Foundations and Trends® in Machine Learning

Data Analytics on Graphs Part I: Graphs and Spectra on Graphs

Suggested Citation: Ljubiša Stanković, Danilo Mandic, Miloš Daković, Miloš Brajović, Bruno Scalzo, Shengxi Li and Anthony G. Constantinides (2020), "Data Analytics on Graphs Part I: Graphs and Spectra on Graphs", Foundations and Trends® in Machine Learning: Vol. 13, No. 1, pp 1–157. DOI: 10.1561/2200000078-1.

Ljubiša Stanković
University of Montenegro
Montenegro
ljubisa@ucg.ac.me

Danilo Mandic
Imperial College London
UK
d.mandic@imperial.ac.uk

Miloš Daković
University of Montenegro
Montenegro
milos@ucg.ac.me

Miloš Brajović
University of Montenegro
Montenegro
milosb@ucg.ac.me

Bruno Scalzo
Imperial College London
UK
bruno.scalzo-dees12@imperial.ac.uk

Shengxi Li
Imperial College London
UK
shengxi.li17@imperial.ac.uk

Anthony G. Constantinides
Imperial College London
UK
a.constantinides@imperial.ac.uk

now
the essence of knowledge
Boston — Delft

Contents

Data Analytics on Graphs Part I: Graphs and Spectra on Graphs

Ljubiša Stanković[1], Danilo Mandic[2], Miloš Daković[3], Miloš Brajović[4], Bruno Scalzo[5], Shengxi Li[6] and Anthony G. Constantinides[7]

[1] *University of Montenegro, Montenegro; ljubisa@ucg.ac.me*
[2] *Imperial College London, UK; d.mandic@imperial.ac.uk*
[3] *University of Montenegro, Montenegro; milos@ucg.ac.me*
[4] *University of Montenegro, Montenegro; milosb@ucg.ac.me*
[5] *Imperial College London, UK; bruno.scalzo-dees12@imperial.ac.uk*
[6] *Imperial College London, UK; shengxi.li17@imperial.ac.uk*
[7] *Imperial College London, UK; a.constantinides@imperial.ac.uk*

ABSTRACT

The area of Data Analytics on graphs promises a paradigm shift, as we approach information processing of new classes of data which are typically acquired on irregular but structured domains (such as social networks, various ad-hoc sensor networks). Yet, despite the long history of Graph Theory, current approaches tend to focus on aspects of optimisation of graphs themselves rather than on eliciting strategies relevant to the objective application of the graph paradigm, such as detection, estimation, statistical and probabilistic inference, clustering and separation from signals and data acquired on graphs. In order to bridge this gap, we first revisit graph topologies from a Data Analytics point of view, to establish a taxonomy of graph networks through a linear algebraic formalism of graph topology (vertices, connections, directivity). This serves as a basis for spectral

---

Ljubiša Stanković, Danilo Mandic, Miloš Daković, Miloš Brajović, Bruno Scalzo, Shengxi Li and Anthony G. Constantinides (2020), "Data Analytics on Graphs Part I: Graphs and Spectra on Graphs", Foundations and Trends® in Machine Learning: Vol. 13, No. 1, pp 1–157. DOI: 10.1561/2200000078-1.

analysis of graphs, whereby the eigenvalues and eigenvectors of graph Laplacian and adjacency matrices are shown to convey physical meaning related to both graph topology and higher-order graph properties, such as cuts, walks, paths, and neighborhoods. Through a number of carefully chosen examples, we demonstrate that the isomorphic nature of graphs enables both the basic properties of data observed on graphs and their descriptors (features) to be preserved throughout the data analytics process, even in the case of reordering of graph vertices, where classical approaches fail. Next, to illustrate the richness and flexibility of estimation strategies performed on graph signals, spectral analysis of graphs is introduced through eigenanalysis of mathematical descriptors of graphs and in a generic way. Finally, benefiting from enhanced degrees of freedom associated with graph representations, a framework for vertex clustering and graph segmentation is established based on graph spectral representation (eigenanalysis) which demonstrates the power of graphs in various data association tasks, from image clustering and segmentation trough to low-dimensional manifold representation. The supporting examples demonstrate the promise of Graph Data Analytics in modeling structural and functional/semantic inferences. At the same time, Part I serves as a basis for Part II and Part III which deal with theory, methods and applications of processing Data on Graphs and Graph Topology Learning from data.

Keywords: graph theory; random data on graphs; big data on graphs; signal processing on graphs; machine learning on graphs; graph topology learning; systems on graphs; vertex-frequency estimation; graph neural networks; graphs and tensors.

1

Introduction

Data analytics on graphs is a multidisciplinary research area, of which the roots can be traced back to the 1970s (Afrati and Constantinides, 1978; Christofides, 1975; Morris *et al.*, 1986), one that is witnessing significant rapid growth. The recent developments, in response to the requirements posed by radically new classes of data sources, typically embark upon the classical results on "static" graph topology optimization, to treat graphs as irregular data domains, which make it possible to address completely new paradigms of "information processing on graphs" and "signal processing on graphs". This has already resulted in advanced and physically meaningful solutions in manifold applications (Grady and Polimeni, 2010; Jordan, 1998; Krim and Hamza, 2015; Marques *et al.*, 2017; Ray, 2012). For example, while the emerging areas of Graph Machine Learning (GML) and Graph Signal Processing (GSP) do comprise the classic methods of optimization of graphs themselves (Bapat, 1996; Bunse-Gerstner and Gragg, 1988; Fujiwara, 1995; Grebenkov and Nguyen, 2013; Jordan, 2004; Maheswari and Maheswari, 2016; O'Rourke *et al.*, 2016), significant progress has been made towards redefining basic data analysis objectives (spectral estimation, probabilistic inference, filtering, dimensionality reduction,

clustering, statistical learning), to make them amenable for direct estimation of signals on graphs (Chen *et al.*, 2014; Ekambaram, 2014; Gavili and Zhang, 2017; Hamon *et al.*, 2016a; Moura, 2018; Sandryhaila and Moura, 2013, 2014a,b; Shuman *et al.*, 2013; Vetterli *et al.*, 2014; Wainwright *et al.*, 2008). Indeed, this is a necessity in numerous practical scenarios where the signal domain is not designated by equidistant instants in time or a regular grid in a space or a transform domain. Examples include modern Data Analytics for e.g., social network modeling or in smart grid – data domains which are typically irregular and, in some cases, not even related to the notions of time or space, where ideally, the data sensing domain should also reflect domain-specific properties of the considered system/network; for example, in social or web related networks, the sensing points and their connectivity may be related to specific individuals, objectives, or topics, and their relations, whereby the processing on irregular domains requires the consideration of data properties other than time or space relationships. In addition, even for the data sensed in well-defined time and space domains, the new contextual and semantic-related relations between the sensing points, introduced through graphs, promise to equip problem definition with physical relevance, and consequently provide new insights into analysis and can lead to enhanced data processing results.

In applications which admit the definition of the data domain as a graph (such as social networks, power grids, vehicular networks, and brain connectivity), the role of classic temporal/spatial sampling points is assumed by graph vertices – the nodes – where the data values are observed, while the edges between vertices designate the existence and nature of vertex connections (directionality, strength). In this way, graphs are perfectly well equipped to exploit the fundamental relations among both the measured data and the underlying graph topology; this inherent ability to incorporate physically relevant data properties has made GSP and GML key technologies in the emerging field of Big Data Analytics (BDA). Indeed, in applications defined on irregular data domains, Graph Data Analytics (GDA) has been shown to offer a quantum step forward from the classical time (or space) series analyses (Brouwer and Haemers, 2012; Cvetković and Doob, 1985; Cvetković

and Gutman 2011; Cvetković *et al.*, 1980; Chung, 1997; Jones, 2013; Mejia *et al.*, 2017; Stanković *et al.*, 2017b, 2019), including the following aspects.

