Community Identification in Networks

Gianluca Campanella

Supervised by Prof. Ricardo Alberto Marques Pereira

Academic Year 2008/2009
Contents

Acknowledgments 1

Chapter 1. Introduction 2
  1. Notation and terminology 2

Chapter 2. Edge removal methods 5
  1. Edge betweenness 5
  2. Edge clustering coefficient 8

Chapter 3. Spectral methods 9
  1. Spectral partitioning and cut size minimization 9
  2. Spectral optimization of modularity 11

Chapter 4. Other methods of modularity optimization 17
  1. Integer Linear Programming 17
  2. Simulated annealing 18
  3. Agglomerative hierarchical clustering 18

Chapter 5. Redefining modularity 23
  1. Modularity matrix 23
  2. Community decomposition of the modularity 28
  3. Application 30

Chapter 6. Case study: the scientific collaboration network of Italian computer scientists 32
  1. Data collection 32
  2. Degree distribution 32
  3. Community identification 33

Appendix A. Diagonal diffusion operator 36

Bibliography 38
Acknowledgments

Many people have contributed to this thesis, to my education, and to my life. It is an honor to have come across so many wonderful people, and it is now my great pleasure to take this opportunity to thank them.

First and foremost, I wish to express my gratitude to my supervisor, Prof. Ricardo Alberto Marques Pereira, who offered invaluable assistance, support and guidance. This thesis would not have been possible without his encouragement, wisdom and feedback.

I also wish to express my love and gratitude to my beloved parents, Anna and Vincenzo, for their endless understanding and unconditional love throughout the duration of my studies, and for their patience during the time we were separated.

I am highly indebted to all the professors I was pleased to meet while in high school, especially Prof. Mariamabile Santi and Prof. Enrico Bizzarri, for the incredible work they have done.

Finally, I must thank my friends and colleagues, in particular Frozan Qarin, Sonia A. Quispe Lisarazo and Willi Cometti, for making my time in Bolzano that much more enjoyable, and for making these years ones I will always look back upon warmly.

Gianluca Campanella
CHAPTER 1

Introduction

The network paradigm is now used in many fields of knowledge as a powerful and concise mathematical tool for representing the structure and topology of complex systems of interacting components.

Metabolic, protein, and genetic regulatory networks are now heavily studied in biology and medicine, as much as the Internet and the World Wide Web are studied in computer and information sciences, food webs and other networks of interaction among species in ecology, personal or social networks in epidemiology, sociology, and the management sciences [NL07].

Identifying community structure in networks can provide insight into how network function and network topology affect each other.

The study of networks, in fact, dates back much further than the current surge of interest in it. However, earlier studies focused on small networks, whereas nowadays networks of thousands or millions of vertices are not unusual, making quantitative analysis conducted by inspection impossible.

For this reason, we have been forced to turn to topological measures, computer algorithms, and statistics to understand the structure of today's networks.

In this thesis, after introducing some useful shorthand notation that will then be used throughout the text, we analyze a number of methods for community identification that have been proposed in the recent literature.

The thesis is organized as follows: starting from edge removal methods, we move to classical graph theory and spectral partitioning, then apply similar methods to optimize a newly defined objective function known as modularity.

We then turn to other methods for optimizing this objective function, namely Integer Linear Programming (ILP) and greedy approaches such as Simulated Annealing and Agglomerative Hierarchical Clustering (AHC).

Furthermore, we extend the classical definition of modularity, and verify its effectiveness by means of some preliminary studies.

We conclude this work by applying the current state-of-the-art algorithm to the real-world network of scientific collaborations among Italian computer scientists.

1. Notation and terminology

The framework for a rigorous mathematical description of networks is found in graph theory. For this reason, we will begin our review with an introduction to the basic elements of this discipline.

Since a substantial body of knowledge in the field has been built up, we will only provide some standard terminology and notation commonly used throughout the text. In particular, we will mainly focus on the matrix representation of graphs.
The interested reader is referred to introductory books on graph theory, such as [Die05].

1.1. Matrices. Since we will be frequently dealing with symmetric matrices, we will begin by introducing a shorthand notation to allow for a more compact representation.

Let \( A = [a_{ij}] \) be a symmetric matrix of order \( n \), with \( i, j = 1, \ldots, n \) and \( A^T = A \).

**Definition 1.1 (Sum of a row or column)**. The sum of row (or column) \( i \) is denoted by
\[
a_i = \sum_j a_{ij}
\]

**Definition 1.2 (Sum of all elements)**. The sum of all elements of the matrix is denoted by
\[
a = \sum_i a_i = \sum_{i,j} a_{ij}
\]

1.2. Undirected graphs. In general terms, networks can be described as undirected graphs whose vertices identify the elements of some system and whose edges represent interactions among them.

**Definition 1.3 (Undirected graph)**. An undirected graph \( G \) is a tuple \( \langle V, A \rangle \), where \( V = \{v_1, v_2, \ldots, v_n\} \) is the set of vertices and the symmetric matrix \( A = A^T \) of order \( n \) with elements in \( \{0, 1\} \) is the adjacency matrix describing the edges.

For all \( v_i, v_j \in V \) it holds that
\[
\begin{align*}
a_{ii} &= 0 \quad \text{(irreflexivity)} \\
a_{ij} &= a_{ji} \in \{0, 1\} \quad \text{(symmetry)}
\end{align*}
\]

**Definition 1.4 (Order of a graph)**. A graph with \( n = |V| \) vertices is said to have order \( n \).

**Definition 1.5 (Size of a graph)**. A graph with \( m \) edges is said to have size \( m = a/2 \), where the factor \( \frac{1}{2} \) accounts for the symmetry of \( A \).

**Remark**. A graph of order \( n \) can have at most \( m_{\max} = n(n-1)/2 \) edges, where the factor \( 1/2 \) accounts for the symmetry of \( A \).

**Definition 1.6 (Complete, dense and sparse graphs)**. If \( m = m_{\max} \) the graph is said to be complete, whereas if \( m \approx m_{\max} \) the graph is said to be dense. Conversely, if \( m \propto n \) the graph is said to be sparse.

**Definition 1.7 (Adjacent vertices or neighbors)**. Two vertices \( v_i \) and \( v_j \) such that \( a_{ij} = 1 \) are said to be adjacent or neighboring vertices.

**Definition 1.8 (Degree of a vertex)**. The degree of a vertex \( v_i \in V \) is simply the number of incident edges, which corresponds to the sum of all elements in row (or column) \( i \), that is \( a_i \in \mathbb{N} \).

1.3. Weighted undirected graphs. Undirected graphs can be naturally extended to weighted undirected graphs, in which edges are assigned a weight representing the strength of the interaction among the vertices they connect.

**Definition 1.9 (Weighted undirected graph)**. A weighted undirected graph \( G \) is a tuple \( \langle V, A \rangle \), where \( V \) is the set of vertices and the symmetric matrix \( A = A^T \) of order \( n \) with elements in \( [0, 1] \) is the adjacency matrix describing the edges and satisfying the properties of Definition 1.3.
DEFINITION 1.10 (Adjacent vertices or neighbors). Two vertices \( v_i \) and \( v_j \) such that \( a_{ij} \neq 0 \) are said to be adjacent or neighboring vertices.

DEFINITION 1.11 (Degree of a vertex). The degree of a vertex \( v_i \in V \) is simply the weight sum of the incident edges, which corresponds to the sum of all elements in row (or column) \( i \), that is \( a_i \in \mathbb{R} \).

1.4. Laplacian matrix. The Laplacian matrix, which is also called admittance matrix or Kirchhoff matrix, is an alternative matrix representation of a graph. As we will see in Section 1, it plays an important role in spectral graph theory.

DEFINITION 1.12 (Laplacian matrix). Given a graph \( G \) represented by the adjacency matrix \( A \), we define the associated symmetric Laplacian matrix \( L = L^T \) of order \( n \) as follows,

\[
l_{ij} = \begin{cases} 
-a_{ij} & i \neq j \\
 0 & i = j 
\end{cases}
\]

PROPOSITION 1.1. The Laplacian matrix \( L \) is positive-semidefinite or, equivalently, all the eigenvalues of \( L \) are non-negative [GR01].

REMARK. Since all row (or column) sums of \( L \) are equal to zero, \( L \) has an eigenvector with associated zero eigenvalue.

In particular, for disconnected graphs with \( k \) separate components, \( L \) is block diagonal and has \( k \) degenerate eigenvectors all corresponding to the zero eigenvalue.

On the other hand, if \( k \) denser components can be identified, but there exist some edges between them, \( L \) has a single eigenvector with associated zero eigenvalue, and \( k - 1 \) eigenvectors with eigenvalues slightly greater than zero [Bo102].
CHAPTER 2

Edge removal methods

The first approaches to community identification presented in the literature are based on the idea that any given network can be partitioned into its constituent communities by iteratively identifying and removing edges that connect them [New04a].

1. Edge betweenness

One of the first iterative removal methods, presented by Girvan and Newman, is based on edge betweenness [GN02], which is defined as follows.

**Definition 2.1 (Shortest path).** The shortest path between two vertices \( v_i \) and \( v_j \) is a path (i.e. a sequence of vertices connected by edges) that minimizes the number of its constituent edges.

**Definition 2.2 (Edge betweenness).** For the edge connecting vertices \( v_i \) and \( v_j \), the edge betweenness is defined as the number of shortest paths passing through it.

**Remark.** The two previous definitions can be extended to the context of weighted graphs in the obvious way: for instance, the shortest path minimizes the sum of the weights of its constituent edges.

**Remark.** It has been observed that the conventional definition of betweenness, seen as a measure of the influence an edge has over the spread of information through the network, implicitly assumes that information spreads only along shortest paths. Different betweenness measures have thus been proposed by relaxing this assumption and including contributions from all paths between vertices, not just the shortest. A measure based on random walks, counting how often a vertex is traversed by a random walk between two other vertices, has been presented in [New05].

