. (The distance is determined by the linkage method).

There are four linkages that we will test. We briefly mention their definitions and how to update the distances after each merge.

- *Single linkage:* the distance between two clusters is the minimum possible distance between a pair of points.

$$d(A, B) = \min_{u \in A,\, v \in B} d(u, v).$$

  Updating formula:

$$d(A \cup B, C) = \min\{d(A, B), d(A, C)\}.$$

- *Complete linkage:* the distance between two clusters is the maximum possible distance between a pair of points.

$$d(A, B) = \max_{u \in A,\, v \in B} d(u, v).$$

Updating formula:

$$d(A \cup B, C) = \max\{d(A, B), d(A, C)\}.$$

- *Average linkage:* The distance between two clusters is the average distance between pairs of points.

  $$d(A, B) = \frac{1}{|A||B|} \sum_{u \in A,\, v \in B} d(u, v).$$

  Updating formula:

  $$d(A \cup B, C) = \frac{|A|}{|A| + |B|} \cdot d(A, C) + \frac{|B|}{|A| + |B|} \cdot d(B, C).$$

- *Ward linkage:* The distance between two clusters is the increase of the within-class sum of squared errors if we merge those two. The error of a point is the distance from that point to the centroid of its cluster, where the centroid is defined as

  $$m_P := \frac{1}{|P|} \sum_{u \in P} u.$$

  The sum of squared errors of a partition $\mathcal{P}$ is

  $$\mathcal{E}(\mathcal{P}) := \sum_{P \in \mathcal{P}} \sum_{u \in P} \|u - m_P\|^2.$$

  The new centroid when we merge two clusters is

  $$m_{A \cup B} = \frac{|A| \cdot m_A + |B| \cdot m_B}{|A| + |B|}.$$

  A simple calculation gives the distance

  $$d(A, B) = \frac{|A||B|}{|A| + |B|} \cdot \|m_A - m_B\|^2.$$

  The updating formula is a bit complicated, so we derive it in detail. Let A, B, C be three disjoint sets. The required formula should represent $d(A \cup B, C)$ using $d(A, C)$, $d(B, C)$, and $d(A, B)$. For convenience, set $a = |A|$, $b = |B|$, $c = |C|$. The new distance is

  $$\begin{aligned} d(A \cup B, C) &= \frac{(a + b)c}{a + b + c} \cdot \|m_{A \cup B} - m_C\|^2 \\ &= \frac{(a + b)c}{a + b + c} \cdot \left\| \frac{am_A + bm_B}{a + b} - m_C \right\|^2. \end{aligned}$$

  which should be representable in the form

  $$x \cdot \|m_A - m_C\|^2 + y \cdot \|m_B - m_C\|^2 + z \cdot \|m_A - m_B\|^2,$$

where x, y, z are some real numbers to be found. Expanding both formulas and identifying the coefficients of each term m_A^2, m_B^2, m_C^2, $m_A \cdot m_B$, $m_A \cdot m_C$, $m_B \cdot m_C$, we easily obtain

$$x = \frac{ac}{a+b+c}, \quad y = \frac{bc}{a+b+c}, \quad z = -\frac{abc}{(a+b)(a+b+c)}.$$

Therefore the updating formula for Ward linkage is

$$d(A \cup B, C) = \frac{a+c}{a+b+c} \cdot d(A,C) + \frac{b+c}{a+b+c} \cdot d(B,C) - \frac{c}{a+b+c} \cdot d(A,B).$$

Figure 3.6 (page 53) illustrates how Ward and single linkage create dendrograms on the rows of the Walktrap matrix. In this simple case, cutting off the dendrograms at two groups recovers exactly the original groups.

Choosing linkage and step size

We test four linkages above combined with three step sizes (2, 5, and 8) on stochastic block models, varying the group sizes, number of groups, densities, and number of nodes. Some results (together with detailed parameter setting) are recorded in Figures 3.7 to 3.14. We describe how the figures are produced.

- In each figure, only one parameter of the stochastic block model is varied, the rest are fixed. All four linkages and three step sizes are tested.
- The planted groups in the block model are considered to be 'ground truth'. For each linkage and step size, we cut off the dendrogram using the known number of groups. The quality of the clustering is measured by the adjusted Rand index against the ground truth.
- For each fixed set of parameters, 20 random graphs are generated, then all twelve methods are performed on those 20 graphs. The score for each method is then averaged over those 20 samples.

The choice of adjusted Rand index for all experiments is mostly out of convenience. A score of 1 means perfect agreement, while a score near 0 or negative means bad quality. For the index's properties and shortcomings, see e.g. [59].

We do not add self-loops to vertices, and do not impose connectivity constraints. In other words, the rows of the Walktrap matrix are treated as normal data points in euclidean space.

Discussion of results. We make the following observations based on the results.

- Ward linkage provides the best performance and should be used in all cases, while single linkage is often the worst.

- The step sizes between 2 and 5 are reasonable choices; sparser graphs benefit from longer steps (i.e. 5) (see Figure 3.14 and the portions with lower densities in earlier figures).
- All methods' performances improve as densities or graph sizes increase, which provide evidence for the (asymptotic) consistency of Walktrap.

The authors of [51] observe empirically that step sizes between 3 and 8 (inclusive) give the best result and recommend choosing $t = 4$ or $t = 5$. For some theoretical results related to step size in Walktrap, see [60].

Practical implementation

After calculating the distances and choosing the linkage, there are two more technical details in the implementation of Walktrap.

- We need a way to cut off the dendrogram if the number of groups is not known beforehand. Modularity is a popular way to do this, similar to its original use in [34]. See Chapter 2 of this thesis for basic properties of modularity as well as its limitations.
- The algorithm in [51] only merges adjacent communities at each step to ensure connected communities and reduce computation. The authors also propose additional heuristics to reduce the number of distances computed; see the original paper for more information.

There is a ready implementation from the authors of the original paper[1], conveniently packaged in the igraph library[2]. Performance of Walktrap has been analyzed carefully in [26], so we do not perform any practical analysis in this thesis.

For completeness, a simple Python implementation is given in Appendix A, using popular open-source Python packages: NetworkX[3] for graph manipulation, and NumPy[4] for matrix calculation. The groups to merge and the modularity at each step are stored. The final partition can be produced by giving the number of communities, or cutting off the dendrogram with modularity (with resolution as an optional parameter). All methods are wrapped in a class. This is just a proof-of-concept program, not a practical one. In particular, we calculate distance using vectors directly rather than efficient updating formulas, and communities are merged by collapsing a new graph, which is rather expensive. We also do not add self-loops to vertices. The program aims

[1]`http://psl.pons.free.fr/index.php?item=prog&item2=walktrap&lang=en`
[2]`https://igraph.org`
[3]`https://networkx.org`
[4]`https://numpy.org`

to illustrate two techniques: how to extract the minimum distance using a heap, and how to keep track of modularity using a weighted graph. Hopefully copious comments in the code are enough for readers to follow the steps.