- Graph-based data processing approaches can be applied not only to technological, biological, and social networks, but also they can lead to both improvements of the existing and even to the creation of radically new methods in classical signal processing and machine learning (Dong *et al.*, 2012; Hamon *et al.*, 2016b; Horaud, 2009; Lu *et al.*, 2014; Masoumi and Hamza, 2017; Masoumi *et al.*, 2016; Stanković *et al.*, 2017a, 2018).

- The involvement of graphs makes it possible for the classical sensing domains of time and space (which may be represented as a linear or circular graph) to be structured in a more advanced way, e.g., by considering the connectivity of sensing points from a signal similarity or sensor association point of view.

The first step in graph data analytics is to decide on the properties of the graph as a new signal/information domain. However, while the data sensing points (graph vertices) may be well-defined by the application itself, that is not the case with their connectivity (graph edges), where:

- In the case of the various computer, social, road, transportation and electrical networks, the vertex connectivity is often naturally defined, resulting in an exact underlying graph topology.

- In many other cases, the data domain definition in a graph form becomes part of the problem definition itself, as is the case with, e.g., graphs for sensor networks, in finance or smart cities. In such cases, a vertex connectivity scheme needs to be determined based on the properties of the sensing positions or from the acquired data, as e.g., in the estimation of the temperature field in meteorology (Stanković *et al.*, 2019).

This additional aspect of the definition of an appropriate graph structure is of crucial importance for a meaningful and efficient application of the GML and GSP approaches.

With that in mind, this monograph was written in response to the urgent need of multidisciplinary data analytics communities for a seamless and rigorous transition from classical data analytics to the corresponding paradigms which operate directly on irregular graph domains. To this end, we start our approach from a review of basic definitions of graphs and their properties, followed by a physical intuition and step-by-step introduction of graph spectral analysis (eigen-analysis). Particular emphasis is on eigendecomposition of graph matrices, an area which serves as a basis for mathematical formalisms in graph signal and information processing. As an example of the ability of GML and GSP to generalize standard methodologies for graphs, we elaborate in a step-by-step way the introduction of Graph Discrete Fourier Transform (GDFT), and show that it simplifies into standard Discrete Fourier Transform (DFT) for directed circular graphs; this also exemplifies the generic nature of graph approaches. Finally, spectral vertex analysis and spectral graph segmentation are used as the basis for understanding relations among distinct but physically meaningful regions in graphs; this is demonstrated through examples of regional infrastructure modeling, brain connectivity, clustering, and dimensionality reduction.

2

Graph Definitions and Properties

Graph theory has been established for almost three centuries as a branch in mathematics, and has become a staple methodology in science and engineering areas including chemistry, operational research, electrical and civil engineering, social networks, and computer sciences. The beginning of graph theory applications in electrical engineering can be traced back to the mid-19th century with the introduction of Kirchoff's laws. Fast forward two centuries or so, the analytics of data acquired on graphs has become a rapidly developing research paradigm in Signal Processing and Machine Learning (Grady and Polimeni, 2010; Krim and Hamza, 2015; Marques *et al.*, 2017; Ray, 2012).

2.1 Basic Definitions

Definition: A graph $\mathcal{G} = \{\mathcal{V}, \mathcal{B}\}$ is defined as a set of vertices, $\mathcal{V}$, which are connected by a set of edges, $\mathcal{B} \subset \mathcal{V} \times \mathcal{V}$, where the symbol $\times$ denotes a direct product operator.

Examples of graph topologies with $N = 8$ vertices, with

$$\mathcal{V} = \{0, 1, 2, 3, 4, 5, 6, 7\}$$

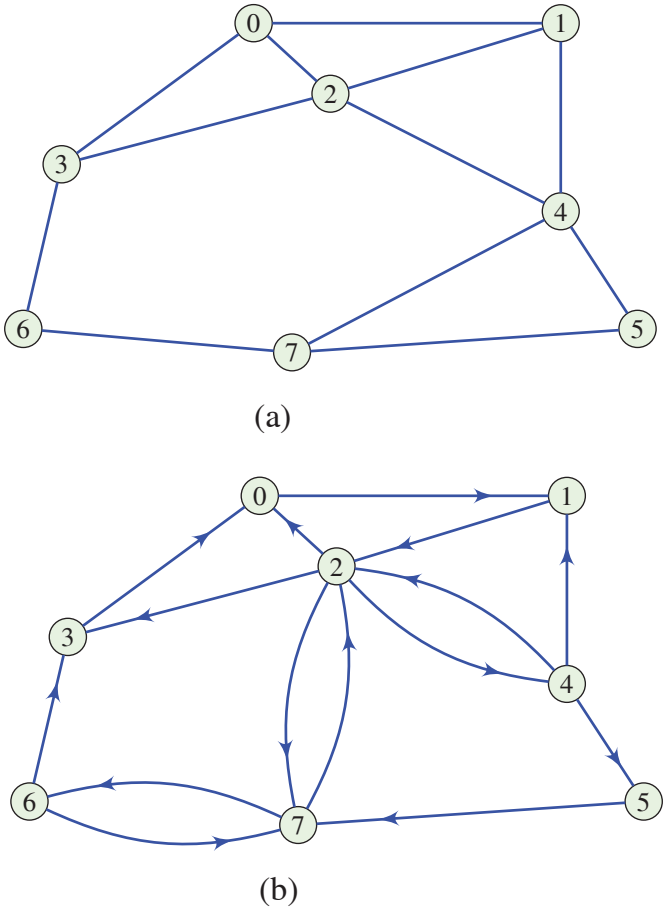


Figure 2.1: Basic graph structures. (a) Undirected graph and (b) Directed graph.

are presented in Figure 2.1, along with the corresponding edges. The vertices are usually depicted as points (circles) and the edges as lines that connect the vertices. More formally, a line between the vertices m and n indicates the existence of an edge between vertices m and n, that is, $(m, n) \in \mathcal{B}$, so that, for example, the graph from Figure 2.1(b) can be described as

$$\begin{aligned}\mathcal{V} &= \{0,1,2,3,4,5,6,7\}\\ \mathcal{B} &\subset \{0,1,2,3,4,5,6,7\} \times \{0,1,2,3,4,5,6,7\}\\ \mathcal{B} &= \{(0,1),(1,2),(2,0),(2,3),(2,4),(2,7),(3,0),\\ &\quad (4,1),(4,2),(4,5),(5,7),(6,3),(6,7),(7,2),(7,6)\}.\end{aligned}$$

Regarding the directionality of vertex connections, a graph can be undirected and directed, as illustrated respectively in Figures 2.1(a) and (b).

Definition: A graph is undirected if the edge connecting a vertex m to a vertex n also connects the vertex n to the vertex m, for all m and n.

In other words, for an undirected graph, if $(n, m) \in \mathcal{B}$ then also $(m, n) \in \mathcal{B}$, as in the case, for example, with edges $(1, 2)$ and $(2, 1)$ in Figure 2.1(a). For directed graphs, in general, this property does not hold, as shown in Figure 2.1(b). Observe, for example, that the edge $(2, 1)$ does not exist, although the edge $(1, 2)$ connects vertices 1 and 2. Therefore, undirected graphs can be considered as a special case of directed graphs.

For a given set of vertices and edges, a graph can be formally represented by its *adjacency matrix*, $\mathbf{A}$, which describes the vertex connectivity; for N vertices $\mathbf{A}$ is an $N \times N$ matrix.