Intuitively, edges with high betweenness are those bridging communities together, and thus are good candidates for removal (Figure 2.1).

The algorithm proceeds as follows:

1. Calculate the edge betweenness for all edges by computing the shortest paths between all pairs of vertices
2. Remove the edge with the highest edge betweenness
3. Repeat until the network is split into two parts, then apply the algorithm iteratively within each partition until the network is reduced to individual vertices

As the algorithm proceeds, it is possible to construct a dendrogram (top down) recording where each division took place (Figure 2.2). Note that, in general, the dendrogram will not branch at every edge removal, since it may be necessary to remove multiple edges before the network is split into two disconnected components.
As a real-world example, the authors consider the well-known "karate club" study of Zachary [Zac77], in which 34 members of a karate club were observed over a period of two years. During the course of the study, a disagreement developed between the administrator of the club and its instructor, which ultimately resulted in the instructor leaving and starting a new club, taking about half of the original club members with him. Figure 2.3 shows the network, with the instructor and the administrator represented by vertices 0 and 33, respectively. The most fundamental split in the network is the first one to occur (and, thus, to be recorded in the dendrogram), which divides the network into two groups of roughly equal size. This split corresponds almost perfectly to the actual division of the club members following the break-up, as revealed by which club they attended afterwards, with the only exception of vertex 2, which is classified incorrectly. Thus, the authors conclude that the application of the algorithm to the empirically observed network of friendships is a good predictor of the subsequent social evolution of the group.

The most computer-intensive part of the algorithm is the computation of the edge betweenness, which in turn requires the computation of shortest paths between
all pairs of vertices. Using the fast methods developed independently by Newman [New01b] and Brandes [Bra01], this can be done in $O(mn)$ time for unweighted graphs, resulting in a total complexity of $O(m^2n)$ (since all $m$ edges are eventually removed), which limits the order of graphs that can be treated with this method and current computer technology to a few thousand vertices.

One might be tempted to avoid recalculating the edge betweenness at each iteration of the algorithm. However, as Girvan and Newman point out [GN02], this recomputation is crucial to the accuracy of the algorithm. This is because, if there are more than two edges connecting two different communities, there is no guarantee that all of them will have high betweenness, but only that at least one of them will.

1.1. Newman’s shortest path algorithm. The algorithm for computing the shortest path between two vertices $v_i$ and $v_j$ described in [New01b] is a modified form of standard breadth-first search, and proceeds as follows:

1. At the beginning, assign vertex $v_j$ distance zero (to indicate that it is zero steps away from itself) and set $d = 0$
2. For each vertex $v_k$ whose assigned distance is $d$, follow each attached edge to the vertex $v_l$ at its other end and, if $v_l$ has not already been assigned a distance, assign it distance $d + 1$ and declare $v_k$ to be a predecessor of $v_l$
3. Increase $d$ to $d + 1$ and repeat from the previous step until $v_i$ is reached
4. At the end, the shortest path from $v_i$ to $v_j$ (if there exist one) is obtained by stepping back from $v_i$ to its predecessor, and then to the predecessor of each successive vertex until $v_j$ is reached

1If $v_l$ has already been assigned distance $d + 1$, there is no need to do this again, but $v_k$ is still declared a predecessor of $v_l$.
2If a vertex has two or more predecessors, then there are two or more shortest paths, each of which must be followed separately if we wish to know all shortest paths between $v_i$ and $v_j$. 
To alleviate the high computational complexity of the algorithm presented in Section 1, Radicchi et al. proposed to substitute the edge betweenness with the edge clustering coefficient \([\text{RCC} + 04]\), which is defined as follows.

**Definition 2.3 (Edge clustering coefficient).** For the edge connecting vertices \(v_i\) and \(v_j\), the edge clustering coefficient \(c_{ij}^{(3)}\) is defined as

\[
c_{ij}^{(3)} = \frac{z_{ij}^{(3)} + 1}{\min(a_i - 1, a_j - 1)}
\]

where \(z_{ij}^{(3)}\) is the number of triangles the edge belongs to, and the denominator is the maximum number of triangles that might possibly include the edge between the two vertices \(v_i\) and \(v_j\) (Figure 2.4).

The idea behind the use of this quantity is that vertices belonging to the same community should have a larger number of “common friends” than vertices belonging to different communities. This, in turn, means that edges connecting vertices in the same community will be part of many triangles, while those connecting vertices in different communities will belong to, at most, a few triangles.

The algorithm proceeds as the one described in Section 1, removing at each iteration the edge with the lowest edge clustering coefficient and building a dendrogram (top down). However, since calculations of the edge clustering coefficients can be carried out using only local information, this algorithm scales as \(O(n^2)\) and is thus much more efficient than the one proposed by Girvan and Newman.

The authors of [\text{RCC} + 04] also consider an extension of the edge clustering coefficient from triangles to squares and higher-order loops, which allows for a smooth interpolation between a local and non-local algorithm in networks with few triangles.
CHAPTER 3

Spectral methods

1. Spectral partitioning and cut size minimization

There is a large literature on spectral analysis, in which network properties are linked to the spectrum of the Laplacian matrix defined in Section 1.4.

However, this method is generally not well suited to the problem of community identification. The reasons for this, though, are interesting and instructive, so we will start by briefly reviewing the traditional spectral partitioning method.

1.1. Spectral partitioning. The graph partitioning problem consists in dividing the vertices into $k$ groups such that the number of edges between groups (the so-called cut size) is minimized.

Problems of this type can be solved exactly in $O(nk^2)$ [GH94]; however, this is already prohibitive even when the smallest possible value $k = 2$ is considered.

For this reason, a number of approximate solution methods have been developed. A classical and widely used one is the spectral partitioning method [Fie73, PSL90], of which we consider the simplest instance where $k = 2$, namely where vertices must be divided into two disjoint subsets such that the number of edges between them is minimized. This method is commonly referred to as spectral bisection.

1.1.1. Spectral bisection.

Definition 3.1 (Index vector). The index vector $h \in \{1, -1\}^n$, representing an assignment of vertices to each of two groups, has elements

\[ h_i = \begin{cases} 
1 & \text{if vertex } i \text{ belongs to the first group} \\
-1 & \text{if vertex } i \text{ belongs to the second group}
\end{cases} \]

Remark. The vector $h$ satisfies the normalization condition $h^T h = n$.

Definition 3.2 (Cut size). The cut size $R$, which in this case corresponds to the number of edges between the two groups of vertices, can be written in terms of the index vector $h$ and the Laplacian matrix $L$ as follows,

\[ R = \frac{1}{4} h^T L h \]

We can write $h$ as a linear combination of the orthonormal eigenvectors $u_i$ of $L$,

\[ h = \sum_i \alpha_i u_i \]
Multiplying the left-hand side of the previous equation by \( u_i^T \), with \( i \) fixed, we get
\[
    u_i^T h = \sum_j \alpha_j u_i^T u_j \\
    = \sum_j \alpha_j \delta_{ij} \\
    = \alpha_i
\]

If instead we multiply the left-hand side of the same equation by \( h^T \), we get the normalization constraint
\[
    n = h^T h \\
    = \sum_{i,j} \alpha_i \alpha_j u_i^T u_j \\
    = \sum_{i,j} \alpha_i \alpha_j \delta_{ij} \\
    = \sum_i \alpha_i^2
\]

Finally, we can expand the cut size definition and obtain
\[
    R = \sum_i \alpha_i u_i^T L \sum_j \alpha_j u_j \\
    = \sum_{i,j} \alpha_i \alpha_j \lambda_j \delta_{ij} \\
    = \frac{1}{4} \sum_i \alpha_i^2 \lambda_i
\]

where \( \lambda_i \) is the eigenvalue of \( L \) corresponding to the eigenvector \( u_i \), and we have made use of \( u_i^T u_j = \delta_{ij} \).

Without loss of generality, we assume that the eigenvalues are labeled in increasing order, \( \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n \).

The task of minimizing \( R \) is then equivalent to that of choosing the non-negative quantities \( \alpha_i^2 \) so as to place as much weight as possible in the terms corresponding to the smallest \( \lambda_i \), while respecting the normalization constraint \( \sum_i \alpha_i^2 = n \).

Since the Laplacian \( L \) has always an eigenvector with zero eigenvalue, and it can be shown that the eigenvalues of \( L \) are all non-negative, the zero eigenvalue is always the smallest eigenvalue \( \lambda_1 = 0 \). Choosing the index vector \( h = (1,1,\ldots) \), proportional to \( u_1 \), leads to \( R = 0 \), which is the smallest value \( R \) can take since it is clearly a non-negative quantity. However, this solution is equivalent to placing all vertices in the first group and none in the second, and is thus uninteresting.

It would be useful to forbid this trivial solution, forcing the method to find a sensible one. The most common approach to achieve this goal is to fix the sizes of the two groups as \( n_1, n_2 \) corresponding to the number of \( +1, -1 \) components of the index vector \( h \). It follows that
\[
    \alpha_1^2 = (u_1^T h)^2 = \frac{(n_1 - n_2)^2}{n}
\]

where we recall that \( u_1 = (1/\sqrt{n}, 1/\sqrt{n}, \ldots) \).

The cut size \( R \) would then be minimized by placing all the weight in the term involving \( \lambda_2 \), which would be done by choosing \( h \) proportional to \( u_2 \). If this were
possible, we would have $\alpha_i^2 = (u_2^T h)^2 = (\sqrt{n})^2 = n$, and $\alpha_i^2 = (u_i^T h)^2 = 0$ for $i = 3, \ldots, n$. Moreover, due to the normalization constraint $\sum_i \alpha_i^2 = n$, we would also necessarily have $\alpha_i^2 = 0$, implying $n_1 = n_2$, which in turn means that the cardinality of the two groups would necessarily be the same.