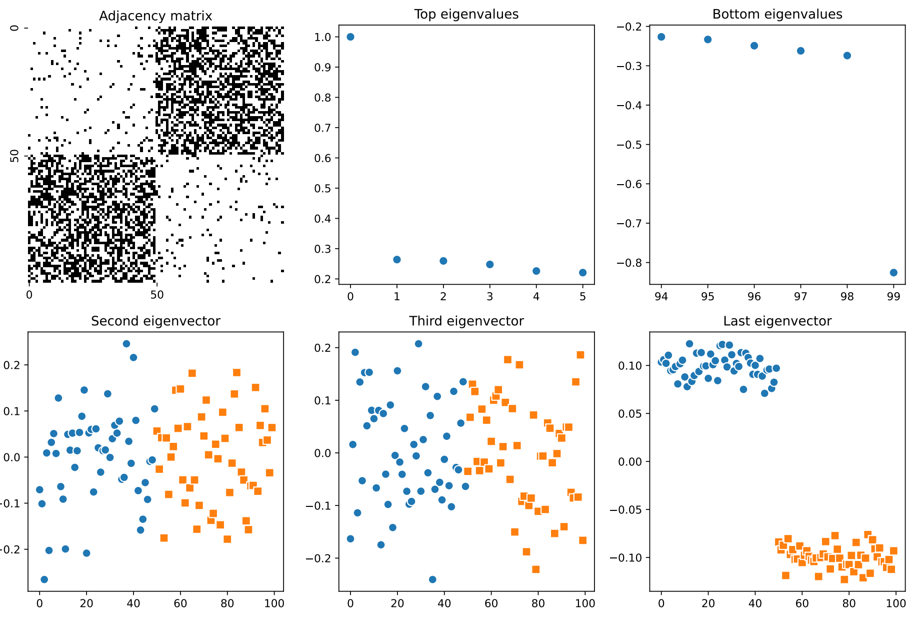
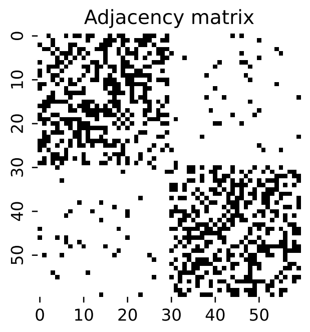
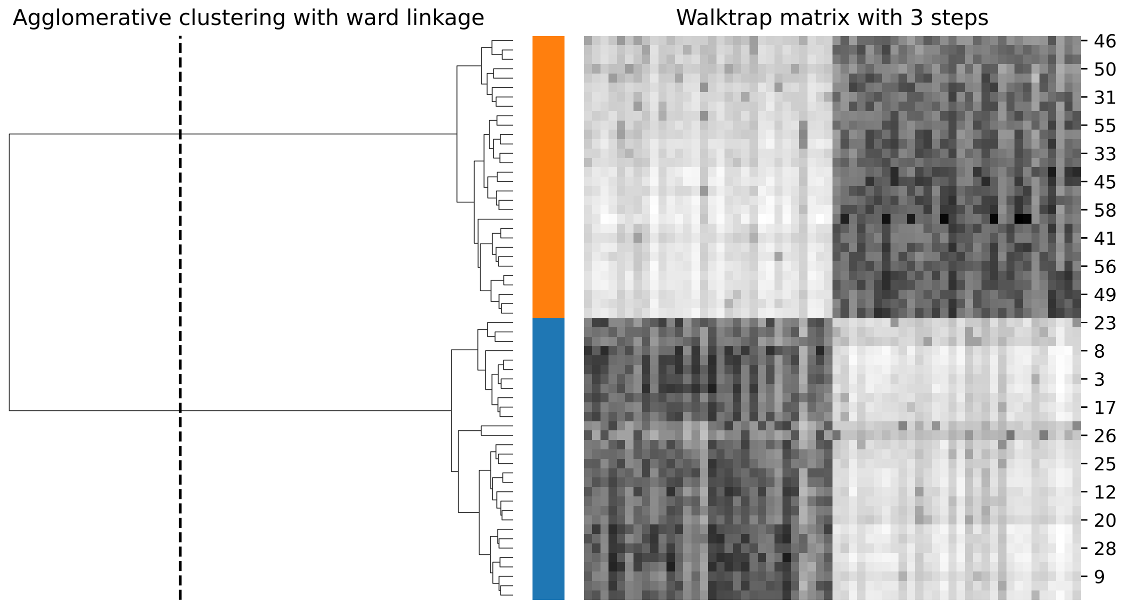


Figure 3.5: Spectral properties of a near-bipartite graph. The internal density of both groups is 0.05, while the external density is 0.50.





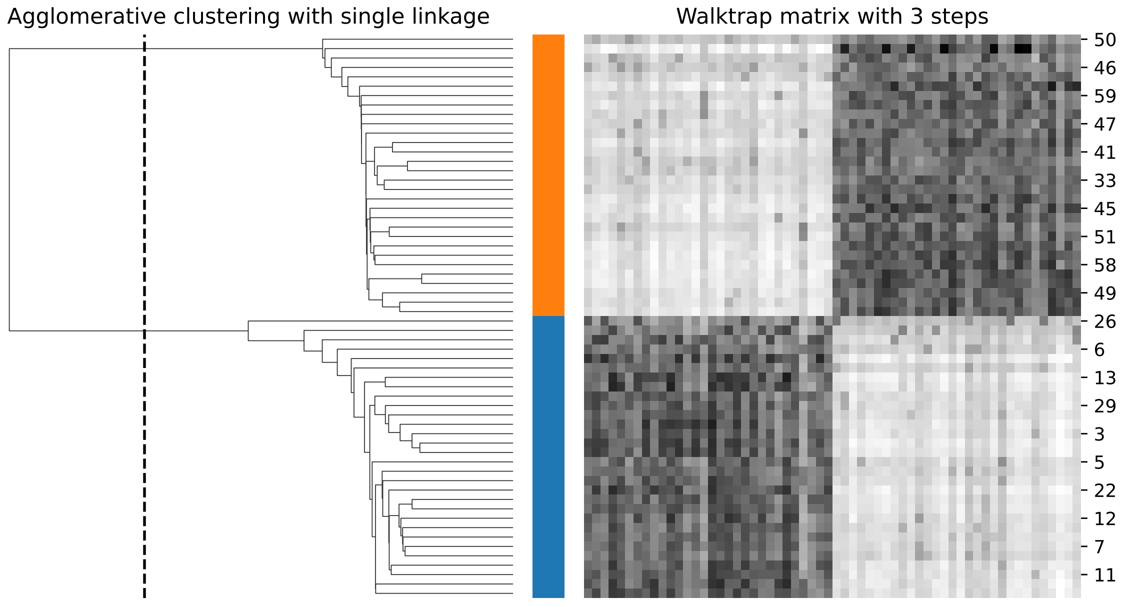


Figure 3.6: Illustration of Ward and single linkage agglomerative clustering on a Walktrap matrix. The graph has two balanced groups of size 30. The external density is 0.05, and the internal density of both groups is 0.4. The Walktrap matrix is calculated using 3 steps.

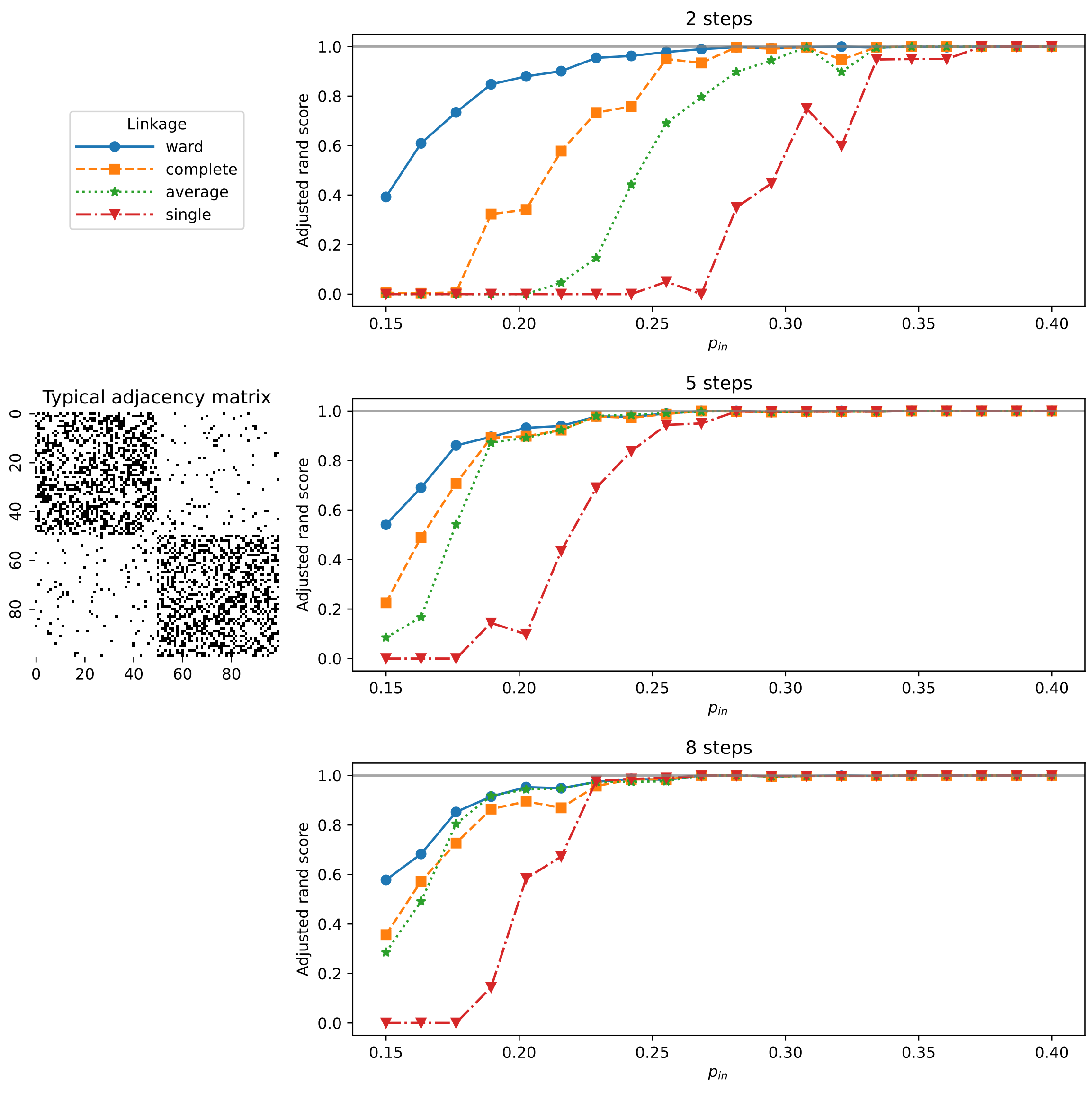


Figure 3.7: Testing Walktrap on graphs with two balanced groups. The external density is fixed at 0.05, while the internal density of both groups varies from 0.15 to 0.40.