Definition: The elements A_{mn} of the adjacency matrix $\mathbf{A}$ assume values $A_{mn} \in \{0, 1\}$. The value $A_{mn} = 0$ is assigned if the vertices m and n are not connected with an edge, and $A_{mn} = 1$ if these vertices are connected, that is

$$A_{mn} \overset{def}{=} \begin{cases} 1, & \text{if } (m, n) \in \mathcal{B} \\ 0, & \text{if } (m, n) \notin \mathcal{B}. \end{cases}$$

Therefore, the respective adjacency matrices, $\mathbf{A}_{\text{un}}$ and $\mathbf{A}_{\text{dir}}$, for the undirected and directed graphs from Figures 2.1(a) and (b) are given by

$$\mathbf{A}_{\text{un}} = \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} \begin{array}{c} \\ \\ \\ \\ \\ \\ \\ \\ \end{array}, \qquad (2.1)$$

$$\begin{array}{cccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array}$$

$$\mathbf{A}_{\text{dir}} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{cccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \left[\begin{array}{cccccccc} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \end{array}\right] \end{array}. \tag{2.2}$$

Adjacency matrices not only fully reflect the structure arising from the topology of data acquisition, but also they admit analysis through linear algebra, and can be sparse, or exhibit some other interesting and useful matrix properties.

Remark 1: The adjacency matrix of an undirected graph is symmetric, that is,

$$\mathbf{A} = \mathbf{A}^T.$$

Since a graph is fully determined by its adjacency matrix, defined over a given set of vertices, any change in vertex ordering will cause the corresponding changes in the adjacency matrix.

Remark 2: Observe that a vertex indexing scheme does not change the graph itself (graphs are isomorphic domains), so that the relation between adjacency matrices of the original and renumerated graphs, $\mathbf{A}_1$ and $\mathbf{A}_2$ respectively, is straightforwardly defined using an appropriate permutation matrix, $\mathbf{P}$, in the form

$$\mathbf{A}_2 = \mathbf{P}\,\mathbf{A}_1\mathbf{P}^T. \tag{2.3}$$

Recall that each row and each column of a permutation matrix has exactly one nonzero element equal to unity.

In general, in the context of an application the edges can also convey information about a relative importance about the vertices they interconnect, through a weighted graph.

Remark 3: The set of weights, $\mathcal{W}$, corresponds morphologically to the set of edges, $\mathcal{B}$, so that a weighted graph represents a generic extension of an unweighted graph. It is commonly assumed that edge

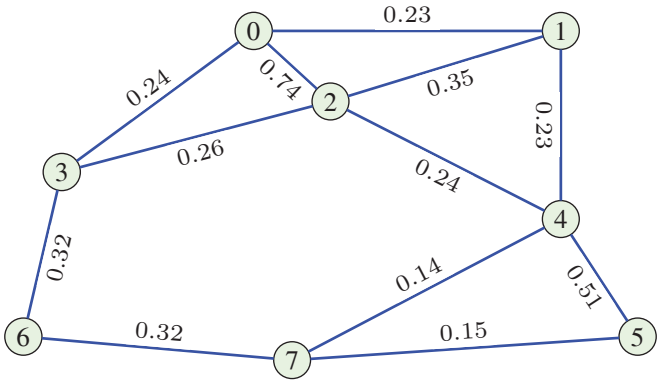


Figure 2.2: Example of a weighted graph.

weights are nonnegative real numbers; therefore, if weight 0 is associated with a nonexisting edge, then the graph can be described by a weight matrix, $\mathbf{W}$, similar to the description by the adjacency matrix $\mathbf{A}$.

Definition: A nonzero element in the weight matrix $\mathbf{W}$, $W_{mn} \in \mathcal{W}$, designates both an edge between the vertices m and n and the corresponding weight. The value $W_{mn} = 0$ indicates no edge connecting the vertices m and n. *The elements of a weight matrix are nonnegative real numbers.*

Figure 2.2 shows an example of a weighted undirected graph, with the corresponding weight matrix given by

$$\mathbf{W} = \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{bmatrix} 0 & 0.23 & 0.74 & 0.24 & 0 & 0 & 0 & 0 \\ 0.23 & 0 & 0.35 & 0 & 0.23 & 0 & 0 & 0 \\ 0.74 & 0.35 & 0 & 0.26 & 0.24 & 0 & 0 & 0 \\ 0.24 & 0 & 0.26 & 0 & 0 & 0 & 0.32 & 0 \\ 0 & 0.23 & 0.24 & 0 & 0 & 0.51 & 0 & 0.14 \\ 0 & 0 & 0 & 0 & 0.51 & 0 & 0 & 0.15 \\ 0 & 0 & 0 & 0.32 & 0 & 0 & 0 & 0.32 \\ 0 & 0 & 0 & 0 & 0.14 & 0.15 & 0.32 & 0 \end{bmatrix} \begin{array}{c} \\ \\ \\ \\ \\ \\ \\ \\ \end{array} \\ \begin{array}{cccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} . \tag{2.4}$$

In this sense, the adjacency matrix, $\mathbf{A}$, can be considered as a special case of the weight matrix, $\mathbf{W}$, whereby all nonzero weights are equal to unity. It then follows that the weight matrix of undirected graphs is also symmetric

$$\mathbf{W} = \mathbf{W}^T, \tag{2.5}$$

while, in general, for directed graphs this property does not hold.

Definition: A degree matrix, $\mathbf{D}$, of an undirected graph is a diagonal matrix with elements, D_{mm}, which are equal to the sum of weights of all edges connected to the vertex m, that is, the sum of elements in the m-th row of the weight matrix, $\mathbf{W}$,

$$D_{mm} \stackrel{def}{=} \sum_{n=0}^{N-1} W_{mn}.$$

Remark 4: For an unweighted and undirected graph, the value of the element D_{mm} is equal to the number of edges connected to the m-th vertex.

The degree matrices for directed graphs will be consider in the Appendix on the Laplacian of directed graphs.

Vertex degree centrality. The degree centrality of a vertex is defined as the number of vertices connected to the considered vertex with a single edge, and in this way it models the importance of a given vertex. For undirected and unweighted graphs, the vertex degree centrality of a vertex m is equal to the element, D_{mm}, of the degree matrix.

Example 1: For the undirected weighted graph from Figure 2.2, the degree matrix is given by

$$\mathbf{D} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{cccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \left[\begin{array}{cccccccc} 1.21 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.81 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.59 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.82 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.12 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.66 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.64 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.61 \end{array}\right] \end{array}. \tag{2.6}$$

Another important descriptor of graph connectivity is the graph Laplacian matrix, $\mathbf{L}$, which combines the weight matrix and the degree matrix.

Definition: The graph Laplacian matrix is defined as

$$\mathbf{L} \stackrel{def}{=} \mathbf{D} - \mathbf{W}, \tag{2.7}$$

where $\mathbf{W}$ is the weight matrix and $\mathbf{D}$ the diagonal degree matrix with elements $D_{mm} = \sum_n W_{mn}$. The elements of a Laplacian matrix are therefore nonnegative real numbers at the diagonal positions, and nonpositive real numbers at the off-diagonal positions.

For an undirected graph, the Laplacian matrix is symmetric, that is, $\mathbf{L} = \mathbf{L}^T$. For example, the graph Laplacian for the weighted graph from Figure 2.2 is given by

$$\mathbf{L} = \begin{bmatrix} 1.21 & -0.23 & -0.74 & -0.24 & 0 & 0 & 0 & 0 \\ -0.23 & 0.81 & -0.35 & 0 & -0.23 & 0 & 0 & 0 \\ -0.74 & -0.35 & 1.59 & -0.26 & -0.24 & 0 & 0 & 0 \\ -0.24 & 0 & -0.26 & 0.82 & 0 & 0 & -0.32 & 0 \\ 0 & -0.23 & -0.24 & 0 & 1.12 & -0.51 & 0 & -0.14 \\ 0 & 0 & 0 & 0 & -0.51 & 0.66 & 0 & -0.15 \\ 0 & 0 & 0 & -0.32 & 0 & 0 & 0.64 & -0.32 \\ 0 & 0 & 0 & 0 & -0.14 & -0.15 & -0.32 & 0.61 \end{bmatrix}. \quad (2.8)$$

For practical reasons, it is often advantageous to use the normalized Laplacian, defined as

$$\mathbf{L}_N \stackrel{def}{=} \mathbf{D}^{-1/2}(\mathbf{D} - \mathbf{W})\mathbf{D}^{-1/2} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}. \quad (2.9)$$

Remark 5: For undirected graphs, the normalized Laplacian matrix is symmetric, and has all diagonal values equal to 1, with its trace equal to the number of vertices N.