In addition, however, since the elements of $h$ must take the values $\pm 1$, it is often not possible to choose $h$ parallel to $u_2$. Instead, quite good approximate solutions can be obtained by choosing $h$ to be as close to parallel to $u_2$ as possible, leading to the choice

$$
h_i = \begin{cases} 
1 & \text{if } (u_2)_i \geq 0 \\
-1 & \text{if } (u_2)_i < 0
\end{cases}
$$

where $(u_2)_i$ is the $i$th element of $u_2$. In the case of this suboptimal choice, we would have $\alpha_1^2 = (n_1 - n_2)^2 / n$ and $\alpha_2^2 = \sum_i |(u_2)_i|$, with possibly non-null higher-order coefficients.

This concludes our presentation of the spectral partitioning method in its simplest form, the spectral bisection method. While there is no guarantee that it will minimize the cut size $R$, it often does so reasonably well, especially if $\lambda_2 \ll \lambda_3$.

2. Spectral optimization of modularity

Despite its success in the graph partitioning field, spectral partitioning is not appropriate for identifying natural community structure in networks, primarily because the group sizes cannot be realistically fixed a priori, and choosing them arbitrarily will usually preclude us from finding the best solution to the problem.

On the other hand, letting the group sizes be free means that the minimum cut size will always be achieved by putting all vertices in one group and none in the other. This statement is considerably broader than the spectral partitioning method itself, since any method that minimizes the cut size without constraints on the number of vertices in each group is sure to find this same trivial division.

Several variations have been proposed to get around this problem, for example by favoring balanced divisions [WC91]. However, these “biased” approaches effectively choose the group sizes, at least approximately.

The fundamental problem with all these methods, however, is that merely optimizing the cut size is not enough: a good division of a network into communities is not one in which the number of inter-group edges is small, but rather one in which the number of such edges is smaller than expected. Only if this number is significantly lower than it would be expected purely by chance can we claim to have found significant community structure.

These considerations lead us to shift our attention from measures based on pure cut size to a modified objective function, the modularity $Q$.

2.1. Modularity. The modularity $Q$ is a function of a particular partition of a network into communities, defined as the number of edges within communities minus the expected number of such edges [NG04]. The precise definition is as follows [New06a].

**Definition 3.3 (Modularity).** Given a symmetric matrix $P = P^T$ of order $n$, whose elements $p_{ij}$ specify the expected number of edges between every pair of vertices $v_i$ and $v_j$, and the symmetric adjacency matrix $A = A^T$, whose elements
2. SPECTRAL OPTIMIZATION OF MODULARITY

\( a_{ij} \) specify the actual number of edges between every pair of vertices \( v_i \) and \( v_j \), we define the modularity \( Q \) as

\[
Q = \frac{1}{a} \sum_{i,j} (a_{ij} - p_{ij}) \delta_c (i, j)
\]

where \( \delta_c (i, j) = 1 \) if the two vertices \( v_i \) and \( v_j \) belong to the same community, and 0 otherwise.

Let us now focus on the second term in the definition of the modularity, the expected number of edges between every pair of vertices \( v_i \) and \( v_j \), and characterize it precisely.

First, we require that \( Q = 0 \) when all vertices are placed in a single group together, since the number of edges within the single group and the expected number of such edges must both be equal to \( m \) in this case. This leads to the constraint \( p = a = 2m \), which restricts our choice to null models in which the expected number of edges in the entire network equals the actual number of edges in the original network.

Moreover, the null model ought to have a degree distribution similar to that of the network being analyzed. To satisfy this requirement, we restrict our attention to models in which the expected and actual degree of each vertex are equal, \( p_i = a_i \).

The simplest null model in this class is that in which edges are placed entirely at random, connecting vertices with a probability that depends only on their degrees. The elements of the symmetric matrix \( P \) are thus defined as

\[
p_{ij} = \frac{a_i a_j}{a}
\]

Given the null model matrix \( P \), the modularity \( Q \) can be written as

\[
Q = \frac{1}{a} \sum_{i,j} \left( a_{ij} - \frac{a_i a_j}{a} \right) \delta_c (i, j)
\]

**Proposition 3.1.** For unweighted graphs, we have \(-1/2 \leq Q \leq 1 \) \cite{BDG08}.

Larger values of \( Q \) indicate stronger community structure. Therefore, we should in principle be able to find good partitions of a network into communities by optimizing \( Q \) over all possible partitions.

Even relatively small graphs, however, have huge partition spaces, making exhaustive exploration impossible. Indeed, modularity optimization is a NP-complete problem \cite{BDG08}, which makes it infeasible in all but the smallest instances.

A number of approximate solution methods have thus been developed. In practical applications a joint strategy is often adopted: firstly, spectral optimization is used to obtain a broad picture of the partitioning into communities; secondly, a refinement step, such as the extremal optimization method presented in \cite{DA05}, is performed.

**Remark.** More recently, it has been realized that the modularity approach is insufficient. Although it is true that networks with strong community structure have high modularity, it has been observed that not all networks with high modularity have strong community structure. This means that high modularity is only a necessary but not sufficient condition for significant community structure. An interesting approach to testing the significance of community assignments, along with a brief review of previous proposals, is described in \cite{KLN08}.

**Remark.** The modularity \( Q \) has the same form as the Hamiltonian of an Ising spin glass \cite{SK75}; in particular, the general form of the modularity for an arbitrary number of communities is equivalent to a Potts spin glass, a connection that has been exploited to create a community identification algorithm \cite{RB06}.
2. SPECTRAL OPTIMIZATION OF MODULARITY

2.2. Spectral optimization of modularity. An approach that relies on spectral analysis to optimize the modularity $Q$ has been presented in [New06a]. We start by defining the modularity matrix $B$, which plays a role analogous to that of the Laplacian matrix $L$ in spectral partitioning.

**Definition 3.4 (Modularity matrix).** The symmetric *modularity matrix* $B = B^T$ of order $n$, defined as $B = A - P$, has elements of the form

$$b_{ij} = a_{ij} - p_{ij} = a_{ij} - \frac{a_i a_j}{a}$$

**Remark.** All row (or column) sums of $B$ are equal to zero, since

$$\sum_j b_{ij} = \sum_j a_{ij} - \sum_j p_{ij} = a_i - a_i = 0$$

This immediately implies that, like the Laplacian matrix, $B$ always has the unit eigenvector $(1/\sqrt{n}, 1/\sqrt{n}, \ldots)$ with associated zero eigenvalue. Unlike the Laplacian matrix, however, the eigenvalues of $B$ are not necessarily all of one sign.

### 2.2.1. Leading eigenvector method.

Proceeding by analogy with Section 1.1.1, we can rewrite $Q$ in matrix form as

$$Q = \frac{1}{2a} h^T B h$$

where the index vector $h$ can be written as usual as a linear combination of the normalized eigenvectors $u_i$, $i = 1, \ldots, n$ of $B$,

$$h = \sum_i \alpha_i u_i$$

where as before $\alpha_i = u_i^T h$ and $\sum_i \alpha_i^2 = n$. Moreover, we have

$$Q = \frac{1}{2a} \sum_i \alpha_i^2 \lambda_i$$

where $\lambda_i$ is the eigenvalue of $B$ corresponding to the eigenvector $u_i$, $i = 1, \ldots, n$. Without loss of generality, we assume that the eigenvalues are labeled in *decreasing* order, $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$.

The task of maximizing $Q$ is then equivalent to that of choosing the non-negative quantities $\alpha_i^2$ so as to place as much weight as possible in the terms corresponding to the largest $\lambda_i$.

As with spectral partitioning, this task would be easily solved by choosing $h$ proportional to the *leading* eigenvector $u_1$ of $B$. However, the elements of $h$ must take the values $\pm 1$, which means that $h$ cannot normally be chosen parallel to $u_1$.

Good approximate solutions can again be obtained by choosing $h$ to be as close to parallel to $u_1$ as possible, which is achieved by setting

$$h_i = \begin{cases} 1 & \text{if } (u_1)_i \geq 0 \\ -1 & \text{if } (u_1)_i < 0 \end{cases}$$

where $(u_1)_i$ is the $i$th element of $u_1$.

**Remark.** The magnitudes of the elements of $u_1$ also contain useful information, indicating the “strength” with which vertices belong to the various communities, see [New06b].

Summarizing, the algorithm proceeds as follows:
2. SPECTRAL OPTIMIZATION OF MODULARITY

(1) Find the eigenvector \( u_1 \) corresponding to the most positive eigenvalue \( \lambda_1 \) of the modularity matrix \( B \).

(2) Divide the network into two groups according to the signs of the elements of \( u_1 \).

2.2.2. Multiple eigenvectors method. The algorithm described in the previous section has two obvious shortcomings. First, it divides networks into only two communities. Second, it uses only the leading eigenvector \( u_1 \) of the modularity matrix \( B \), potentially throwing away useful information contained in the other eigenvectors. Both of these shortcomings are remedied by generalizing the method as follows.

Remark (Indices). Indices that refer to entries referring to vertices will be in lowercase, while those that address entries referring to communities will be in uppercase.

Definition 3.5 (Membership matrix). The membership matrix \( H \in \{0,1\}^{n \times K} \), representing an assignment of \( n \) vertices to \( K \leq n \) communities, has elements

\[
h_{iJ} = \begin{cases} 1 & \text{if vertex } i \text{ belongs to community } J \\ 0 & \text{otherwise} \end{cases}
\]

where each column is a membership vector with binary \( \{0,1\} \) components describing a single community, \( H = (h_1|h_2|\ldots|h_K) \).