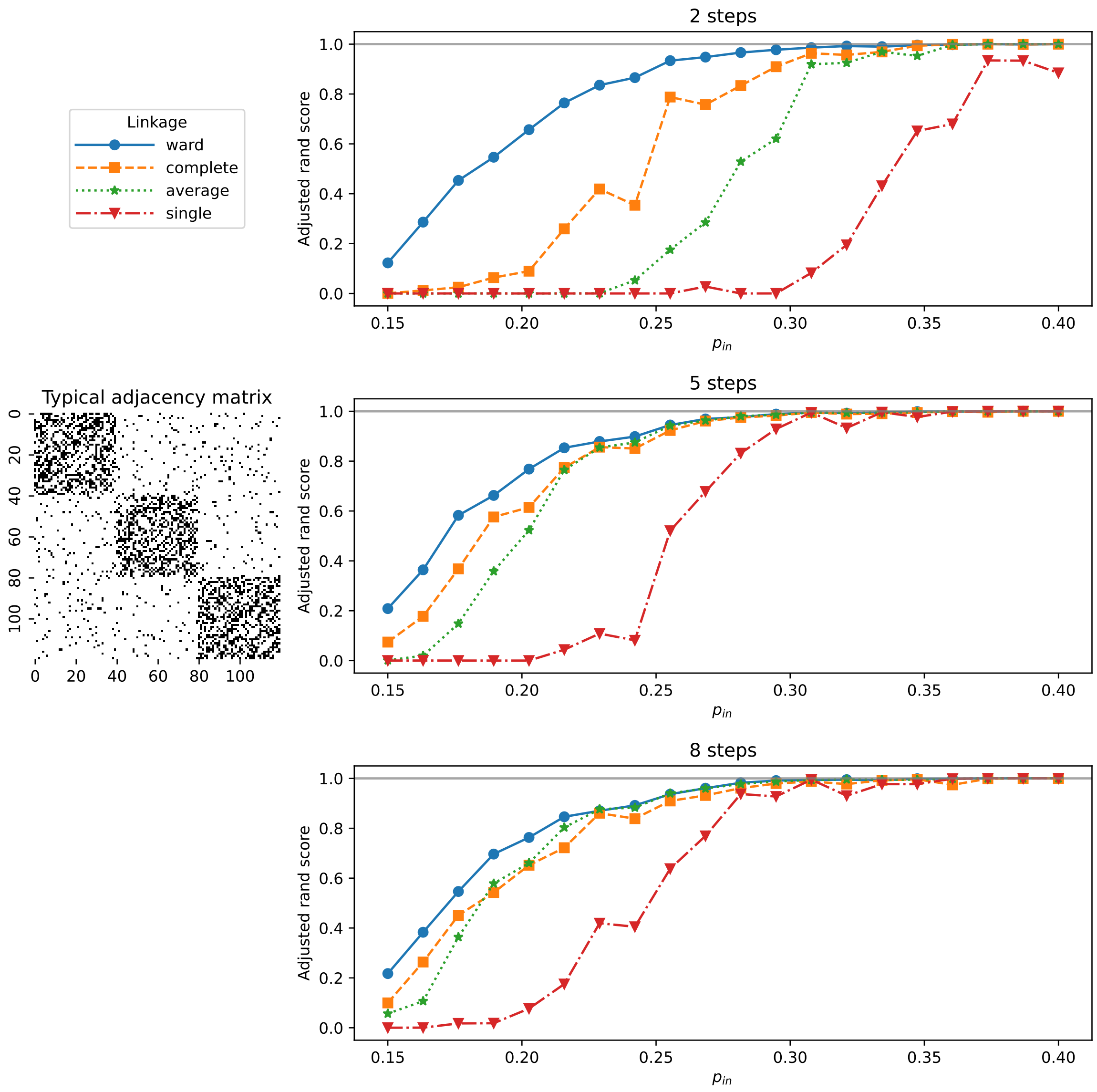


Figure 3.8: Testing Walktrap on graphs with three balanced groups. The external density is fixed at 0.05, while the internal density of all three groups varies from 0.15 to 0.40.

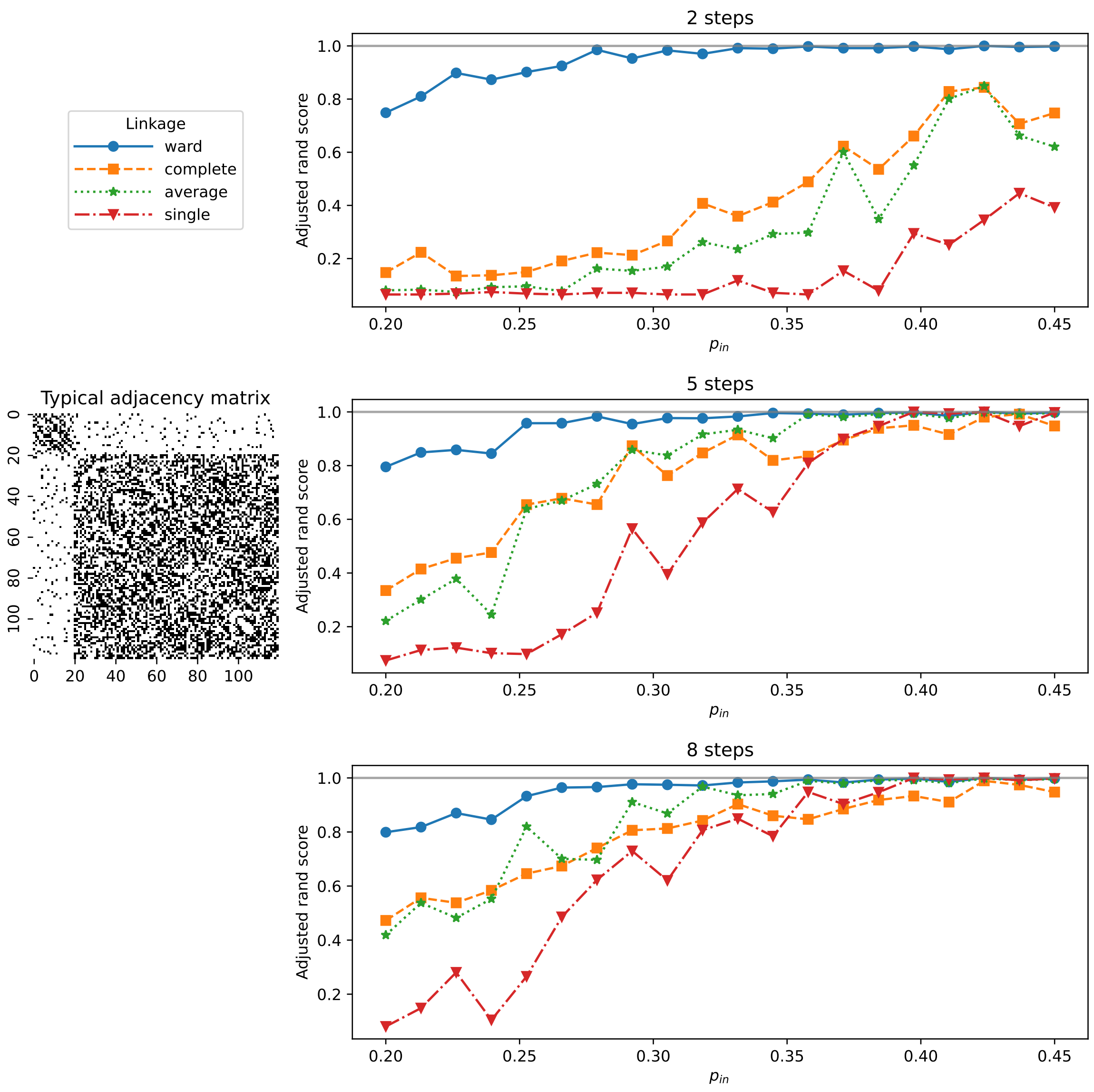


Figure 3.9: Testing Walktrap on graphs with two unbalanced groups. The small group has size 20, while the large group has size 100. The external density is fixed at 0.05, while the internal density of both groups varies from 0.20 to 0.45.