Other interesting properties, obtained through Laplacian normalization, shall be described later in the various application contexts.

One more form of the graph Laplacian is the so called **random-walk Laplacian**, defined as

$$\mathbf{L}_{RW} \stackrel{def}{=} \mathbf{D}^{-1}\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{W}. \quad (2.10)$$

The random-walk graph Laplacian is rarely used, since it has lost the symmetry property of the original graph Laplacian for undirected graphs, $\mathbf{L}_{RW} \neq \mathbf{L}_{RW}^T$.

Vertex-weighted graphs. Most of the applications of graph theory are based on edge-weighted graphs, where edge-weighting is designated by the weight matrix, $\mathbf{W}$. Note that weighting can be also introduced into graphs based on vertex-weighted approaches (although rather

rarely), whereby a weight is assigned to each vertex of a graph. To this end, we can use a diagonal matrix, $\mathbf{V}$, to define the vertex weights v_i, $i = 0, 1, \ldots, N-1$, with one possible (Chung and Langlands, 1996) version of the vertex-weighted graph Laplacian, given by

$$\mathbf{L}_V \overset{def}{=} \mathbf{V}^{1/2}\mathbf{L}\mathbf{V}^{1/2}. \tag{2.11}$$

Observe that for $\mathbf{V} = \mathbf{D}^{-1}$, the vertex-weighted graph Laplacian in (2.11) reduces to the standard edge-weighted normalized graph Laplacian in (2.9).

2.2 Some Frequently Used Graph Topologies

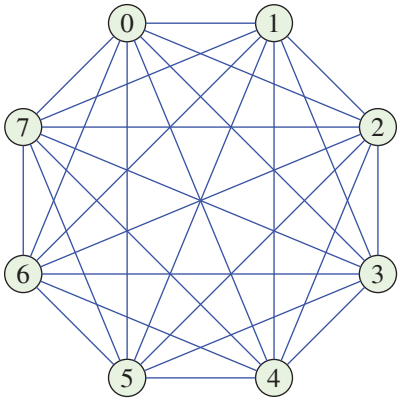
When dealing with graphs, it is useful to introduce a taxonomy of graph topologies, as follows.

1. **Complete graph.** A graph is complete if there exists an edge between every pair of its vertices. Therefore, the adjacency matrix of a complete graph has elements $A_{mn} = 1$ for all $m \neq n$, and $A_{mm} = 0$, that is, no self-connections are present. Figure 2.3(a) gives an example of a complete graph.

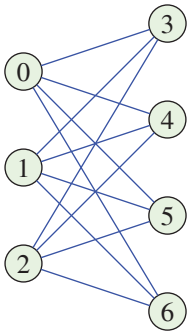
2. **Bipartite graph.** A graph for which the vertices, $\mathcal{V}$, can be partitioned into two disjoint subsets, $\mathcal{E}$ and $\mathcal{H}$, whereby $\mathcal{V} = \mathcal{E} \cup \mathcal{H}$ and $\mathcal{E} \cap \mathcal{H} = \emptyset$, such that there are no edges between the vertices within the same subset $\mathcal{E}$ or $\mathcal{H}$, is referred to as a bipartite graph. Figure 2.3(b) gives an example of a bipartite undirected graph with $\mathcal{E} = \{0, 1, 2\}$ and $\mathcal{H} = \{3, 4, 5, 6\}$, whereby all edges designate only connections between the sets $\mathcal{E}$ and $\mathcal{H}$. Observe also that the graph in Figure 2.3(b) is a complete bipartite graph, since all possible edges between the sets $\mathcal{E}$ and $\mathcal{H}$ are present.

   For convenience of mathematical formalism, if vertex ordering is performed in a such way that all vertices belonging to $\mathcal{E}$ are indexed before the vertices belonging to $\mathcal{H}$, then the resulting adjacency matrix can be written in a block form

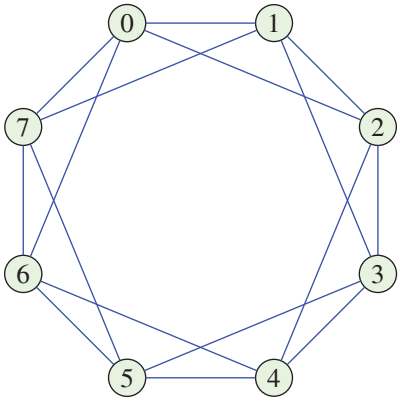
$$\mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{A}_{\mathcal{E}\mathcal{H}} \\ \mathbf{A}_{\mathcal{H}\mathcal{E}} & \mathbf{0} \end{bmatrix}, \tag{2.12}$$



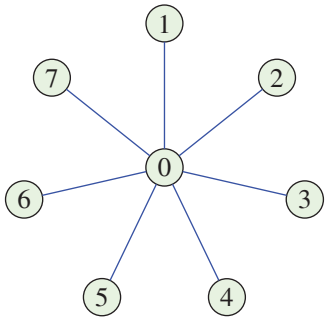
(a) Complete graph



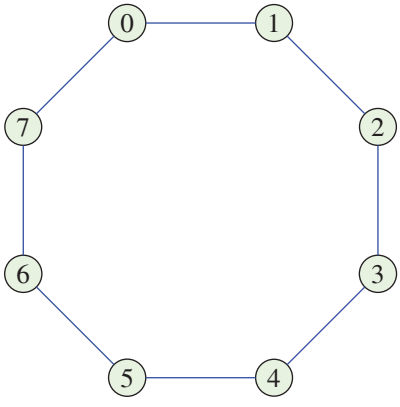
(b) Bipartite graph



(c) Regular graph

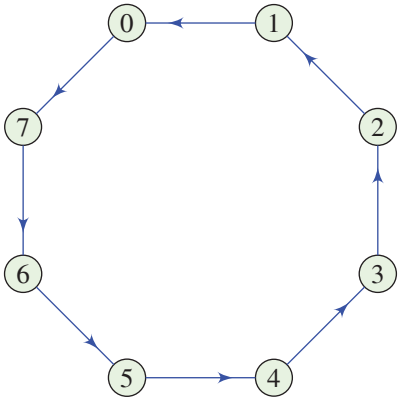


(d) Star graph



(e) Circular graph

(f) Path graph



(g) Directed circular graph

(h) Directed path graph

Figure 2.3: Special graph topologies. (a) Complete graph with 8 vertices. (b) Complete bipartite graph. (c) Regular graph whereby each vertex is connected to 4 vertices. (d) Star graph. (e) Circular graph. (f) Path graph. (g) Directed circular graph. (h) Directed path graph.

where the submatrices $\mathbf{A}_{\mathcal{E}\mathcal{H}}$ and $\mathbf{A}_{\mathcal{H}\mathcal{E}}$ define the respective connections between the vertices belonging to the disjoint sets $\mathcal{E}$ and $\mathcal{H}$. Observe that for an undirected bipartite graph, $\mathbf{A}_{\mathcal{E}\mathcal{H}} = \mathbf{A}_{\mathcal{H}\mathcal{E}}^T$. Bipartite graphs are also referred to as Kuratowski graphs, denoted by $K_{N_\mathcal{E},N_\mathcal{H}}$, where $N_\mathcal{E}$ and $N_\mathcal{H}$ are the respective numbers of vertices in the sets $\mathcal{E}$ and $\mathcal{H}$. It is important to mention that a complete bipartite graph with three vertices in each of the sets, $\mathcal{H}$ and $\mathcal{E}$, is referred to as *the first Kuratowski graph*, denoted by $K_{3,3}$, which may be used to define conditions for a graph to be planar (more detail is given in the sequel).