Remark. The columns of \( H \) are mutually orthogonal, its rows each sum to unity, and the matrix satisfies the normalization condition \( \text{Tr}(H^T H) = n \), since

\[
\delta_c(i,j) = \sum_K h_{iK} h_{jK}
\]

Substituting this expression in the modularity definition we obtain

\[
Q = \frac{1}{a} \sum_{i,j} b_{ij} h_{iK} h_{jK}
\]

Writing \( B = U D U^T \), where \( U = (u_1|u_2|\ldots|u_n) \) is the matrix of eigenvectors of \( B \), and \( D \) is the diagonal matrix of its eigenvalues, \( d_{ii} = \lambda_i \), we find

\[
Q = \frac{1}{a} \sum_{i,j} \lambda_i (u_i^T h_j)^2
\]

A choice of \( K \) communities corresponds then to choosing \( K - 1 \) independent, mutually orthogonal columns \( h_1, h_2, \ldots, h_{K-1} \), since the last column \( h_K \) is fixed by the condition that all rows of \( H \) sum to unity.

The modularity \( Q \) would then be maximized by choosing the columns proportional to the leading eigenvectors of \( B \). However, only eigenvectors corresponding to positive eigenvalues can contribute positively to \( Q \), so the optimal modularity would be achieved by choosing as many independent columns of \( H \) as there are positive eigenvalues, or equivalently by choosing \( K \) to be one greater than the number of positive eigenvalues.

Nevertheless, the problem has the additional constraint that the membership vectors \( h_j \) can only take the values \( \{0,1\} \). According to [New06a], this means that the number of positive eigenvalues plus one is only an upper bound on the number of communities, since it might not be possible to find as many membership vectors positively contributing to the modularity as the set of positive eigenvalues suggests.
Newman then recasts the problem of community identification into a vector partitioning problem, but does not in fact present a specific algorithmic proposal for solving it (such an algorithm can, however, be found in [WSO08]). Instead, Newman takes the approach of repeated subdivision into two communities, starting from a single community encompassing the entire network, and partitioning it into smaller and smaller communities. The algorithm explicitly computes the modularity contribution $\Delta Q$ for all possible subdivisions of a given community, declining to subdivide it if no subdivision exists that will increase the modularity of the network as a whole (communities with the property of having no subdivision such that $\Delta Q > 0$ are said to be indivisible).

### 2.3. Multidimensional spectral analysis.

The method presented in [DM04] exploits spectral properties of the Laplacian matrix $L$, combined with hierarchical clustering techniques, in an attempt to maximize the modularity $Q$.

The idea behind this method is that, if there are more than two weakly-connected components, it should be possible to identify them by inspecting the eigenvalue spectrum more accurately, instead of considering just the first non-trivial eigenvector as in Section 1.1.1. This is particularly relevant when the number of edges between different groups is large, which results in communities becoming more entangled and difficult to extract from the first non-trivial eigenvector.

This difficulty can be circumvented by enlarging the projection space to include the first $d$ non-trivial eigenvectors, and representing each vertex by a point in this $d$-dimensional space with coordinates corresponding to the individual eigenvector components (Figures 2.3 and 2.3).

Intuitively, vertices close in this $d$-dimensional space belong to the same community; therefore, by defining an appropriate similarity metric, we can apply standard hierarchical clustering techniques [JD88, ELL09], which proceed by removing all edges from the network and using a greedy optimization (hill climbing) approach...
that adds them back one by one in order of their similarity metric value, thus building a dendrogram (bottom up) detailing where each joining took place. The authors consider both Euclidean distance and angular distance, noting that the latter often outperforms the former.

Summarizing the ideas introduced so far, the algorithm proceeds as follows:

1. Perform spectral analysis of the Laplacian matrix $L$, projecting vertices into a space of (variable) dimensionality $d$
2. Define some similarity metric and apply standard hierarchical clustering techniques, obtaining a dendrogram
3. Maximize the modularity $Q_d$ for the $d$-dimensional space over all possible partitions recorded in the dendrogram
4. Repeat, maximizing the modularity $Q$ over all possible dimensions $d = 1, \ldots, n - 1$ of the eigenvector space

Remark. The maximum dimension is $n - 1$, and not $n$, since we do not need to consider the first, trivial eigenvalue $\lambda_1 = 0$.

The eigenvector computation is the most computer-intensive part of the algorithm, requiring $O(n^3)$ time. However, since the Laplacian matrix is usually sparse and not all eigenvectors are required, the faster Lanczos method [GL96] can be used.
CHAPTER 4

Other methods of modularity optimization

As described in Section 2.1, the modularity measure is a way to quantitatively evaluate the appropriateness of a partition of the graph. We have described modularity optimization methods using spectral analysis in Sections 2.2 and 2.3.

The problem of modularity optimization, however, can be tackled in other ways, which have proven highly effective in practice and which are presented in this chapter.

In the following sections we first analyze an Integer Linear Programming formulation of the problem, then we move to two other methods for modularity optimization based on classical greedy approaches.

We shall always keep in mind that, since maximizing \( Q \) over the set of all possible partitions is a global optimization problem, there is no guarantee that a greedy algorithm such as those presented in this section will effectively find the partition with maximum modularity, especially in large search spaces.

Indeed, for the method presented in Section 3 below, there exists a class of graphs on which the algorithm obtains a partition of modularity 0, but for which the optimum partition has a modularity value close to \( 1/2 \) [BDG’08].

1. Integer Linear Programming

The problem of maximizing modularity can be cast into a very simple and intuitive Integer Linear Programming (ILP) problem [BDG’08].

Given a graph with \( n \) vertices, we define \( n^2 \) decision variables \( x_{ij} \in \{0, 1\} \), one for every pair of vertices \( v_i \) and \( v_j \). These variables can be interpreted as an equivalence relation over the set of vertices \( V \), and thus describe a partition of the graph.

To ensure consistency, we enforce the following constraints for all vertices \( v_i, v_j, v_k \in V \), with \( i, j, k = 1, \ldots, n \),

\[
\begin{align*}
  x_{ii} &= 1 & \text{(reflexivity)} \\
  x_{ij} &= x_{ji} & \text{(symmetry)} \\
  x_{ij} + x_{jk} - 2x_{ik} & \leq 1 & \text{(transitivity)} \\
  x_{ik} + x_{ij} - 2x_{jk} & \leq 1 \\
  x_{jk} + x_{ik} - 2x_{ij} & \leq 1
\end{align*}
\]

The objective function \( Q \) then becomes

\[
Q = \frac{1}{a} \sum_{i,j} \left( a_{ij} - \frac{a_i a_j}{a} \right) x_{ij}
\]

Remark. This ILP problem can be simplified by pruning redundant variable and constraints, leaving only \( \binom{n}{2} \) variables and \( \binom{n}{3} \) constraints.
2. Simulated annealing

An algorithm for modularity optimization based on simulated annealing [Kir07] has been presented in [MD05].

The algorithm proceeds as follows:

1. At the start, place each vertex into a community of its own and set a high starting temperature.
2. At each step, randomly choose a vertex and a community (which could be any of the communities the vertex does not currently belong to, or an empty one), and compute the change in $Q$, denoted $\Delta Q$, should the vertex be moved to that community.
3. If $\Delta Q > 0$, always accept the move; otherwise, accept the move with probability $e^{\beta \Delta Q}$, where $\beta$ is the inverse temperature.
4. Repeat, decreasing the temperature, until $Q$ becomes (asymptotically) constant because no further moves are accepted (all $\Delta Q$s are negative and their probability is too low).

To increase the likelihood of finding the global optimum, it is also possible to periodically quench the system. This is done by calculating $\Delta Q$ for all possible moves of each vertex to each community, and taking the move with the highest $\Delta Q$.

3. Agglomerative hierarchical clustering

3.1. Community decomposition of the modularity. We first need to introduce a different definition for the modularity $Q$, which we shall subsequently prove to be equivalent to Definition 3.3.

Remark (Indices). Indices that refer to entries referring to vertices will be in lowercase, while those that address entries referring to communities will be in uppercase.

Definition 4.1 (Graph partition). The graph partition $H$ is a collection of sets $\{S_I, I = 1, \ldots, K\}$ representing a partition of a graph $G = (V, A)$ into $K$ non-empty and disjoint communities.

For all $S_I, S_J \in H$ with $I \neq J$ it holds that $S_I \subseteq V$ and $S_I \cap S_J = \emptyset$.

Furthermore, we have that

$$\bigcup_I S_I = V$$

Definition 4.2 (Community adjacency matrix). Consider a particular partition of the graph into $K$ communities and define the symmetric matrix $E = E^T$ of order $K$, whose elements $e_{I,J}$ are the fractions of all edges in the original graph that connect vertices in community $S_I$ to vertices in community $S_J$,

$$e_{I,J} = \frac{1}{d} \sum_{i \in S_I} \sum_{j \in S_J} a_{ij}$$
The row (or column) sums $e_I = \sum_J e_{IJ}$ then represent the fractions of all edges incident to vertices in community $S_I$,

$$e_I = \sum_J e_{IJ}$$

$$= \sum_J \left( \frac{1}{a} \sum_{i \in S_I} \sum_{j \in S_J} a_{ij} \right)$$

$$= \frac{1}{a} \sum_{i \in S_I} \left( \sum_{j \in S_J} a_{ij} \right)$$

$$= \frac{1}{a} \sum_{i \in S_I} \left( \sum_{j} a_{ij} \right)$$

$$= \frac{1}{a} \sum_{i \in S_I} a_i$$

This result implies the additivity of $e_I$, i.e. $e_{I \cup J} = e_I + e_J$, plus also that $\sum_I e_I = 1$.