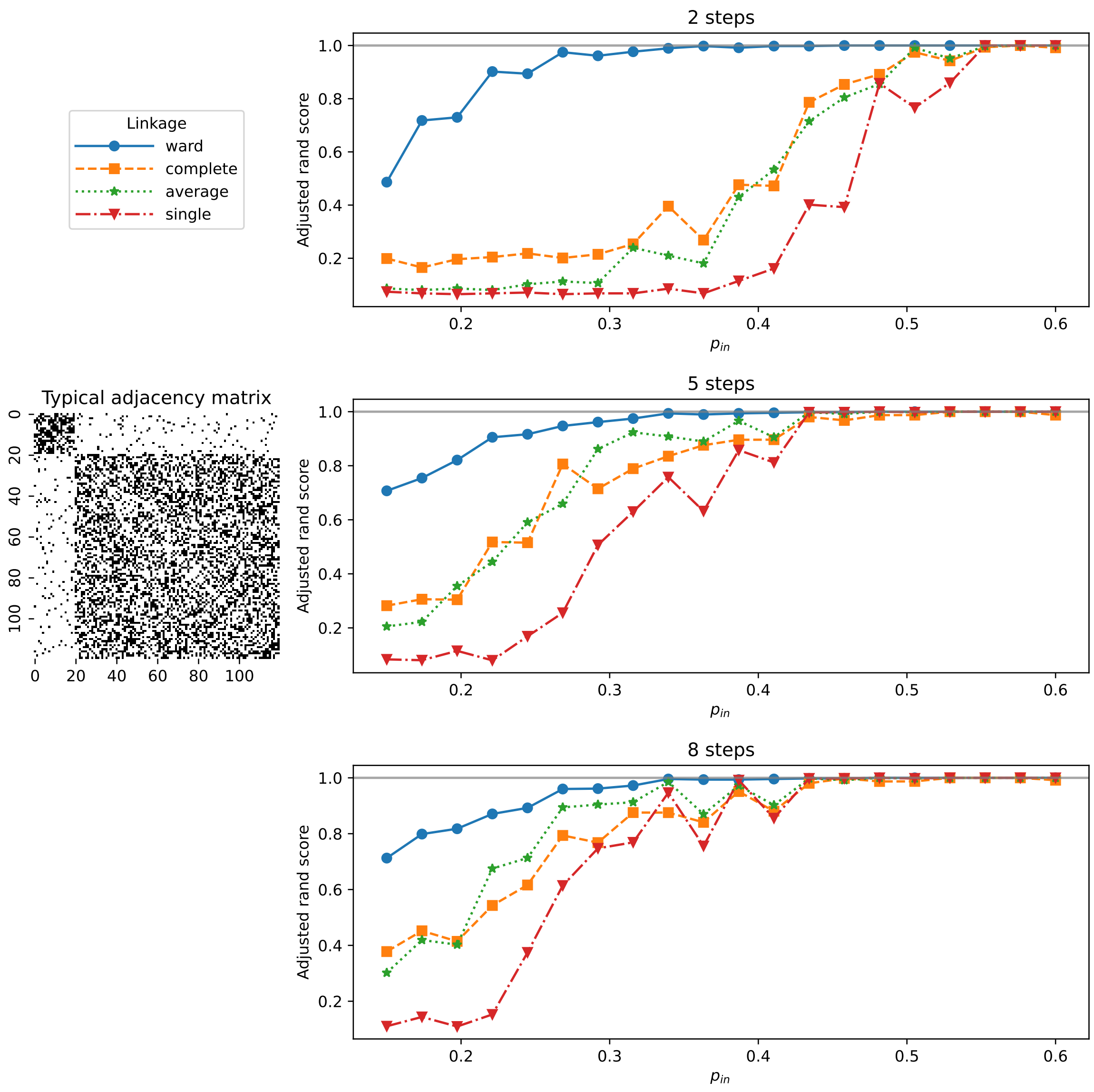


Figure 3.10: Testing Walktrap on graphs with two unbalanced groups with different densities. The small group has size 20, while the big group has size 100. The external density is fixed at 0.05, and the internal density of the big group is fixed at 0.4. The internal density of the small group varies from 0.15 to 0.60.

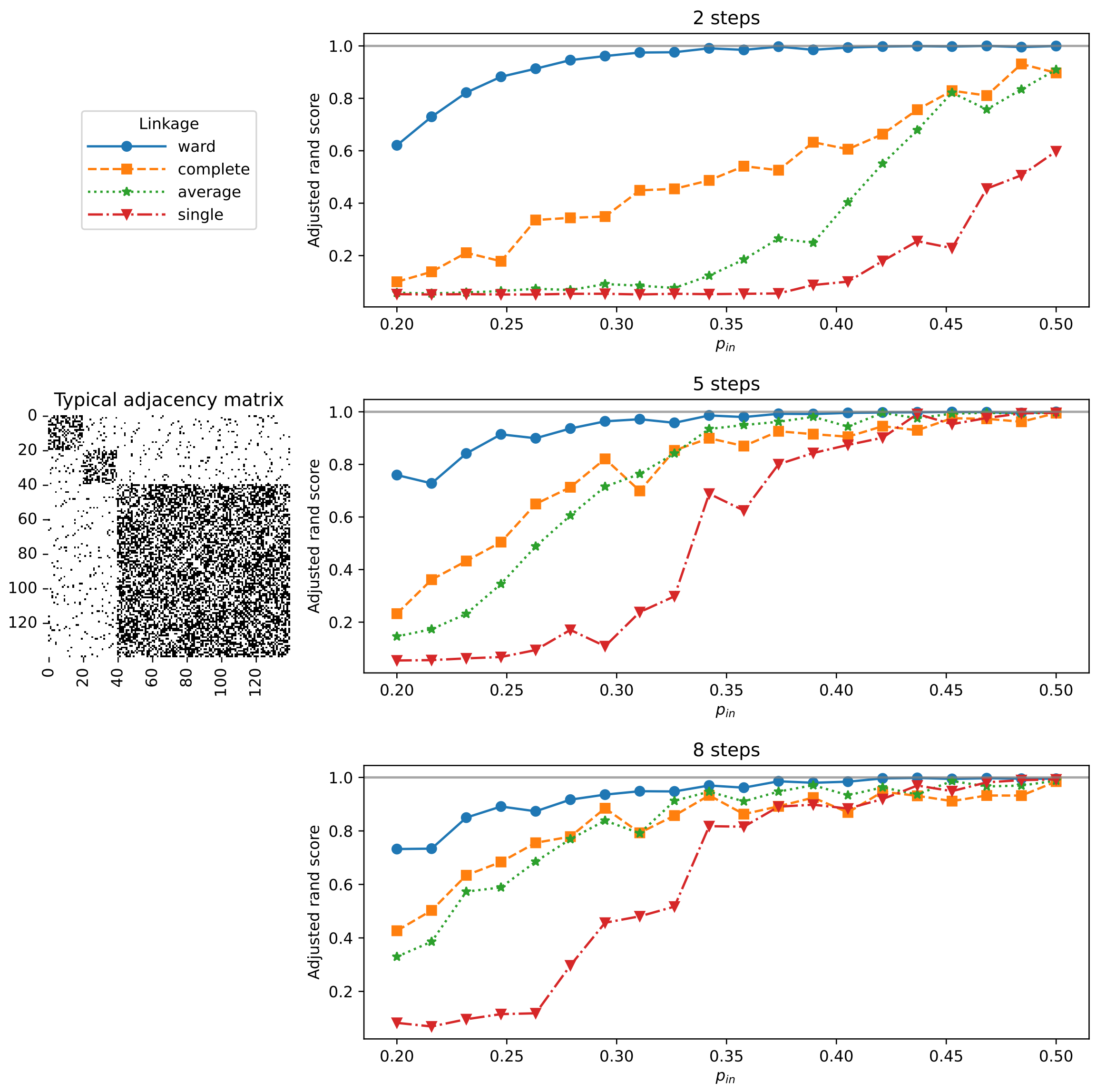


Figure 3.11: Testing Walktrap on graphs with three unbalanced groups. The two small groups have the same size 20, while the big group has size 100. The external density is fixed at 0.05, while the internal density of all three groups varies from 0.20 to 0.50.