Multipartite graph. A generalization of the concept of bipartite graph is a multipartite (M-partite) graph for which the vertices are partitioned into M subsets, whereby each edge connects only vertices that belong to different subsets.

3. **Regular graph.** An unweighted graph is said to be regular (or $\mathcal{J}$-regular) if all its vertices exhibit the same *degree of connectivity*, $\mathcal{J}$, which is defined as the number of edges connected to each vertex. An example of a regular graph with $\mathcal{J} = 4$ is given in Figure 2.3(c). From (2.7) and (2.9), the Laplacian and the normalized Laplacian of a $\mathcal{J}$-regular graph are

$$\mathbf{L} = \mathcal{J}\,\mathbf{I} - \mathbf{A} \quad \text{and} \quad \mathbf{L}_N = \mathbf{I} - \frac{1}{\mathcal{J}}\mathbf{A}. \tag{2.13}$$

4. **Planar graph.** A graph that can be drawn on a two-dimensional plane without the crossing of any of its edges is called planar.

   For example, if the edges $(0, 2)$, $(2, 4)$, $(4, 6)$, and $(6, 0)$ in the regular graph from Figure 2.3(c) are plotted as arches outside the circle defined by the vertices, all instances of edge crossing will be avoided and such graph presentation will be planar. The graphs shown in Figures 2.3(d)–(h) are examples of planar graphs.

5. **Star graph.** This type of graph has one central vertex that is connected to all other vertices, with no other edges present. An example of star graph is given in Figure 2.3(d). Observe that a star graph can be considered as a special case of a complete

bipartite graph, with only one vertex in the first set, $\mathcal{E}$. The vertex degree centrality for the central vertex of a star graph with N vertices is therefore $N-1$.

6. **Circular (ring) graph.** A graph is said to be circular if the degree of its every vertex is $\mathcal{J}=2$. This graph is also a regular graph with $\mathcal{J}=2$. An example of a circular graph with 8 vertices is given in Figure 2.3(e).

7. **Path graph.** A series of connected vertices defines a path graph, whereby the first and the last vertex are of connectivity degree $\mathcal{J}=1$, while all other vertices are of the connectivity degree $\mathcal{J}=2$. An example of a path graph with 5 vertices is presented in Figure 2.3(f).

8. **Directed circular graph.** A directed graph is said to be circular if each vertex is related to only one predecessor vertex and only one successor vertex. An example of a directed circular graph with 8 vertices is given in Figure 2.3(g), with the adjacency matrix

$$\mathbf{A}=\begin{array}{c} \\ 0\\1\\2\\3\\4\\5\\6\\7 \end{array}\begin{array}{c} \begin{array}{cccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{bmatrix} 0&0&0&0&0&0&0&1\\ 1&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&1&0 \end{bmatrix} \end{array}. \tag{2.14}$$

Remark 6: The adjacency matrix of any directed or undirected circular graph is a circulant matrix.

9. **Directed path graph.** A directed path graph consists of a series of vertices connected in only one direction, whereby the first and the last vertex do not have a respective predecessor or successor.

An example of a directed path graph with 5 vertices is presented in Figure 2.3(h).

Remark 7: Path and circular graphs (directed and undirected) are of particular interest in Data Analytics, since their domain properties correspond to classical time or space domains. Therefore, any graph signal processing or machine learning paradigm which is developed for path and circular graphs is equivalent to its corresponding standard time and/or spatial domain paradigm.

10. **Erdös-Renyi graph model.** This is an N-vertex graph model, denoted by $\mathcal{G}(N, p)$ and introduced by Gilbert, which is formed in such a way that the presence of an edge between any two vertices m and n is designated with a probability p. Since the number of edges in a complete graph is $N(N-1)/2$, the expected number of edges in this graph model is $pN(N-1)/2$. A variant of this model, denoted by $\mathcal{G}(N, M)$, is obtained when exactly M randomly chosen edges are used in a graph with N vertices.

    These two closely related graph models are commonly used within probabilistic approaches, for example, to demonstrate that a certain property holds for almost all graphs.

11. **Stochastic block graph model.** Here, the N vertices of a graph are grouped into K communities, each comprising sets of vertices that behave similarly (we shall later refer to these groups as clusters of vertices). The vertices are then randomly connected with edges, typically with denser connections within one community, than between the different communities. The probabilities of the existence of an edge connection between the community i and the community k are denoted by p_{ik}, where $i, k = 1, 2, \ldots, K$ are the community indices; this means that commonly $p_{kk} > p_{ik}$ for $i \neq k$. If p_{ik} is constant, then this model reduces to a special case of the Erdös-Renyi model, since there is no inter-community preference on the probability for edge existence.

12. **Preferential attachment model.** In graphs that model real-world social and other networks, it is not uncommon that the

number of vertices (representing the users) increases over time. Consider a graph with N vertices, and assume that a new, $(N+1)$th vertex, is added. In the preferential attachment graph model, this new vertex, $(N+1)$, is connected with other vertices, n, with a probability proportional to their degrees, $p_n = D_{nn}/\sum_{m=1}^{N} D_{mm}$, calculated before the new vertex is added. In this way, the more connected vertices accumulate more new edges (connections).

2.3 Properties of Graphs and Associated Matrices

The notions from graph analysis that are most relevant to the processing of data on graphs are as follows.

M_1: *Symmetry:* For an undirected graph, the matrices $\mathbf{A}$, $\mathbf{W}$, and $\mathbf{L}$ are all symmetric.

M_2: *A walk* between a vertex m and a vertex n is a connected sequence of edges and vertices that begins at the vertex m and ends at the vertex n. Edges and vertices can be included in a walk more than once. There is also more than one walk between vertices m and n.

The length of a walk is equal to the number of included edges in unweighted graphs. The number of walks of length K, between a vertex m and a vertex n, is equal to the value of the mn-th element of the matrix $\mathbf{A}^K$, which can be proved through mathematical induction, as follows (Duncan, 2004).

(i) The elements, A_{mn}, of the adjacency matrix $\mathbf{A}$, by definition, indicate the existence of a walk of length $K = 1$ (an edge, in this case) between the vertices m and n in a graph.

(ii) Assume that the elements of matrix $\mathbf{A}^{K-1}$ are equal to the number of walks of length $K-1$, between two arbitrary vertices m and n.

(iii) The number of walks of length K between two vertices, m and n, is then equal to the number of all walks of length $K-1$, between the vertex m and an intermediate vertex s, $s \in \mathcal{V}$, which itself is indicated by the element at the position ms of the matrix

$\mathbf{A}^{K-1}$, according to (ii), for all s for which there is an edge from vertex s to the destination vertex n. If an edge between the intermediate vertex s and the final vertex n exists, then $A_{sn} = 1$. This means that the number of walks of length K between the vertices m and n is obtained as the inner product of the m-th row of $\mathbf{A}^{K-1}$ with the n-th column in $\mathbf{A}$, to yield the element mn of matrix $\mathbf{A}^{K-1}\mathbf{A} = \mathbf{A}^K$.