**Theorem 4.1 (Community decomposition of the modularity).** The modularity measure introduced in Definition 3.3 can be written as

$$Q = \sum_I (e_{II} - e_I^2)$$

**Proof.** Let us start by recalling Definition 3.3, in which we defined $Q$ as

$$Q = \frac{1}{a} \sum_{i,j} \left( a_{ij} - \frac{a_i a_j}{a} \right) \delta_c(i, j)$$

The term $\delta_c(i, j)$ makes sure that the only elements being summed are those for which the corresponding vertices $v_i$ and $v_j$ belong to the same community. This can be equivalently expressed by summing over all intra-community edges for all communities, yielding

$$Q = \frac{1}{a} \sum_I \sum_{i,j \in S_I} \left( a_{ij} - \frac{a_i a_j}{a} \right)$$

$$= \sum_I \left( \frac{1}{a} \sum_{i,j \in S_I} a_{ij} - \left( \frac{1}{a} \sum_{i \in S_I} a_i \right)^2 \right)$$

Substituting the expression for the elements $e_{IJ}$ of the community adjacency matrix $E$, we obtain directly

$$Q = \sum_I (e_{II} - e_I^2)$$

□

This result can be interpreted as follows. In analogy with the construction of the null model associated with the adjacency matrix $A$, the product value $e_{I \cap J}$ can be regarded as the expected joint probability that an edge begins at a vertex in $S_I$ and ends at a vertex in $S_J$. Correspondingly, the expected fraction of edges inside community $S_I$ is then simply $e_I^2$. 

---

**3. AGGLOMERATIVE HIERARCHICAL CLUSTERING**
On the other hand, we know that the real fraction of edges between community $S_I$ and community $S_J$ is $e_{IJ}$, and the real fraction of edges inside community $S_I$ is $e_{II}$.

In this way, the community decomposition of the modularity $Q = \sum_I (e_{II} - e_{I}^2)$ corresponds to the application of the null model principle to each individual community level.

### 3.2. The agglomerative hierarchical clustering algorithm

The attempt at direct modularity maximization presented in [New04b] falls in the general category of agglomerative hierarchical clustering methods [JD88, ELL09].

Instead of starting with a single community encompassing the whole network and considering how to partition it into its constituent communities, one can take a different perspective on the problem: one can start with all vertices belonging to a community of their own, and join, or agglomerate, vertices which are likely to be in the same community, in a hill climbing fashion.

The algorithm described in [New04b] proceeds as follows:

1. At the start, place each vertex $v_i$ in a community $S_I$ of its own
2. At each step, compute the change in $Q$ should any two communities $S_I$, $S_J$ be joined, $\Delta Q_{IJ} = 2(e_{IJ} - e_{I}e_{J})$ as derived below, and join the pair producing the largest increase (or smallest decrease) in modularity
3. Repeat until the whole network has been reconstructed as a single community

As the algorithm proceeds, it is possible to construct a dendrogram (bottom up), recording where each joining took place.

Since the change in $Q$ upon joining two communities $S_I$ and $S_J$, $\Delta Q_{IJ}$, can be computed in constant time, this algorithm is very efficient and scales as $\mathcal{O}((m + n) n)$.  

#### 3.2.1. Improved version

A second, improved version of the same algorithm has been presented in [CNM04], and rests on the idea that, even though $\Delta Q_{IJ}$ can be computed in constant time, doing so for all pairs of communities $S_I$ and $S_J$ and finding the largest value becomes time-consuming.

Therefore, rather than maintaining the matrix $E$, the algorithm maintains and updates a matrix $\Delta$ with elements $\delta_{IJ} = \Delta Q_{IJ}$, additionally indexing the largest element in each row and in the whole matrix for fast retrieval.

The algorithm proceeds as follows:

1. At the start, place each vertex $v_i$ in a community $S_I$ of its own, compute the initial value of $Q$, initialize the matrix $\Delta$ with elements
   \[
   \delta_{IJ} = \begin{cases} 
   \frac{2a_{ij}}{a} - \frac{2a_i a_j}{a^2} & \text{if } v_i \text{ and } v_j \text{ are connected} \\
   0 & \text{otherwise}
   \end{cases}
   \]
   and initialize the vector $e$ as $e_I = a_i/a$

   **Remark.** The initialization $\delta_{IJ} = 0$ when $v_i$ and $v_j$ are not connected is more efficient than the plain $\delta_{IJ} = -(2a_i a_j)/a^2$, as it renders the matrix $\Delta$ sparse, and does not affect the final result.

2. At each step, find the maximal element $\delta_{IJ}$ of $\Delta$, join the corresponding communities $S_I$ and $S_J$, update $\Delta$ and its indexes, and add $\delta_{IJ}$ to $Q$
3. Repeat until the whole network has been reconstructed as a single community
Let us now derive the update rules for $\Delta$.
Consider two communities $S_I$ and $S_J$ and their respective contributions $Q_I$ and $Q_J$ to the modularity.

Labeling $S_{(I\cup J)}$ the combined community $S_I \cup S_J$, we have

$$Q_{(I\cup J)} = e_{(I\cup J)(I\cup J)} - e_{(I\cup J)(I\cup J)}^2$$

$$= (e_I + e_{IJ} + 2e_J) - (e_I + e_J)^2$$

$$= e_I + e_{IJ} + 2e_J - e_I^2 - e_J^2 - 2e_Ie_J$$

The change in modularity upon joining $S_I$ and $S_J$ is thus given by

$$\delta_{IJ} = Q_{(I\cup J)} - Q_I - Q_J$$

$$= e_I + e_{IJ} + 2e_J - e_I^2 - e_J^2 - 2e_Ie_J - e_{IJ} + e_{IJ}^2$$

$$= 2e_Ie_J$$

which is the expression used for the initialization of $\delta_{IJ} = \Delta Q_{IJ}$ in the second step of the algorithm.

Consider now the joining of $S_{(I\cup J)}$ with a third community $S_K$. We have

$$\delta_{(I\cup J)K} = Q_{(I\cup J\cup K)} - Q_{(I\cup J)} - Q_K$$

$$= e_{(I\cup J\cup K)(I\cup J\cup K)} - e_{(I\cup J\cup K)(I\cup J\cup K)}^2 - Q_{(I\cup J)} - Q_K$$

$$= e_{(I\cup J\cup K)(I\cup J)} + e_{KK} + 2e_{(I\cup J\cup K)K} - [(e_I + e_J + e_K)^2$$

$$- e_{(I\cup J\cup K)(I\cup J)} + (e_I + e_J)^2 - e_{KK} + e_{KK}^2$$

$$= 2e_{(I\cup J\cup K)K} - 2(e_I + e_J + e_K)$$

$$= 2(e_{IK} + e_{JK}) - 2(e_I + e_J)e_K$$

$$= 2(e_{IK} - 2e_Ie_K + 2e_K - 2e_Je_K)$$

$$= \delta_{IJ} + \delta_{JK}$$

where we have made use of

$$2e_{(I\cup J\cup K)K} = 2(e_{IK} + e_{JK})$$

since edges between $S_K$ and $S_{(I\cup J)}$ are actually between $S_K$ and either $S_I$ or $S_J$.

This yields the following update conditions for a community $S_K \neq S_I, S_J$

$$\delta'_{(I\cup J)K} = \begin{cases} 
\delta_{IK} + \delta_{JK} & \text{if } S_K \text{ is connected to both } S_I \text{ and } S_J \\
\delta_{IK} - 2a_Ia_K & \text{if } S_K \text{ is connected to } S_I \text{ but not to } S_J \\
\delta_{JK} - 2a_Ia_K & \text{if } S_K \text{ is connected to } S_J \text{ but not to } S_I 
\end{cases}$$

Remark. The last two cases account for not setting $\delta_{IJ}$ for every pair of disconnected communities $S_I$ and $S_J$ in the initialization step of the algorithm, which was done to keep $\Delta$ sparse.

Furthermore, we need to update the vector $e$ as follows: $e'_j = e_I + e_J$ and $e'_I = 0$.

Remark. These equations imply that $Q$ has a single peak over the course of the algorithm, a property known as single-peakedness. This means that, after the largest $\delta_{IJ}$ becomes negative, $Q$ can only decrease, and the algorithm can thus be stopped.

This version of the algorithm is extremely efficient, scaling as $O(md \log n)$, where $d$ is the depth of the dendrogram. Furthermore, since many real-world networks are sparse and hierarchical, with $m \propto n$ and $d \propto \log n$, this reduces essentially to linear time, $O(n \log^2 n)$. 
This algorithm has been successfully applied in the study of a recommender network of books from a large on-line retailer, with more than 400,000 vertices and 2 million edges [CNM04]. The case study we present in Section 6 uses the same algorithm. 

We have also applied it to the karate club study of Zachary [Zac77], which we first encountered in Section 1. Note that the resulting network configuration, shown in Figure 4.1, corresponds to a modularity $Q \approx 0.38067$, whereas the partition originally obtained by Newman (see Section 1) only achieves a slightly lower score.
CHAPTER 5

Redefining modularity

1. Modularity matrix

1.1. Classical definition. The classical null model used in the definition of the modularity matrix \( B \) of Section 2.2 was introduced in [New06a] and briefly reviewed in Section 2.1.

It is defined as the symmetric matrix \( P = P^T \) of order \( n \) with elements

\[
  p_{ij} = \frac{a_i a_j}{a}
\]

It can be immediately seen that some of the information encoded in the adjacency matrix \( A \) is distributed along the diagonal elements \( p_{ii} \), associated to self-loops in the null model graph, which have no correspondence with the original graph described by the adjacency matrix \( A \).

For this reason, we will define below a new null model which complies with the general assumption that no self-loops are allowed.