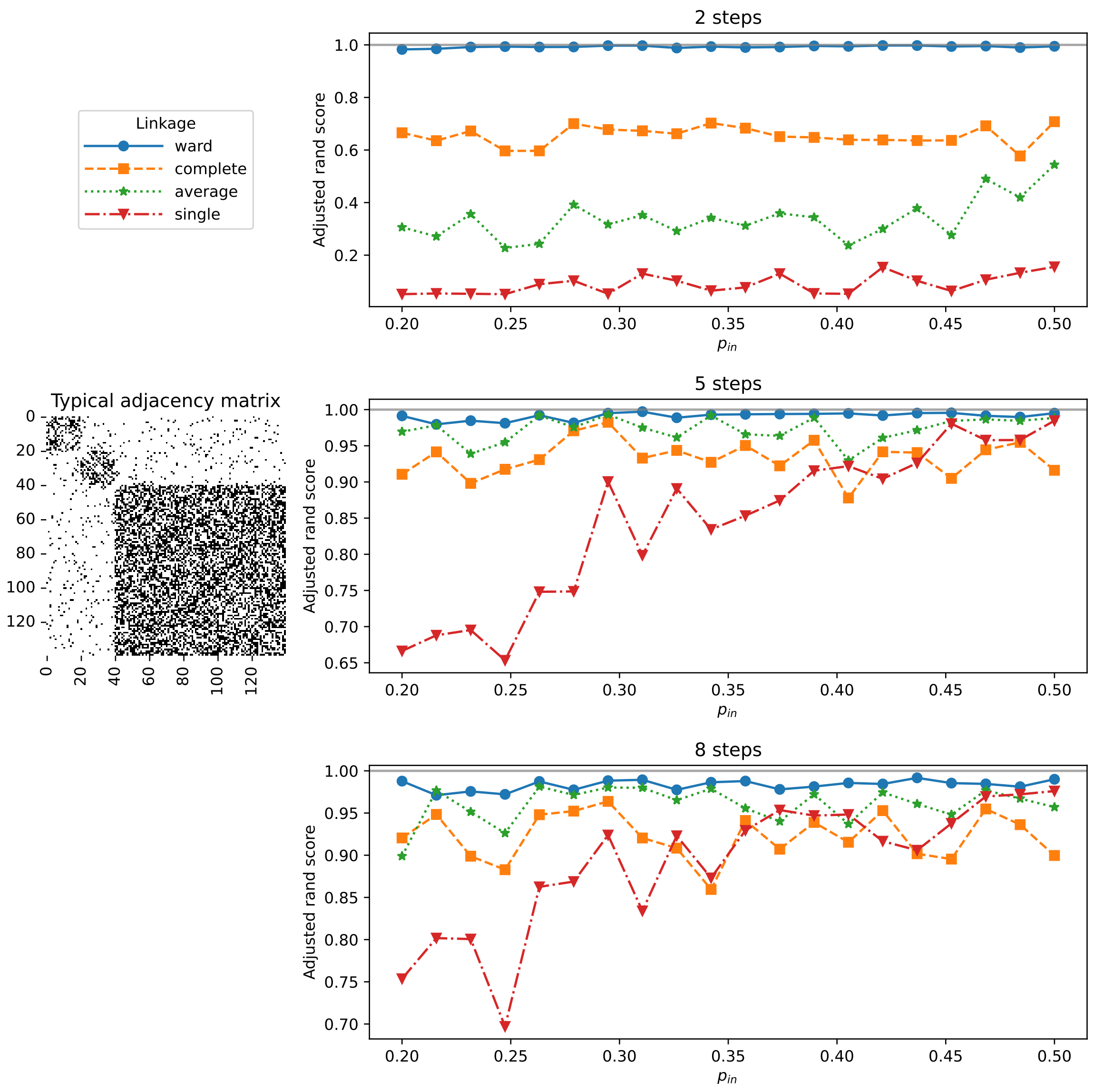


Figure 3.12: Testing Walktrap on graphs with three unbalanced groups with different densities. The two small groups have size 20, while the big group has size 100. The external density is fixed at 0.05, and the internal density of the two small groups is fixed at 0.4. The internal density of the big group varies from 0.20 to 0.50.

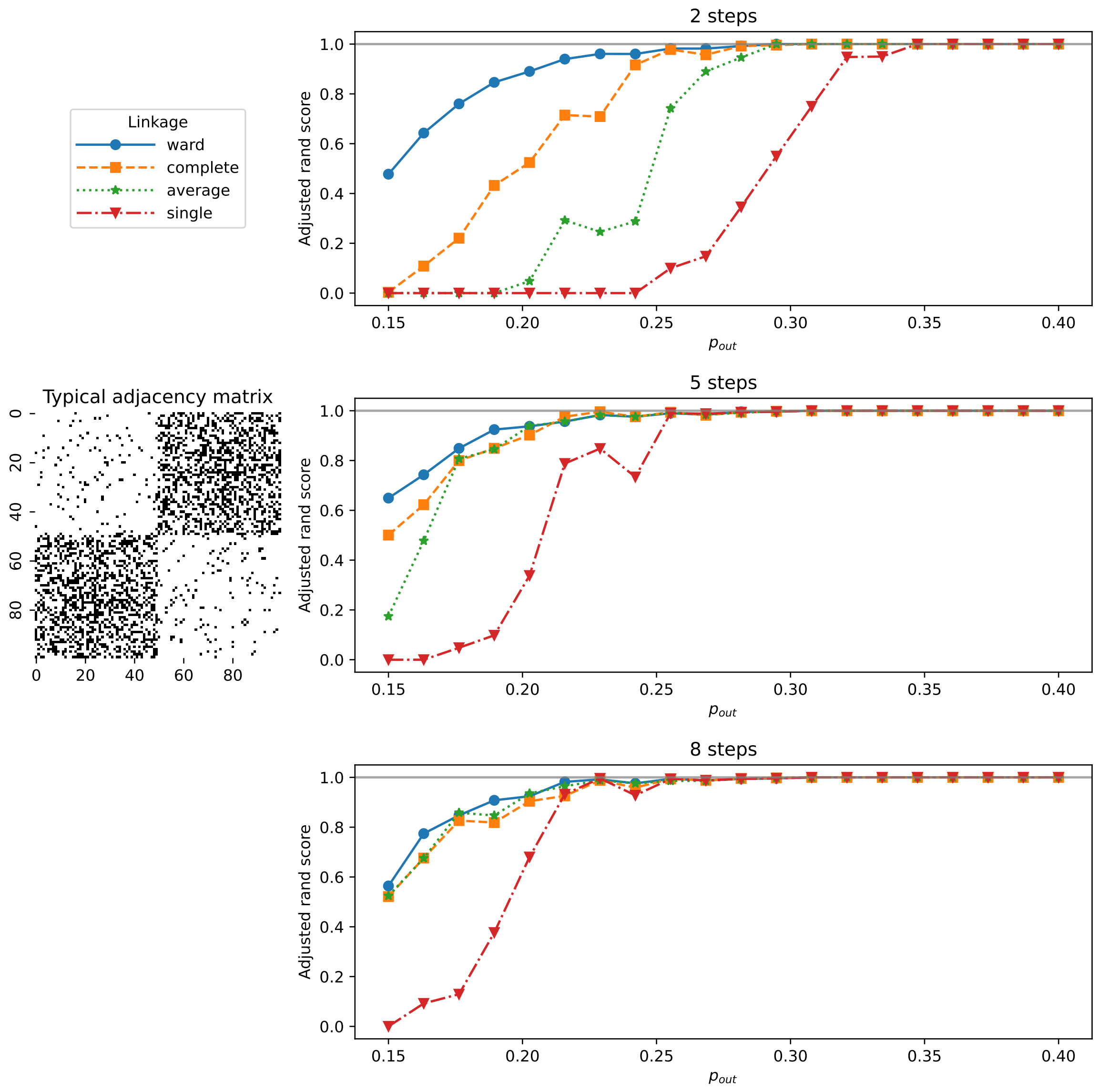


Figure 3.13: Testing Walktrap on near-bipartite graphs with two groups of the same size 50. The internal density of both groups is fixed at 0.05, while the external density varies from 0.15 to 0.40.