Example 2: Consider the vertex 0 and the vertex 4 in the graph from Figure 2.4, and only the walks of length $K = 2$. The adjacency matrix for this graph is given in (2.1). There are two such walks ($0 \to 1 \to 4$ and $0 \to 2 \to 4$), so that the element A^2_{04} in the first row and the fifth column of matrix $\mathbf{A}^2$, is equal to 2, as designated in bold font in the matrix $\mathbf{A}^2$ below,

$$\mathbf{A}^2 = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{cccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{bmatrix} 3 & 1 & 2 & 1 & \mathbf{2} & 0 & 1 & 0 \\ 1 & 3 & 2 & 2 & 1 & 1 & 0 & 1 \\ 2 & 2 & 4 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 1 & 3 & 1 & 0 & 0 & 1 \\ 2 & 1 & 1 & 1 & 4 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 2 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 2 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 3 \end{bmatrix} \end{array}, \tag{2.15}$$

thus indicating $K = 2$ walks between these vertices.

M_3: *The number of walks* of length not higher than K, between the vertices m and n, is given by the mn-th element of the matrix

$$\mathbf{B}_K = \mathbf{A} + \mathbf{A}^2 + \cdots + \mathbf{A}^K, \tag{2.16}$$

that is, by a value in its m-th row and n-th column. In other words, the total number of walks is equal to the sum of all walks, which are individually modeled by $\mathbf{A}^k$, $k = 1, 2, \ldots, K$, as stated in property M_2.

M_4: *The K-neighborhood* of a vertex is defined as a set of vertices that are reachable from this vertex in walks whose length is up to K.

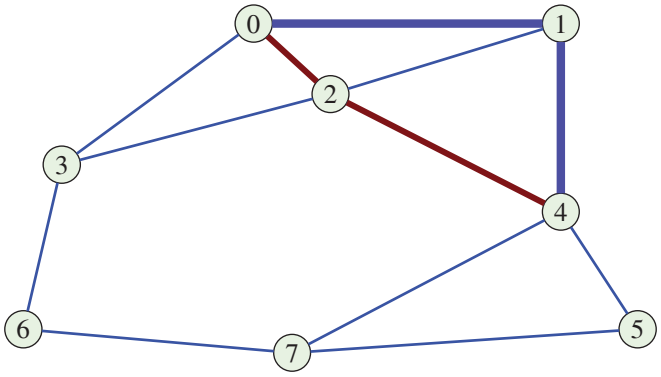


Figure 2.4: Walks of length $K = 2$ from vertex 0 to vertex 4 (thick blue and brown lines).

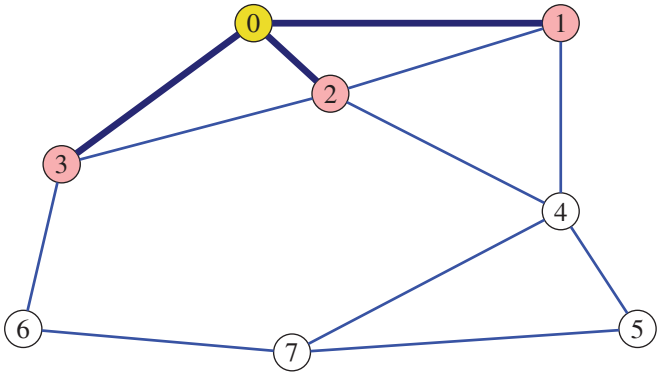
For a vertex m, based on the property M_3, the K-neighborhood is designated by the positions and the numbers of non-zero elements in the m-th row of matrix $\mathbf{B}_K$ in (2.16). The K-neighborhoods of vertex 0 for $K = 1$ and $K = 2$ are illustrated in Figure 2.5.

M_5: *A path* is a special kind of walk whereby each vertex can be included only once, whereby the number of edges included in a path is referred to as the *path cardinality* or *path length*, while the *path weight* is defined as the sum of weights along these edges.

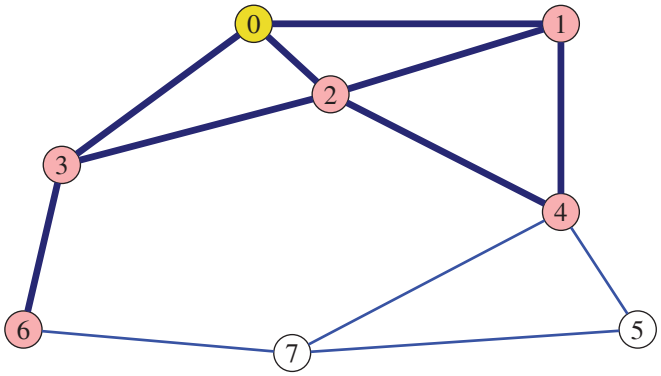
An *Euler path* is a graph path that uses every edge of a graph exactly once. An Euler path for an unweighted graph does exist if and only if at most two of its vertices are of an odd degree. An Euler path which starts and ends at the same vertex is referred to as an *Euler circuit*, and it exists if and only if the degree of every vertex is even.

A *Hamiltonian path* is a graph path between two vertices of a graph that visits each vertex in a graph exactly once, while a cycle that uses every vertex in a graph exactly once is called a *Hamiltonian cycle*.

M_6: *The distance*, r_{mn}, between two vertices m and n in an unweighed graph is equal to the minimum path length between these two vertices. For example, for the graph in Figure 2.4, the distance between vertex 1 and vertex 5 is $r_{15} = 2$.



(a)



(a)

Figure 2.5: The K-neighborhoods of vertex 0 for the graph from Figure 2.4, where: (a) $K = 1$ and (b) $K = 2$. The neighboring vertices are shaded.

M_7: *The diameter, d, of a graph* is equal to the largest distance (number of edges) between all pairs of its vertices, that is, $d = \max_{m,n \in \mathcal{V}} r_{mn}$. For example, the diameter of a complete graph is $d = 1$, while the diameter of the graph in Figure 2.4 is $d = 3$, with one of the longest paths being $6 \rightarrow 3 \rightarrow 2 \rightarrow 1$.

M_8: *Vertex closeness centrality.* The farness (remoteness) of a vertex is equal the sum of its distances to all other vertices, $f_n = \sum_{m \neq n} r_{nm}$. The vertex closeness is defined then as an inverse to the farness, $c_n = 1/f_n$, and can be interpreted as a measure of how long it will take for data to sequentially shift from the considered vertex to all other vertices. For example, the vertex farness and closeness

for the vertices $n = 2$ and $n = 5$ in Figure 2.1(a) are respectively $f_2 = 10$, $f_5 = 14$, and $c_2 = 0.1$, $c_5 = 0.071$.

M_9: *Vertex or edge betweenness.* Vertex/edge betweenness of a vertex n or edge (m, n) is equal to the number of times that this vertex/edge acts as a bridge along the shortest paths between any other two vertices.

M_{10}: *Spanning tree and minimum spanning tree.* The spanning tree of a graph is a subgraph that is tree-shaped and connects all its vertices together. A tree does not have cycles and cannot be disconnected. *The cost of the spanning tree* represents the sum of the weights of all edges in the tree. *The minimum spanning tree* is a spanning tree for which the cost is minimum among all possible spanning trees in a graph. Spanning trees are typically used in graph clustering analysis.

In the classical literature on graph theory, it is commonly assumed that the values of edge weights in weighted graphs are proportional to the standard vertex distance, r_{mn}. However, this is not the case in data analytics on graphs, where the edge weights are typically defined as a function of vertex distance, for example, through a Gaussian kernel, $W_{mn} \sim \exp(-r_{mn}^2)$, or some other data similarity metric. The cost function to minimize for the Minimum Spanning Tree (MST) can then be defined as a log-sum of distances, $r_{mn} = -2 \ln W_{mn}$. A spanning tree for the graph from Figure 2.2 is shown in Figure 2.6. The cost for this spanning tree, calculated as a sum of all distances (log-weights), r_{mn}, is 15.67.

M_{11}: *An undirected graph is called connected* if there exists a walk between each pair of its vertices.