1.2. New definition. To overcome the limitation expressed above, we can apply the diagonal diffusion operator described in Appendix A to the matrix \( P \), obtaining the new null model matrix \( R = \tilde{P} \) with off-diagonal elements \( r_{ij}, i \neq j \) of the form

\[
  r_{ij} = \frac{1}{a} \left[ a_i a_j + \frac{1}{n-2} \left( a_i^2 + a_j^2 - \frac{1}{n-1} \sum_i a_i^2 \right) \right]
\]

1.3. Properties of \( R \). In order to illustrate our newly-defined null model, let us now consider a number of cases for which \( R \) is an invariant. In these cases \( R = A \), and thus \( B = A - R = 0 \).

Remark. In what follows we refer to the general case of weighted graphs.

Theorem 5.1 (Graphs of order 3). For all graphs of order 3 it holds that \( R = A \).

Proof. Consider a graph with \( n = 3 \) vertices, whose adjacency matrix \( A \) has three degrees of freedom corresponding to the elements \( a_{12}, a_{13} \) and \( a_{23} \).
Substituting $n = 3$ in the general definition of $r_{ij}$, we obtain, with $k \neq i, j$:

$$r_{ij} = \frac{1}{a} \left(a_i a_j + a_i^2 + a_j^2 - \frac{1}{2} \sum_i a_i^2\right)$$

$$= \frac{1}{a} \left(a_i a_j + \frac{1}{2} a_i^2 + \frac{1}{2} a_j^2 - \frac{1}{2} a_k^2\right)$$

$$= \frac{1}{a} \left[\frac{1}{2} (a_i + a_j)^2 - \frac{1}{2} a_k^2\right]$$

$$= \frac{1}{2a} (a_i + a_j + a_k)(a_i + a_j - a_k)$$

$$= (a_i + a_j + a_k)(a_i + a_j - a_k)$$

$$= \frac{1}{2} (a_i + a_j - a_k)$$

For $r_{12}$, for instance, we have that

$$r_{12} = \frac{1}{2} (a_1 + a_2 - a_3)$$

$$= \frac{1}{2} (a_{12} + a_{13} + a_{12} + a_{23} - a_{13} - a_{23})$$

$$= a_{12}$$

It can be easily checked that $r_{ij} = a_{ij}$ for all the remaining elements, leading to the conclusion $R = A$. $\square$

**Theorem 5.2 (Uniform graphs).** For all uniform graphs, i.e. graphs in which all edges have the same weight, it holds that $R = A$.

**Proof.** Consider a graph with $n$ vertices, all of which are connected to each other with uniform edge weight $v$.

The uniform graph is represented by an adjacency matrix of the form

$$A = \begin{bmatrix}
0 & v & v & \cdots & v \\
v & 0 & v & \cdots & v \\
v & v & 0 & \vdots & \\
\vdots & \vdots & \ddots & \ddots & v \\
v & v & \cdots & v & 0
\end{bmatrix}$$

Sums along any row take the form

$$a_i = (n - 1)v$$

from which we can easily derive

$$a = \sum a_i = n(n - 1)v$$

and

$$\frac{1}{n-1} \sum_i a_i^2 = \frac{1}{n-1} n(n-1)^2 v^2$$

$$= n(n-1) v^2$$
Substituting in the general definition of \( r_{ij} \) yields

\[
r_{ij} = \frac{1}{n(n-1)} \left\{ (n-1)^2 \upsilon^2 + \frac{1}{n-2} \left[ 2(n-1)^2 \upsilon^2 - n(n-1) \upsilon^2 \right] \right\}
\]

\[
= \frac{1}{n} \left\{ (n-1) \upsilon + \frac{1}{n-2} (2n-2-n) \upsilon \right\}
\]

\[
= \frac{1}{n} (n-1) \upsilon
\]

\[
= \upsilon
\]

from which it follows that \( R = A \). □

**Theorem 5.3 (Graphs with a “loner”).** For all graphs with a “loner”, i.e. graphs containing a uniform subgraph of order \( n-1 \) whose vertices are connected to the “loner” vertex with the same edge weight, it holds that \( R = A \).

**Proof.** Consider a graph with \( n \) vertices, having \( n-1 \) vertices all connected to each other by edges of weight \( \upsilon \), and to the “loner” vertex \( v_l \) by edges of weight \( \zeta \neq \upsilon \).

**Remark.** Uniform graphs can be considered a special case in which we allow \( \zeta = \upsilon \).

The “loner” graph is represented by an adjacency matrix of the form

\[
A = \begin{bmatrix}
0 & \upsilon & \upsilon & \cdots & \upsilon & \zeta & \upsilon & \cdots & \upsilon & \upsilon \\
\upsilon & 0 & \upsilon & \cdots & \upsilon & \zeta & \upsilon & \cdots & \upsilon & \upsilon \\
\upsilon & \upsilon & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
& & & & & & & & & \\
\upsilon & \upsilon & \cdots & 0 & \zeta & \upsilon & \cdots & \upsilon & \upsilon & \\
\zeta & \zeta & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
\upsilon & \upsilon & \cdots & \upsilon & \zeta & 0 & \cdots & \upsilon & \upsilon & \\
& & & & & & & & & \\
\zeta & \zeta & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
& & & & & & & & & \\
\upsilon & \upsilon & \cdots & \upsilon & \zeta & \upsilon & \cdots & \upsilon & \upsilon & \\
& & & & & & & & & \\
\end{bmatrix}
\]

Sums along any row but \( l \) take the form

\[
a_{i \neq l} = \zeta + (n - 2) \upsilon
\]

whereas row \( l \) has sum

\[
a_l = (n - 1) \zeta
\]

We can now easily compute

\[
a = \sum_i a_i
\]

\[
= a_l + \sum_{i \neq l} a_i
\]

\[
= (n - 1) \zeta + (n - 1) \left[ \zeta + (n - 2) \upsilon \right]
\]

\[
= (n - 1) \zeta + (n - 1) \zeta + (n - 1) (n - 2) \upsilon
\]

\[
= 2 (n - 1) \zeta + (n - 1) (n - 2) \upsilon
\]
and
\[
\frac{1}{n-1} \sum_i a_i^2 = \frac{1}{n-1} \left( a_l^2 + \sum_{i \neq l} a_i^2 \right)
\]
\[
= \frac{1}{n-1} \left\{ (n-1) \zeta^2 + (n-1) \left[ \zeta + (n-2) \nu \right]^2 \right\}
\]
\[
= (n-1) \zeta^2 + \zeta^2 + (n-2)^2 \nu^2 + 2 (n-2) \zeta \nu
\]
\[
= n \zeta^2 + (n-2)^2 \nu^2 + 2 (n-2) \zeta \nu
\]

It is easy to prove that, since all row sums but \(a_l\) are equal, we only need to consider the following two cases in computing the values \(r_{ij}\) of \(R\):

1. vertex \(v_l\) with any other vertex, i.e. \(i = l, j \neq i\)
2. any two vertices different from \(v_l\), namely \(i \neq l \neq j\)

Let us now analyze independently the two cases.

**Lemma 5.1 (Vertex \(v_l\) with any other vertex).**

\[
r_{ij} = \frac{1}{\alpha} \left[ 2 (n-1) \zeta^2 + (n-1) (n-2) \zeta \nu \right]
\]
\[
= \frac{\zeta}{2} \left[ (n-1) \zeta + (n-1) (n-2) \nu \right]
\]
\[
= \zeta
\]

**Lemma 5.2 (Any two vertices different from \(v_l\)).**

\[
r_{ij} = \frac{1}{\alpha} \left[ 2 (n-1) \zeta \nu + (n-1) (n-2) \nu^2 \right]
\]
\[
= \frac{\nu}{2} \left[ (n-1) \zeta + (n-1) (n-2) \nu \right]
\]
\[
= \nu
\]

Thus, \(R = A\). \(\square\)

### 1.4. Comparison of the null models.

In what follows we shall compare the two null models in a number of sample cases with graphs of limited order, where all edge weights are chosen out of two possible values \(\alpha, \beta\).

We will exclude from our analysis graphs of order 3 (which are invariants for \(R\) according to Theorem 5.1), and all graphs with a “loner” (also invariants according to Theorem 5.3).

**Example 5.1 (Graph of order 4 divided in two communities of size 2).** The graph is shown in Figure 1.4. Its adjacency matrix is of the form

\[
A = \begin{bmatrix}
0 & \alpha & \beta & \alpha \\
\alpha & 0 & \alpha & \beta \\
\beta & \alpha & 0 & \alpha \\
\alpha & \beta & \alpha & 0
\end{bmatrix}
\]

where we assume \(\beta > \alpha\).

Analyzing \(A\) by means of spectral analysis, we obtain the following lists of eigenvalues \(\Lambda_P\) and \(\Lambda_R\), using as null model \(P\) or \(R\) respectively:

\[
\Lambda_P = \left\{ \beta - 2\alpha, -\beta, 0 \right\}
\]
\[
\Lambda_R = \left\{ \frac{4}{3} (\beta - \alpha), \frac{2}{3} (\alpha - \beta), 0 \right\}
\]
In both cases, multiplicities are $(1, 2, 1)$, and the eigenvector associated to the leading eigenvalue (the first one) is $(1, -1, 1, -1)$, a correct partitioning of the network in two communities.

However, for $\Lambda_P$ the value $\beta - 2\alpha$ is greater than zero only for $\beta > 2\alpha$, quite a strong requirement compared to our initial assumption $\beta > \alpha$. This is not the case for $\Lambda_R$, whose eigenvalue $4/3(\beta - \alpha)$ is positive exactly for $\beta > \alpha$.

It is also interesting to understand what happens if $\beta < \alpha$.