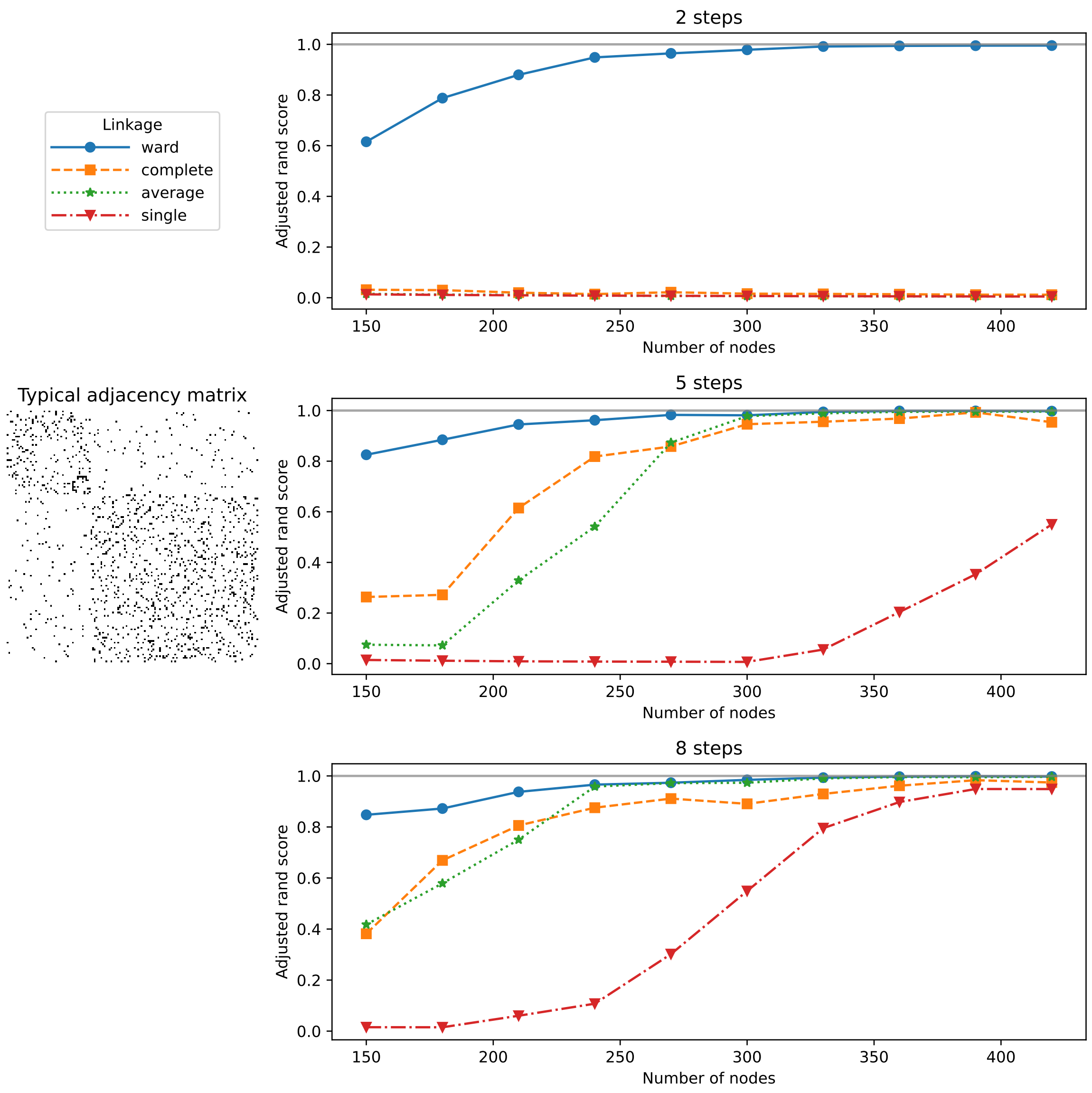


Figure 3.14: Testing the consistency of Walktrap on graphs with two unbalanced groups, as the number of nodes increases. The two groups have size $(k, 2k)$, as k varies from 50 to 140. The external density is fixed at 0.02, and the internal density of both groups is fixed at 0.10.

Chapter 4

CONCLUSION

This chapter summarizes the main content of the thesis and introduces some further directions.

4.1 Summary of the thesis

This thesis is an exposition of two topics in network community detection: a popular quality function called modularity, and the clustering properties of the random walk matrix of a graph. Here we summarize the main content and contribution of this thesis.

- Chapter 2 is a detailed exposition of modularity, its basic properties, and its use in practical tasks.

  Section 2.1 presents the definition of modularity, derives some simple bounds, and carefully explains the connection with the configuration random graph model.

  Section 2.2 collects many theoretical properties of modularity. The results come from many sources; we organize them and give detailed proofs with some additional detail and possible simplification. In particular, the proof of Theorem 2.13, which is based on the proof of Theorem 2.12, is more elementary than current available proofs (which use linear algebra).

  Section 2.3 explains several shortcomings of modularity when used in community detection. The resolution limit and difficulty in interpreting modularity are illustrated by practical examples.

- In Chapter 3, we explore spectral clustering and the Walktrap algorithm for graphs, mostly from the practical perspective.

  Section 3.1 briefly recalls the basic properties of random walks on graphs and the associated random walk matrices.

  Section 3.2 introduces the clustering properties of the spectrum and eigenvectors of the random walk matrices. Those properties are then illustrated on various graphs generated from the stochastic block models.

Section 3.3 introduces the Walktrap algorithm, with particular focus on the clustering property of the Walktrap distance and its connection to spectral clustering. We perform extensive experiments on small random graphs to the check the effects of two parameters: the random walk step size, and the linkage method in the agglomerative clustering step.

The main theoretical content of the thesis lies in Section 2.2. In Sections 2.3, 3.2 and 3.3, we produce copious illustrations and experiments using the Python programming language and several libraries (information can be found on page 50); moreover, a complete implementation of the Walktrap algorithm is provided in Appendix A.

4.2 Some further directions

The linear algebraic perspective on modularity

There are connections between modularity and spectra of graphs. Let d be the $n \times 1$ degree vector of the graph G, i.e. $d(u) = \deg(u)$, and A the adjacency matrix of G. The *modularity matrix* of G is the $n \times n$ matrix

$$M := A - \frac{1}{2m} dd^\top.$$

Equation (2.3) then becomes

$$q_{\mathcal{P}}(G) = \frac{1}{2m} \sum_{u,v \in V(G)} M_{uv} \cdot 1_{\sigma_{\mathcal{P}}(u) = \sigma_{\mathcal{P}}(v)}.$$

The *algebraic modularity* of G is

$$\mu(G) := \max_{x^\top \mathbf{1} = 0} \frac{x^\top M x}{x^\top x}.$$

The following is one of the key results for algebraic modularity.

Theorem 4.1 ([61, Theorem 5.4]). *For any graph G,*

$$q^*(G) \leq \frac{n-1}{2m} \cdot \mu(G).$$

This is how the results of [38, 39] imply Theorem 2.13.

There are also connections between the number of positive eigenvalues of M and the number of communities in G. For more details, see the survey [61].

The axiomatic approach to quality functions

There are many variants of modularity, some specifically designed to avoid resolution limit, but they all have some shortcomings. A principled approach to evaluating and designing quality functions is the axiomatic approach: we formalize the desired properties of such functions, then systematically check them. A set of such axioms is proposed in [62]. There are two key properties that modularity does not satisfies.

- Modularity is not *local*. We do not present the definition here, but intuitively it means that changes in a corner of a graph should not affect the clustering in another corner. The resolution limit violates this property.
- Modularity is not *monotonic*. For a partition $\mathcal{P}$, the modularity $q(\mathcal{P})$ should increase if we improve the community structure of $\mathcal{P}$ itself by deleting edges between members and/or adding edges inside members, but that is not the case.

  For example, let G be a graph with vertex set $V = \{1, 2, 3, 4\}$ and edge set $E = \{(1, 2), (3, 4)\}$. Consider the partition $\mathcal{P} = \{\{1\}, \{2\}, \{3, 4\}\}$, with modularity $q(\mathcal{P}) = 1/8$. If we delete the edge $(1, 2)$, modularity decreases to 0.

A different approach, specifically focusing on resolution limit, is presented in [63].

Bibliography

[1] Michele Coscia. The Atlas for the Aspiring Network Scientist, 2021. arXiv:2101.00863v2.

[2] Katharina A. Zweig. *Network Analysis Literacy.* Springer Vienna, 2016.

[3] Ulrik Brandes, Garry Robins, Ann McCranie, and Stanley Wasserman. What is network science? *Network Science*, 1(1):1–15, 2013.

[4] Daniel Kostić. Mechanistic and topological explanations: an introduction. *Synthese*, 195(1):1–10, 2018.

[5] Laura Turnbull, Marc-Thorsten Hütt, Andreas A. Ioannides, Stuart Kininmonth, Ronald Poeppl, Klement Tockner, Louise J. Bracken, Saskia Keesstra, Lichan Liu, Rens Masselink, and Anthony J. Parsons. Connectivity and complex systems: learning from a multi-disciplinary perspective. *Applied Network Science*, 3(1):1–49, 2018.

[6] Leo Torres, Ann S. Blevins, Danielle Bassett, and Tina Eliassi-Rad. The why, how, and when of representations for complex systems. *SIAM Review*, 63(3):435–485, 2021.

[7] Stephen P. Borgatti, Ajay Mehra, Daniel J. Brass, and Giuseppe Labianca. Network analysis in the social sciences. *Science*, 323(5916):892–895, 2009.

[8] César A. Hidalgo. Disconnected, fragmented, or united? A trans-disciplinary review of network science. *Applied Network Science*, 1(1):1–19, 2016.