M_{12}: If *the graph is not connected*, then it consists of two or more disjoint but locally connected subgraphs (*graph components*). Back to mathematical formalism, such disjoint graphs impose a block-diagonal form on the adjacency matrix, $\mathbf{A}$, and the Laplacian, $\mathbf{L}$. For M disjoint components (subgraphs) of a graph, these matrices

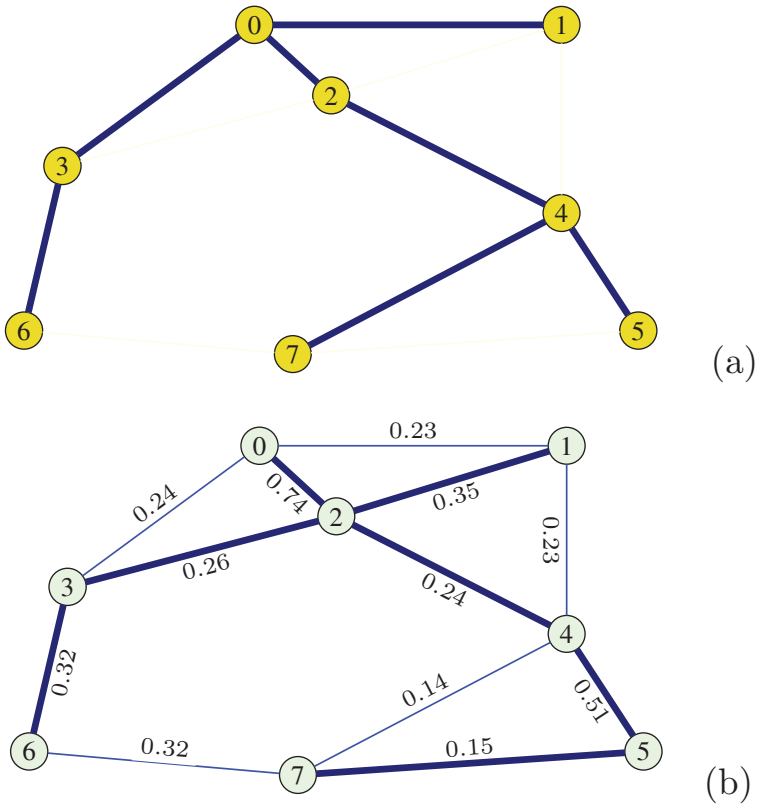


Figure 2.6: Concept of the spanning tree for graphs. (a) A spanning tree for the unweighted graph from Figure 2.1(a). (b) A spanning tree for the weighted graph from Figure 2.2, designated by thick blue edges. The graph edges in thin blue lines are not included in this spanning tree.

take the form

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{A}_M \end{bmatrix} \tag{2.17}$$

$$\mathbf{L} = \begin{bmatrix} \mathbf{L}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{L}_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{L}_M \end{bmatrix}. \tag{2.18}$$

Note that this block diagonal form is obtained only if the vertex numbering follows the subgraph structure.

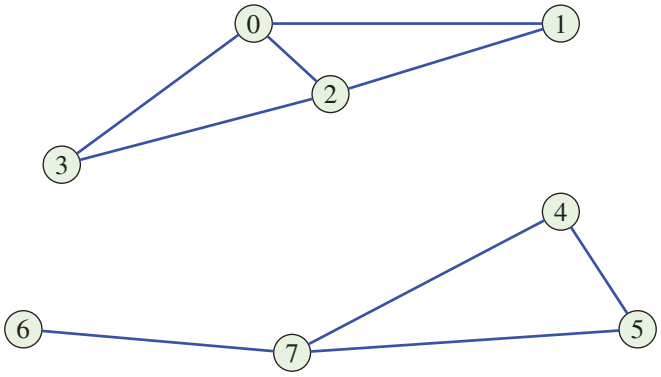


Figure 2.7: A disconnected graph which consists of two sub-graphs.

Example 3: Consider a graph derived from Figure 2.1(a) by removing some edges, as shown in Figure 2.7. The adjacency matrix for this graph is given by

$$\mathbf{A} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{cccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \left[\begin{array}{cccc:cccc} 0 & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 & 0 & 0 \\ \mathbf{1} & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 \\ \mathbf{1} & \mathbf{1} & 0 & \mathbf{1} & 0 & 0 & 0 & 0 \\ \mathbf{1} & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 \\ \hdashline 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & \mathbf{1} \\ 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & \mathbf{1} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 \end{array}\right] \end{array} \tag{2.19}$$

with the corresponding Laplacian in the form

$$\mathbf{L} = \left[\begin{array}{cccc:cccc} 3 & -1 & -1 & -1 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 3 & -1 & 0 & 0 & 0 & 0 \\ -1 & 0 & -1 & 2 & 0 & 0 & 0 & 0 \\ \hdashline 0 & 0 & 0 & 0 & 2 & -1 & 0 & -1 \\ 0 & 0 & 0 & 0 & -1 & 2 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & -1 & -1 & -1 & 3 \end{array}\right]. \tag{2.20}$$

Observe that, as elaborated above, these matrices are in a block-diagonal form with the two constituent blocks clearly separated.

Therefore, for an isolated vertex in a graph, the corresponding row and column of the matrices $\mathbf{A}$ and $\mathbf{L}$ will be zero-valued.

M_{13}: For two graphs defined on the same set of vertices, with the corresponding adjacency matrices $\mathbf{A}_1$ and $\mathbf{A}_2$, *the summation* operator produces a new graph, for which the adjacency matrix is given by

$$\mathbf{A} = \mathbf{A}_1 + \mathbf{A}_2.$$

To maintain the binary values in the resultant adjacency matrix, $A_{mn} \in \{0, 1\}$, a logical (Boolean) summation rule, e.g., $1 + 1 = 1$, may be used for matrix addition. In this monograph, the arithmetic summation rule is assumed in data analytics algorithms, as for example, in Equation (2.16) in property M_3.

M_{14}: *The Kronecker (tensor) product* of two disjoint graphs $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{B}_1)$ and $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{B}_2)$ yields a new graph $\mathcal{G} = (\mathcal{V}, \mathcal{B})$ where $\mathcal{V} = \mathcal{V}_1 \times \mathcal{V}_2$ is a direct product of the sets $\mathcal{V}_1$ and $\mathcal{V}_2$, and $((n_1, m_1), (n_2, m_2)) \in \mathcal{B}$ only if $(n_1, n_2) \in \mathcal{B}_1$ and $(m_1, m_2) \in \mathcal{B}_2$.

The adjacency matrix $\mathbf{A}$ of the resulting graph $\mathcal{G}$ is then equal to the Kronecker product of the individual adjacency matrices $\mathbf{A}_1$ and $\mathbf{A}_2$, that is

$$\mathbf{A} = \mathbf{A}_1 \otimes \mathbf{A}_2.$$

An illustration of the Kronecker product for two simple graphs is given in Figure 2.8.