In this case, all eigenvalues in $\Lambda_P$ are non-positive, leaving a leading eigenvalue of 0 to signify that no community can be identified. For $\Lambda_R$, instead, the (second) eigenvalue $4/3(\alpha - \beta)$ is positive exactly when $\beta < \alpha$. Since this eigenvalue has multiplicity 2, we can generate from the associated eigenvectors $(1, 0, -1, 0)$ and $(0, 1, 0, -1)$ the two reasonable partitions of the original network in two communities: $(1, 1, -1, -1)$ and $(1, -1, -1, 1)$.
Example 5.2 (Graph of order 5 divided in two communities of size 2 and 3). The adjacency matrix for this graph is of the form
\[
A = \begin{bmatrix}
0 & \beta & \alpha & \alpha & \alpha \\ 
\beta & 0 & \alpha & \alpha & \alpha \\ 
\alpha & \alpha & 0 & \beta & \beta \\ 
\alpha & \alpha & \beta & 0 & \beta \\ 
\alpha & \alpha & \beta & \beta & 0
\end{bmatrix}
\]
where we assume $\beta > \alpha$.

Analysing $A$ by means of spectral analysis, we obtain the following lists of eigenvalues $\Lambda_P$ and $\Lambda_R$, using as null model $P$ or $R$ respectively:
\[
\Lambda_P = \left\langle \frac{5\beta^2 - 15\alpha^2}{4\beta + 6\alpha}, -\beta, 0 \right\rangle
\]
\[
\Lambda_R = \left\langle \frac{75\beta^2 + 50\alpha\beta - 125\alpha^2}{48\beta + 72\alpha}, \frac{25\alpha^2 - 10\alpha\beta - 15\beta^2}{16\beta + 24\alpha}, \frac{25\alpha^2 - 10\alpha\beta - 15\beta^2}{48\beta + 72\alpha}, 0 \right\rangle
\]
Multiplicities are $\langle 1, 3, 1 \rangle$ and $\langle 1, 1, 2, 1 \rangle$ respectively. In both cases, the eigenvector associated to the first eigenvalue is $\langle 1, 1, -2/3, -2/3, -2/3 \rangle$, a correct partitioning of the network in two communities.

For $\Lambda_P$, the leading eigenvalue (the first one) is greater than zero only for $\beta > \sqrt{3}\alpha$, still a strong requirement compared to our initial assumption $\beta > \alpha$. For $\Lambda_R$, instead, the leading eigenvalue is positive exactly for $\beta > \alpha$.

Let us now analyze what happens if $\beta < \alpha$.

In this case, all eigenvalues in $\Lambda_P$ are non-positive, leaving a leading eigenvalue of 0 to signify that no community can be identified. For $\Lambda_R$, instead, the second and third eigenvalues are positive exactly when $\beta < \alpha$. Using the eigenvector $\langle 1, -1, 0, 0, 0 \rangle$ associated with the second eigenvalue, and combining the eigenvectors $\langle 0, 0, 1, 0, -1 \rangle$ and $\langle 0, 0, 0, 1, -1 \rangle$ associated with the third eigenvalue, we can generate reasonable partitions of the original network.

Example 5.3 (Graph of order 6 divided in two communities of size 2 and 4). This graph configuration leads to results similar to what we already described in Example 5.2. The single possibly positive eigenvalue found using $P$ is greater than zero only for $\beta > 2\sqrt{3}\alpha$, whereas $R$ retains the original assumption $\beta > \alpha$.

Example 5.4 (Graph of order 6 divided in two communities of size 3). This situation is similar to the one we already analyzed in Example 5.1, and it does in fact behave similarly. For the leading eigenvalue to be positive, $P$ requires $\beta > 3/2\alpha$, whereas $R$ sticks to $\beta > \alpha$.

Example 5.5 (Graph of order 6 divided in three communities of size 2). This configuration also leads to the same result we had in Example 5.1, yielding exactly the same condition $\beta > 2\alpha$ for $P$, and $\beta > \alpha$ for $R$.

These results seem to corroborate the original motivation to redefine the null model, excluding self-loops, insofar as the information encoded in the self-loops seems to be at least partially disregarded.

2. Community decomposition of the modularity

Following a procedure similar to the one of Section 3.1, we give here the community decomposition of the redefined modularity.
Starting from the general definition of the elements \( r_{ij} \),

\[
Q = \frac{1}{a} \sum_{i \neq j} \left\{ a_{ij} - \frac{1}{a} \left[ a_i a_j + \frac{1}{n-2} \left( a_i^2 + a_j^2 - \frac{1}{n-1} \sum_k a_k^2 \right) \right] \right\} \delta_c(i,j)
\]

We notice that the term \( \delta_c(i,j) \), which makes sure that the only elements being summed are those for which the corresponding vertices \( v_i \) and \( v_j \) belong to the same community, can be equivalently expressed by summing exclusively over intra-community edges, yielding

\[
Q = \frac{1}{a} \sum_{i \neq j} \sum_{l \neq j \in S_l} \left\{ a_{ij} - \frac{1}{a} \left[ a_i a_j + \frac{1}{n-2} \left( a_i^2 + a_j^2 - \frac{1}{n-1} \sum_k a_k^2 \right) \right] \right\}
\]

Computing the various terms, we have

\[
\frac{1}{a} \sum_{i \neq j \in S_l} a_{ij} = \frac{1}{a} \sum_{i,j \in S_l} a_{ij} = c_{11}
\]

\[
\frac{1}{a^2} \sum_{i \neq j \in S_l} a_i a_j = \frac{1}{a^2} \sum_{i,j \in S_l} a_i a_j = e_1 = \frac{1}{a^2} \sum_{i \in S_l} a_i^2
\]

\[
\frac{1}{a^2} \sum_{i \neq j \in S_l} (a_i^2 + a_j^2) = \frac{1}{a^2} \sum_{i,j \in S_l} (a_i^2 + a_j^2) = 2 \left( \frac{n_l}{a^2} \right) \sum_{i \in S_l} a_i^2
\]

\[
\frac{1}{a^2} \sum_{i \neq j \in S_l} \left( \sum_k a_k^2 \right) = \frac{n_l}{a^2} \sum_{i \in S_l} a_i^2 + \frac{n_l (n_l - 1)}{a^2} \sum_{i \in S_l^C} a_i^2
\]

where \( n_l \) is the cardinality of community \( S_l \), and \( S_l^C \) denotes the complementary community with cardinality \( n_l^C \).

Substituting in the previous equation leads to

\[
Q = \sum_I \left\{ e_{11} - e_1^2 + \frac{1}{a^2} \left[ 1 - \frac{2(n_l - 1)}{n-2} \right] \sum_{i \in S_l} a_i^2 + \frac{1}{a^2} \left[ \frac{n_l (n_l - 1)}{(n-1)(n-2)} \right] \sum_{i \in S_l^C} a_i^2 \right\}
\]

\[
= \sum_I \left\{ e_{11} - e_1^2 + \frac{1}{a^2 (n-1)(n-2)} \left[ n_l^C (n_l^C - 1) \sum_{i \in S_l} a_i^2 + n_l (n_l - 1) \sum_{i \in S_l^C} a_i^2 \right] \right\}
\]

**Remark.** The new term associated with community \( S_l \) is strictly positive, being null only when \( n_l = n_l^C = 1 \), and symmetric.

We now prove an interesting property of the redefined modularity, which does not hold for the classical definition.
Theorem 5.4 (Modularity of the trivial partition into \( n \) singleton communities). The modularity of the trivial partition into \( K = n \) communities, obtained by placing each vertex \( v_i \in V \) in its own community \( S_I = \{v_i\} \), is null.

Proof. Assume that each vertex \( v_i \in V \) is placed in its own community \( S_I = \{v_i\} \). For all communities \( S_I \) we have that \( e_{II} = 0 \) and \( e_I = a_i/\alpha \). Furthermore, for all communities \( S_I \), we have that \( n_I = 1 \) and \( n^C_I = n - 1 \).

The contribution to the modularity \( Q_I \) of each community \( S_I \) is thus

\[
Q_I = e_{II} - e_I^2 + \frac{1}{\alpha^2(n-1)(n-2)} \left[ n^C_I (n^C_I - 1) \sum_{i \in S_I} a_i^2 + n_I (n_I - 1) \sum_{i \in S^C_I} a_i^2 \right]
\]

\[
= -\left( \frac{a_i}{\alpha} \right)^2 + \frac{1}{\alpha^2(n-1)(n-2)} \left[ (n-1)(n-2) \sum_{i \in S_I} a_i^2 \right]
\]

\[
= -\frac{a_i^2}{\alpha^2} + \frac{a_i^2}{\alpha^2}
\]

\[
= 0
\]

where we have used \( \sum_{i \in S_I} a_i^2 = a_i^2 \) since, for all communities \( S_I \), \( S_I = \{v_i\} \).

Therefore, since \( Q = \sum_I Q_I \), it follows that \( Q = 0 \). \( \square \)

3. Application

As for the improved version of the algorithm presented in Section 3, we can derive an update rule for \( \Delta \) for the redefined modularity, and use it to guide the algorithm.

Quite interestingly, the update rule turns out to be the same we found for the classical definition of modularity, namely

\[
\delta'_{(I\cup J)K} = \delta_{IK} + \delta_{JK}
\]

The only difference lies in the initialization rule for \( \Delta \), which in the case of the redefined modularity is given by

\[
\delta_I = \frac{2a_i}{\alpha} - \frac{2a_i a_j}{\alpha^2} - \frac{2}{\alpha^2(n-1)(n-2)} \left[ (n-1) \left( a_i^2 + a_j^2 \right) - \sum_i a_i^2 \right]
\]

In a preliminary study, where we applied the new null model approach to the karate club study of Zachary [Zac77] (Figure 5.3), we obtained a modularity \( Q \approx 0.39423 \), approximately 0.01356 higher than the one we found using the classical modularity in Section 3.