[9] Mathieu Jacomy. Epistemic clashes in network science: Mapping the tensions between idiographic and nomothetic subcultures. *Big Data and Society*, 7(2), 2020.

[10] Cristopher Moore. The computer science and physics of community detection: Landscapes, phase transitions, and hardness. *Bulletin of EATCS*, 1(121), 2017.

[11] Mengjia Xu. Understanding graph embedding methods and their applications. *SIAM Review*, 63(4):825–853, 2021.

[12] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5):75–174, 2010.

[13] Santo Fortunato and Darko Hric. Community detection in networks: A user guide. *Physics Reports*, 659:1–44, 2016.

[14] Conrad Lee and Pádraig Cunningham. Community detection: effective evaluation on large social networks. *Journal of Complex Networks*, 2(1):19–37, 2014.

[15] Darko Hric, Tiago P. Peixoto, and Santo Fortunato. Network structure, metadata, and the prediction of missing nodes and annotations. *Physical Review X*, 6(3):031038, 2016.

[16] Leto Peel, Daniel B. Larremore, and Aaron Clauset. The ground truth about metadata and community detection in networks. *Science Advances*, 3(5):e1602548, 2017.

[17] Salvatore Citraro and Giulio Rossetti. Identifying and exploiting homogeneous communities in labeled networks. *Applied Network Science*, 5(1):1–20, 2020.

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

[19] Vinh-Loc Dao, Cécile Bothorel, and Philippe Lenca. Community detection methods can discover better structural clusters than ground-truth communities. In *2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 395–400. IEEE, 2017.

[20] Liudmila Prokhorenkova and Alexey Tikhonov. Community detection through likelihood optimization: in search of a sound model. In *The World Wide Web Conference*, pages 1498–1508, 2019.

[21] Andrea Lancichinetti, Mikko Kivelä, Jari Saramäki, and Santo Fortunato. Characterizing the community structure of complex networks. *PloS One*, 5(8):e11976, 2010.

[22] Tiago P. Peixoto. Descriptive vs. inferential community detection: pitfalls, myths and half-truths, 2022. arXiv:2112.00183v4.

[23] Martin Rosvall, Jean-Charles Delvenne, Michael T. Schaub, and Renaud Lambiotte. Different approaches to community detection. In Patrick Doreian, Vladimir Batagelj, and Anuška Ferligoj, editors, *Advances in Network Clustering and Blockmodeling*, pages 105–119. John Wiley & Sons, 2020.

[24] Satu Elisa Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27–64, 2007.

[25] Michele Coscia, Fosca Giannotti, and Dino Pedreschi. A classification for community discovery methods in complex networks. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 4(5):512–546, 2011.

[26] Vinh Loc Dao, Cécile Bothorel, and Philippe Lenca. Community structure: A comparative evaluation of community detection methods. *Network Science*, 8(1):1–41, 2020.

[27] Zhao Yang, René Algesheimer, and Claudio J. Tessone. A comparative analysis of community detection algorithms on artificial networks. *Scientific Reports*, 6(1):1–18, 2016.

[28] Amir Ghasemian, Homa Hosseinmardi, and Aaron Clauset. Evaluating overfit and underfit in models of network community structure. *IEEE Transactions on Knowledge and Data Engineering*, 32(9):1722–1735, 2019.

[29] Jure Leskovec, Kevin J. Lang, Anirban Dasgupta, and Michael W. Mahoney. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, 6(1):29–123, 2009.

[30] Puck Rombach, Mason A. Porter, James H. Fowler, and Peter J. Mucha. Core-periphery structure in networks (revisited). *SIAM Review*, 59(3):619–646, 2017.

[31] Lucas G.S. Jeub, Prakash Balachandran, Mason A. Porter, Peter J. Mucha, and Michael W. Mahoney. Think locally, act locally: Detection of small, medium-sized, and large communities in large networks. *Physical Review E*, 91(1):012821, 2015.

[32] Vinh-Loc Dao, Cécile Bothorel, and Philippe Lenca. Community structures evaluation in complex networks: A descriptive approach. In *International Conference and School on Network Science*, pages 11–19. Springer, 2017.

[33] Vinh-Loc Dao, Cécile Bothorel, and Philippe Lenca. An empirical characterization of community structures in complex networks using a bivariate map of quality metrics. *Social Network Analysis and Mining*, 11(1):1–20, 2021.

[34] Mark Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69(2):026113, 2004.

[35] Ulrik Brandes, Daniel Delling, Marco Gaertler, Robert Gorke, Martin Hoefer, Zoran Nikoloski, and Dorothea Wagner. On modularity clustering. *IEEE Transactions on Knowledge and Data Engineering*, 20(2):172–188, 2007.

[36] Fiona Skerman. *Modularity of networks.* PhD thesis, University of Oxford, 2015.

[37] Colin McDiarmid and Fiona Skerman. Modularity of regular and treelike graphs. *Journal of Complex Networks*, 6(4):596–619, 2018.

[38] Snježana Majstorović and Dragan Stevanović. A note on graphs whose largest eigenvalue of the modularity matrix equals zero. *The Electronic Journal of Linear Algebra*, 27:611–618, 2014.

[39] Marianna Bolla, Brian Bullins, Sorathan Chaturapruek, Shiwen Chen, and Katalin Friedl. Spectral properties of modularity matrices. *Linear Algebra and Its Applications*, 473:359–376, 2015.

[40] Colin McDiarmid and Fiona Skerman. Modularity of Erdős-Rényi random graphs. *Random Structures & Algorithms*, 57(1):211–243, 2020.

[41] Thang N. Dinh and My T. Thai. Finding community structure with performance guarantees in scale-free networks. In *2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing*, pages 888–891. IEEE, 2011.

[42] Thang N. Dinh, Xiang Li, and My T. Thai. Network clustering via maximizing modularity: Approximation algorithms and theoretical limits. In *2015 IEEE International Conference on Data Mining*, pages 101–110. IEEE, 2015.

[43] Benjamin H. Good, Yves-Alexandre De Montjoye, and Aaron Clauset. Performance of modularity maximization in practical contexts. *Physical Review E*, 81(4):046106, 2010.

[44] Santo Fortunato and Marc Barthelemy. Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, 104(1):36–41, 2007.

[45] Andrea Lancichinetti and Santo Fortunato. Limits of modularity maximization in community detection. *Physical Review E*, 84(6):066122, 2011.

[46] Vincent A. Traag, Gautier Krings, and Paul Van Dooren. Significant scales in community structure. *Scientific Reports*, 3(1):1–10, 2013.

[47] Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.

[48] Bailey K. Fosdick, Daniel B. Larremore, Joel Nishimura, and Johan Ugander. Configuring random graph models with fixed degree sequences. *Siam Review*, 60(2):315–355, 2018.

[49] Roger A. Horn and Charles R. Johnson. *Matrix Analysis.* Cambridge University Press, 2nd edition, 2012.

[50] Martin G. Everett and Stephen P. Borgatti. Partitioning multimode networks. In Patrick Doreian, Vladimir Batagelj, and Anuška Ferligoj, editors, *Advances in Network Clustering and Blockmodeling*, pages 251–265. John Wiley & Sons, 2020.

[51] Pascal Pons and Matthieu Latapy. Computing communities in large networks using random walks. *Journal of Graph Algorithms and Applications*, 10(2):191–218, 2006.

[52] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[53] Maria C.V. Nascimento and Andre C.P.L.F. De Carvalho. Spectral methods for graph clustering–a survey. *European Journal of Operational Research*, 211(2):221–231, 2011.

[54] Marina Meila. Spectral clustering. In Christian Hennig, Marina Meila, Fionn Murtagh, and Roberto Rocci, editors, *Handbook of Cluster Analysis*, pages 125–141. CRC Press, 2015.

[55] Frank Bauer and Jürgen Jost. Bipartite and neighborhood graphs and the spectrum of the normalized graph Laplace operator. *Communications in Analysis and Geometry*, 21(4):787–845, 2013.

[56] Ronald R. Coifman and Stéphane Lafon. Diffusion maps. *Applied and Computational Harmonic Analysis*, 21(1):5–30, 2006.

[57] Stéphane Lafon and Ann B. Lee. Diffusion maps and coarse-graining: A unified framework for dimensionality reduction, graph partitioning, and data set parameterization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9):1393–1403, 2006.

[58] Brian S. Everitt, Sabine Landau, Morven Leese, and Daniel Stahl. *Cluster Analysis*. John Wiley & Sons, 5th edition, 2011.

[59] Martijn M. Gösgens, Alexey Tikhonov, and Liudmila Prokhorenkova. Systematic analysis of cluster similarity indices: How to validate validation measures. In *International Conference on Machine Learning*, pages 3799–3808. PMLR, 2021.

[60] Michèle Thieullen and Alexis Vigot. Length of clustering algorithms based on random walks with an application to neuroscience. *Chaos, Solitons & Fractals*, 45(5):629–639, 2012.

[61] Dario Fasino and Francesco Tudisco. An algebraic analysis of the graph modularity. *SIAM Journal on Matrix Analysis and Applications*, 35(3):997–1018, 2014.

[62] Twan Van Laarhoven and Elena Marchiori. Axioms for graph clustering quality functions. *Journal of Machine Learning Research*, 15(1):193–215, 2014.

[63] Vincent A. Traag, Paul Van Dooren, and Yurii Nesterov. Narrow scope for resolution-limit-free community detection. *Physical Review E*, 84(1):016114, 2011.

Appendix A

A Python implementation of Walktrap

This is a simple Python implementation of Walktrap, designed to work on NetworkX graphs. We use the concepts introduced in Section 2.1: edge contribution, degree tax, and modularity of weighted graphs.

As mentioned in Section 3.3, we do not add self-loops to the graphs. The code below can easily be modified to accommodate self-loops.

```python
# -*- coding: utf-8 -*-
# file: walktrap.py

import copy
import heapq
import numpy as np
import networkx as nx

class Walktrap:
    '''
    A class implementing the Walktrap clustering method for networkx Graphs.

    This implementation only works for undirected and unweighted graphs.
    Loops and edge weights should have been discarded before.

    It is assumed that the graph is connected and the nodes are labeled
    using consecutive integers 0..(n-1).

    The best partition is determined using modularity. You can vary the
    resolution parameter for modularity calculation; generally higher
    resolutions give more communities.

    Basic usage:
        clustering = Walktrap(G)
        clustering.fit(n_steps=3)
        print(clustering.best_partition())
    '''
    def __init__(self, G):
        '''
        Initialize the Walktrap object.

        Parameters
        ----------
        G : networkx Graph
            It is assumed that the nodes of G are labeled using
            consecutive integers 0..(n-1).
        '''
        self.G = G

    def fit(self, n_steps=3):
        '''
        Perform the Walktrap clustering.

        Parameters
        ----------
        n_steps : int, default=3
            The number of random walk steps; operationally this is the power
            to which we raise the transition matrix. Recommended choices are
            between 3 to 8, inclusive. Generally, sparser graphs require
            longer step sizes.

        Returns
        ----------
        self : Walktrap object
        '''
        # A 'symbolic' graph, where vertices represent clusters, and edge
        # weights represent the number of edges between clusters.
        # This graph will gradually be collapsed.
        H = copy.deepcopy(self.G)
        for e in H.edges:
            H.edges[e]['weight'] = 1

        n = H.number_of_nodes()
        m = H.number_of_edges()

        # CREATE THE WALKTRAP MATRIX

        A = nx.to_numpy_array(
            H,
            nodelist=list(range(n)),
            weight=None,
            dtype=np.float64
        )
        degrees = np.sum(A, axis=1)
        P = A / degrees[:, np.newaxis] # the transition matrix
        P = np.linalg.matrix_power(P, n_steps) # walking
        W = P / np.sqrt(degrees) # final walktrap matrix

        # AUXILIARY OBJECTS

        # Storing the merging steps.
        # At step i, clusters children[i,0] and children[i,1] are merged
        # to form cluster n+i.
        children = np.zeros((n-1, 2), dtype=np.int64)

        # Quickly check when we encountered deleted vertices
        deleted = np.zeros(2*n - 1, dtype=np.bool8)

        # Reuse the walktrap matrix to store new rows
        # comm_to_row[u] is the row in W corresponding to cluster u
        comm_to_row = np.zeros(2*n - 1, dtype=np.int64)
        comm_to_row[:n] = range(n)

        # sizes[u] is the number of vertices in cluster u
        sizes = np.zeros(2*n - 1, dtype=np.int64)
        sizes[:n] = 1

        # vols[u] is the sum of degrees of vertices in cluster [u]
        vols = np.zeros(2*n - 1, dtype=np.int64)
        vols[:n] = [H.degree(v) for v in range(n)]

        # internal_ecount[u] is the number of edges inside cluster u
        internal_ecount = np.zeros(2*n - 1, dtype=np.int64)

        # delta_econ[i] is the change in edge contribution at step i
        delta_econ = np.zeros(n-1, dtype=np.float64)
        # delta_dtax[i] is the change in degree tax at step i
        delta_dtax = np.zeros(n-1, dtype=np.float64)

        # A min heap for efficient extraction of the minimum change in sse.
        delta_sse_heap = [
            (
                np.linalg.norm(
                    W[v1, :] - W[v2, :], ord=2
                )**2 / 2,
                v1,
                v2
            ) for v1, v2 in self.G.edges()
        ]
        heapq.heapify(delta_sse_heap)

        # MERGING
```

```

    for i in range(n - 1):
        u = n + i # the new node

        # Get the two communities to merge
        while True:
            delta_sse, v1, v2 = heapq.heappop(delta_sse_heap)
            if (not deleted[v1]) and (not deleted[v2]):
                break # found!

        # Update the auxiliaries
        children[i, :] = [v1, v2]
        deleted[v1] = deleted[v2] = True
        sizes[u] = sizes[v1] + sizes[v2]
        vols[u] = vols[v1] + vols[v2]
        internal_ecount[u] = (
            internal_ecount[v1] + internal_ecount[v2]
            + H.edges[v1, v2]['weight']
        )

        comm_to_row[u] = comm_to_row[v1]
        W[comm_to_row[u], :] = (
            sizes[v1] * W[comm_to_row[v1], :]
            + sizes[v2] * W[comm_to_row[v2], :]
        ) / sizes[u]

        delta_econ[i] = H.edges[v1, v2]['weight'] / m
        delta_dtax[i] = (vols[v1] * vols[v2]) / (2 * m**2)

        # Adding to the cluster graph and the heap
        v1_neighbors = set(H.neighbors(v1))
        v2_neighbors = set(H.neighbors(v2))
        u_neighbors = (v1_neighbors | v2_neighbors) - set([v1, v2])
        H.add_node(u)
        for v in u_neighbors:
```

```python
                H.add_edge(u, v)
                weight = 0
                if v in v1_neighbors:
                    weight += H.edges[v, v1]['weight']
                if v in v2_neighbors:
                    weight += H.edges[v, v2]['weight']
                H.edges[u,v]['weight'] = weight
                heapq.heappush(
                    delta_sse_heap,
                    (
                        np.linalg.norm(
                            W[comm_to_row[u], :] - W[comm_to_row[v], :],
                            ord=2
                        )**2 * sizes[u] * sizes[v] / (sizes[u] + sizes[v]),
                        u,
                        v
                    )
                )
            # Clean up the cluster graph
            H.remove_nodes_from([v1, v2])

        # STORING USEFUL ATTRIBUTES

        # Compute edge contribution and degree tax at each step
        mod_econ = np.zeros(n, dtype=np.float64)
        mod_econ[1:] = np.cumsum(delta_econ)
        mod_dtax = np.zeros(n, dtype=np.float64)
        mod_dtax[1:] = np.cumsum(delta_dtax)
        mod_dtax += np.sum([vols[v]**2 for v in range(n)]) / (4 * m**2)

        # Storing
        self.children = children
        self.mod_econ = mod_econ
        self.mod_dtax = mod_dtax

        return self

    def modularities(self, resolution=1.0):
        '''
        Return all the modularity at all merging steps

        Parameters
        ----------
        resolution : float, default=1.0
            The resolution (gamma) in the modularity formula

        Returns
        ----------
        A numpy array of floats (n,), where index i store the modularity
        at level i
        '''
        return self.mod_econ - resolution*self.mod_dtax

    def partition(self, n_groups):
        '''
        Compute the partition with the specified number of communities.

        Parameters
        ----------
        n_groups : integer, from 1, 2, ... , n
            The number of communities to return.

        Returns
        ----------
        A partition as a list of sets.
        '''
        n = self.G.number_of_nodes()
        clusters = {v:{v} for v in range(n)}
        for i in range(n - n_groups):
            v1, v2 = self.children[i]
            v1_cluster = clusters.pop(v1)
            v2_cluster = clusters.pop(v2)
            clusters[n + i] = v1_cluster | v2_cluster

        return list(clusters.values())

    def best_partition(self, resolution=1.0):
        '''
        Return the partition with the best modularity.

        Parameters
        ----------
        resolution : float, default=1.0
            The resolution parameter in the modularity formula.

        Returns
        ----------
        The partition with the best modularity, as a list of sets.
        '''
        return self.partition(
            self.G.number_of_nodes() -
            np.argmax(
                self.mod_econ - resolution*self.mod_dtax
            )
        )
```