This result suggests that the redefined modularity is indeed a better heuristics for the agglomerative hierarchical clustering algorithm.
Figure 5.3: The friendship network from Zachary’s karate club study [Zac77], analyzed by means of agglomerative hierarchical clustering guided by the redefined modularity, with vertices belonging to different communities drawn in different shapes and colors.
Case study: the scientific collaboration network of Italian computer scientists

The study of scientific collaboration networks is a common benchmark for community identification algorithms [New01c, New01a, New01b]. We applied the improved version of the algorithm described in Section 3 to a network we built from publicly available data. The results of this analysis are presented in the following sections.

1. Data collection

Data used in building the graph of scientific collaborations among Italian computer scientists has been collected from two databases, namely:

(1) Cercar Università\(^1\), from which we extracted information about university scholars (professors and researchers), their field of activity (settore scientifico-disciplinare) and the university they belong to

(2) The DBLP Computer Science Bibliography\(^2\), from which we extracted information about publications

Names in both databases were matched using a simple algorithm based on edit distance and checked manually to ensure consistency.

The graph thus obtained has \(n = 1741\) vertices, linked by \(m = 3878\) edges. It is divided in 89 connected components, with the giant central component encompassing 1455 vertices, more than 80\% of the whole graph. There are also 66 small components of order 2 and 3.

2. Degree distribution

The most basic statistical characterization of a graph is given by the study of the vertex degrees \(a_i\), or their relative probability distribution \(P(k)\), which is defined for undirected graphs as the probability that any randomly chosen vertex has degree \(k\).

Remark. We consider here a Boolean representation of the network, in which two authors are linked whenever they have co-authored at least one paper.

The graph of Figure 6.1 clearly shows that \(P(k)\) follows a heavy-tailed power-law distribution (characterizing a heterogeneous connectivity pattern) \(k^{-\gamma}\) with \(\gamma \approx 2\). This in turn means that there is a finite probability of finding vertices whose degrees are much larger than the average \(\langle k \rangle\). For this reason, the average degree cannot be considered a characteristic scale for the system, and the network is said to be scale-free, as is typical of social networks [New03].

---

\(^1\)http://cercauniversita.cineca.it/ (in Italian)

\(^2\)http://dblp.uni-trier.de/
Figure 6.1: The actual degree distribution $P(k)$ compared with the power-law distribution $k^{-2}$

Figure 6.2: Change in the modularity value over the course of the algorithm

3. Community identification

The analysis was conducted on the weighted representation of the network, with every edge representing the number of papers co-authored by the two authors involved.

In order to weigh single collaborations more than multiple ones, as well as to keep all edge weights within the interval $[0, 1]$, the number of co-authored papers was filtered through the sigmoid function

$$\sigma(x) = \frac{1}{1 - e^{-x}}$$
Starting from an initial modularity $Q \approx -0.00095$ and applying the improved version of the algorithm detailed in Section 3, we obtained a final modularity value $Q \approx 0.80727$. The change in the modularity value over the course of the algorithm is shown in Figure 6.2.

The graph has been divided into 124 communities. Communities belonging to the giant connected component are shown in Figure 6.3, in which each node represents a community. Communities having the highest degree $k_{\text{max}} = 9$ are drawn in color.

Figure 6.4 shows how many communities include members from a certain number of different universities. For example, there are 12 communities containing authors from one single university, and there are 44 communities containing authors from two different universities. We note that around 45% of the communities contain authors from at most two universities, and more than 60% contain authors from at most three.

Figure 6.5 depicts a similar study, in which we consider instead the number of different SSDs (*settori scientifico-disciplinari*).

For example, there are 38 communities containing authors from one single SSD, and there are 39 communities containing authors from two different SSDs. In this
Figure 6.4: Number of communities containing a certain number of different universities

Figure 6.5: Number of communities containing a certain number of different SSDs

case, we note that more than 60% of the communities contain authors belonging to at most two SSDs, and almost 80% contain authors belonging to at most three.
APPENDIX A

Diagonal diffusion operator

We define a new linear operator $\sim$ acting on symmetric matrices $A = A^T$ of order $n$. The operator maps any symmetric matrix $A$ into the corresponding symmetric matrix $\tilde{A} = \tilde{A}^T$ of order $n$ satisfying the following properties for all $i, j = 1, \ldots, n$:

(A.1) $\tilde{a}_{ii} = 0$ (zero diagonal)

(A.2) $\tilde{a}_{ij} = a_{ij} \quad$ (preservation of row sums)

**Definition A.1 (Diagonal diffusion operator $\sim$).** Let $A = A^T$ be a symmetric matrix of order $n$. We denote $\hat{a} = \sum_i a_{ii}$. By linearly spreading the diagonal elements of $A$ over its off-diagonal positions, while keeping row sums intact, we define the symmetric, zero-diagonal matrix $\tilde{A}$ with elements

$$\tilde{a}_{ij} = \begin{cases} a_{ij} + \alpha (a_{ii} + a_{jj}) + \frac{1}{n} \beta \hat{a} & i \neq j \\ 0 & i = j \end{cases}$$

and we find suitable values for $\alpha$ and $\beta$ by imposing condition (A.2) on the preservation of row sums. Note that, by construction, $\tilde{A}$ is symmetric and zero-diagonal, thus also satisfying property (A.1).

Summing all elements in row $i$ of $\tilde{A}$ gives

$$\tilde{a}_i = \sum_j \tilde{a}_{ij} = \sum_{j \neq i} \tilde{a}_{ij} \quad (\text{since, for all } v_i \in V, \tilde{a}_{ii} = 0)$$

$$= (a_i - a_{ii}) + \alpha (n - 1) a_{ii} + \alpha (n\hat{a} - a_i) + \beta (n - 1) \frac{\hat{a}}{n}$$

$$= a_i + a_{ii} [\alpha (n - 2) - 1] + \hat{a} \left( \frac{\alpha n + \beta (n - 1)}{n} \right)$$

Imposing $\tilde{a}_i = a_i$ leads to the following values for $\alpha$ and $\beta$,

$$\begin{cases} \alpha (n - 2) - 1 & = 0 \\ [\alpha n + \beta (n - 1)] / n & = 0 \end{cases} \quad \begin{cases} \alpha = \frac{1}{n-2} \\ \beta = \frac{1}{(n-1)(n-2)} \end{cases}$$

Elements of $\tilde{A}$ will thus be of the form

$$\tilde{a}_{ij} = \begin{cases} a_{ij} + \frac{1}{n-2} (a_{ii} + a_{jj}) - \frac{1}{(n-1)(n-2)} \hat{a} & i \neq j \\ 0 & i = j \end{cases}$$

**Theorem A.1.** The $\sim$ operator of Definition A.1 is linear, i.e. it satisfies the following properties for all symmetric matrices $A = A^T, B = B^T$ and all $\alpha \in \mathbb{R}$,

$$\tilde{(A + B)} = \tilde{A} + \tilde{B}$$

$$\tilde{(\alpha A)} = \alpha \tilde{A}$$
**First property.** Let \( \mathbf{A} = \mathbf{A}^T \) and \( \mathbf{B} = \mathbf{B}^T \) be symmetric matrices of order \( n \). We denote \( \hat{a} = \sum_i a_{ii} \) and \( \hat{b} = \sum_i b_{ii} \).

According to Definition A.1, off-diagonal elements of \( \hat{\mathbf{A}} \) are of the form
\[
\tilde{a}_{ij} = a_{ij} + \frac{1}{n-2} (a_{ii} + a_{jj}) - \frac{\hat{a}}{(n-1)(n-2)}
\]
and similarly for \( \tilde{\mathbf{B}} \).

Consider now the sum of \( \mathbf{A} \) and \( \mathbf{B} \), \( \mathbf{C} = \mathbf{A} + \mathbf{B} \). Clearly, \( \hat{c} = \hat{a} + \hat{b} \).

According to Definition A.1, off-diagonal elements of \( \hat{\mathbf{C}} \) are of the form
\[
\tilde{c}_{ij} = c_{ij} + \frac{1}{n-2} (s_{ii} + s_{jj}) - \frac{\hat{c}}{(n-1)(n-2)}
= (a_{ij} + b_{ij}) + \frac{1}{n-2} ((a_{ii} + b_{ii}) + (a_{jj} + b_{jj})) - \frac{\hat{a} + \hat{b}}{(n-1)(n-2)}
\]

Thus, \( \hat{\mathbf{C}} = \hat{\mathbf{A}} + \hat{\mathbf{B}} \). \( \square \)

**Second property.** Let \( \mathbf{A} = \mathbf{A}^T \) be a symmetric matrix of order \( n \), and \( \alpha \in \mathbb{R} \) any real number. We denote \( \hat{a} = \sum_i a_{ii} \).

According to Definition A.1, off-diagonal elements of \( \hat{\mathbf{A}} \) are of the form
\[
\tilde{a}_{ij} = a_{ij} + \frac{1}{n-2} (a_{ii} + a_{jj}) - \frac{\hat{a}}{(n-1)(n-2)}
\]
Consider now the symmetric matrix \( \mathbf{C} = \alpha \mathbf{A} \), for which we have \( \hat{c} = \sum_i \alpha a_{ii} = \alpha \sum_i a_{ii} = \alpha \hat{a} \).

According to Definition A.1, off-diagonal elements of \( \hat{\mathbf{C}} \) are of the form
\[
\tilde{c}_{ij} = \alpha a_{ij} + \frac{1}{n-2} (\alpha a_{ii} + \alpha a_{jj}) - \frac{\alpha \hat{a}}{(n-1)(n-2)}
= \alpha \left[ a_{ij} + \frac{1}{n-2} (a_{ii} + a_{jj}) - \frac{\hat{a}}{(n-1)(n-2)} \right]
= \alpha \tilde{a}_{ij}
\]

Thus, \( \hat{\mathbf{C}} = \hat{\alpha \mathbf{A}} = \alpha \hat{\mathbf{A}} \). \( \square \)
Bibliography


38
BIBLIOGRAPHY