M_{15}: *The Cartesian product (graph product)* of two disjoint graphs $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{B}_1)$ and $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{B}_2)$ gives a new graph $\mathcal{G} = \mathcal{G}_1 \square \mathcal{G}_2 = (\mathcal{V}, \mathcal{B})$, where $\mathcal{V} = \mathcal{V}_1 \times \mathcal{V}_2$ is a direct product of the sets $\mathcal{V}_1$ and $\mathcal{V}_2$, and $((m_1, n_1), (m_2, n_2)) \in \mathcal{B}$, only if

$$\begin{aligned} m_1 = m_2 \quad &\text{and} \quad (n_1, n_2) \in \mathcal{B}_2 \quad \text{or} \\ n_1 = n_2 \quad &\text{and} \quad (m_1, m_2) \in \mathcal{B}_1. \end{aligned}$$

The adjacency matrix of a Cartesian product of two graphs is then given by the Kronecker sum

$$\mathbf{A} = \mathbf{A}_1 \otimes \mathbf{I}_{N_2} + \mathbf{I}_{N_1} \otimes \mathbf{A}_2 \stackrel{def}{=} \mathbf{A}_1 \oplus \mathbf{A}_2,$$

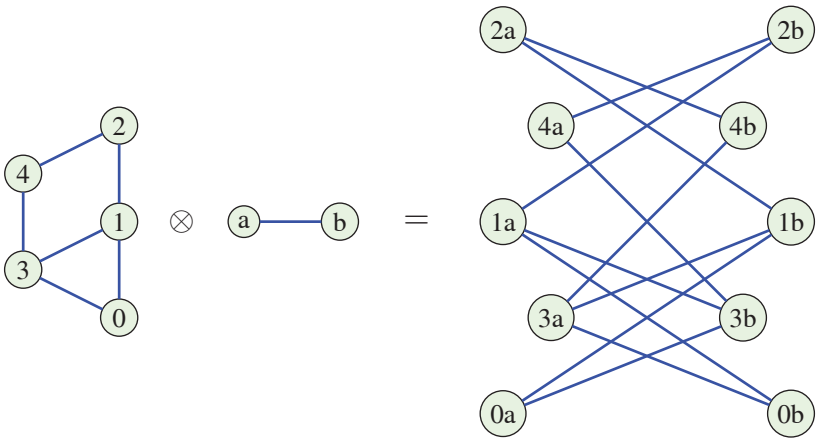


Figure 2.8: Kronecker (tensor) product of two graphs.

where $\mathbf{A}_1$ and $\mathbf{A}_2$ are the respective adjacency matrices of graphs $\mathcal{G}_1$, $\mathcal{G}_2$, while N_1 and N_2 are the corresponding numbers of vertices in $\mathcal{G}_1$ and $\mathcal{G}_2$, with $\mathbf{I}_{N_1}$ and $\mathbf{I}_{N_2}$ being the identity matrices of orders N_1 and N_2. The Cartesian product of two simple graphs is illustrated in Figure 2.9. Notice that a Cartesian product of two

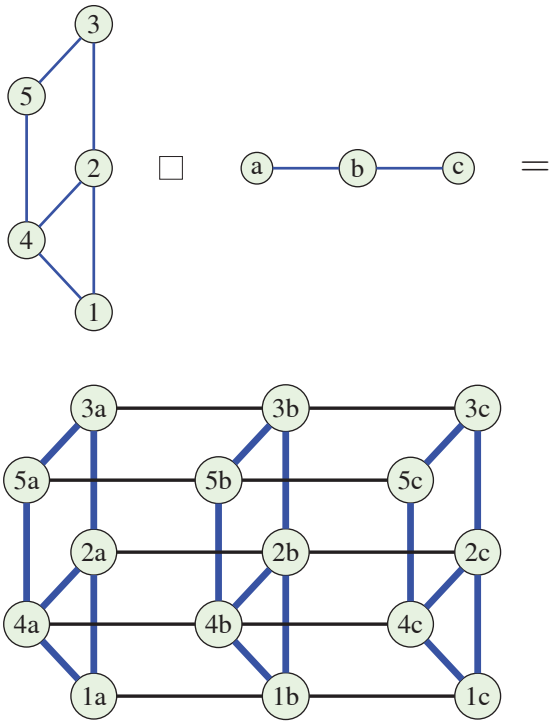


Figure 2.9: Cartesian product of two graphs.

graphs that reside in a two-dimensional space can be considered as a three-dimensional structure of vertices and edges (*cf.* tensors Saito *et al.*, 2018).

3

Spectral Decomposition of Graph Matrices

As a prerequisite for the optimization and data analytics on graphs, we next introduce several intrinsic connections between standard linear algebraic tools and graph topology (Bapat, 1996; Brouwer and Haemers, 2012; Chung, 1997; Cvetković *et al.*, 1980; Fujiwara, 1995; Jones, 2013; Maheswari and Maheswari, 2016; O'Rourke *et al.*, 2016).

3.1 Eigenvalue Decomposition of the Adjacency Matrix

Like any other general matrix, graph description matrices can be analyzed using eigenvalue decomposition. In this sense, a column vector $\mathbf{u}$ is an eigenvector of the adjacency matrix $\mathbf{A}$ if

$$\mathbf{A}\mathbf{u} = \lambda\mathbf{u}, \tag{3.1}$$

where the constant λ, that corresponds to the eigenvector $\mathbf{u}$, is called the eigenvalue.

The above relation can be equally written as $(\mathbf{A} - \lambda\mathbf{I})\mathbf{u} = \mathbf{0}$, and a nontrivial solution for $\mathbf{u}$ does exist if

$$\det|\mathbf{A} - \lambda\mathbf{I}| = 0.$$

In other words, the problem turns into that of finding zeros of $\det|\mathbf{A}-\lambda\mathbf{I}|$ as roots of a polynomial in λ, called *the characteristic polynomial* of

matrix $\mathbf{A}$, which is given by

$$P(\lambda) = \det|\mathbf{A} - \lambda\mathbf{I}| = \lambda^N + c_1\lambda^{N-1} + c_2\lambda^{N-2} + \cdots + c_N. \qquad (3.2)$$

Remark 8: The order of the characteristic polynomial of graphs has the physical meaning of the number of vertices, N, within a graph while the eigenvalues represent the roots of the characteristic polynomial, that is, $P(\lambda) = 0$.

In general, for a graph with N vertices, its adjacency matrix has N eigenvalues, $\lambda_0, \lambda_1, \ldots, \lambda_{N-1}$. Some eigenvalues may also be repeated, which indicates that zeros of algebraic multiplicity higher than one exist in the characteristic polynomial. The total number of roots of a characteristic polynomial, including their multiplicities, must be equal to its degree, N, whereby

- the *algebraic multiplicity of an eigenvalue*, λ_k, is equal to its multiplicity when considered as a root of the characteristic polynomial;
- *the geometric multiplicity of an eigenvalue*, λ_k, represents the number of linearly independent eigenvectors that can be associated with this eigenvalue.

The geometric multiplicity of an eigenvalue is always equal or lower than its algebraic multiplicity.

Denote the distinct eigenvalues in (3.2) by $\mu_1, \mu_2, \ldots, \mu_{N_m}$, and their corresponding algebraic multiplicities by $p_1, p_2, \ldots, p_{N_m}$, where $p_1 + p_2 + \cdots + p_{N_m} = N$ is equal to the order of the considered matrix/polynomial and $N_m \leq N$ is the number of distinct eigenvalues. The characteristic polynomial can now be rewritten in the form

$$P(\lambda) = (\lambda - \mu_1)^{p_1}(\lambda - \mu_2)^{p_2} \cdots (\lambda - \mu_{N_m})^{p_{N_m}}.$$

Definition: The minimal polynomial of the considered adjacency matrix, $\mathbf{A}$, is obtained from its characteristic polynomial by reducing the algebraic multiplicities of all eigenvalues to unity, and has the form

$$P_{min}(\lambda) = (\lambda - \mu_1)(\lambda - \mu_2) \cdots (\lambda - \mu_{N_m}).$$

3.1.1 Properties of the Characteristic and Minimal Polynomial

P_1: The degree of the characteristic polynomial is equal to the number of vertices in the considered graph.

P_2: For $\lambda = 0$, $P(0) = \det(\mathbf{A}) = -\lambda_0(-\lambda_1)\cdots(-\lambda_{N-1})$.

P_3: The sum of all the eigenvalues is equal to the sum of the diagonal elements of the adjacency matrix, $\mathbf{A}$, that is, its trace, $\text{tr}\{\mathbf{A}\}$. For the characteristic polynomial of the adjacency matrix, $P(\lambda)$, this means that the value of c_1 in (3.2) is $c_1 = \text{tr}\{\mathbf{A}\} = 0$.

P_4: The coefficient c_2 in $P(\lambda)$ in (3.2) is equal to the number of edges multiplied by -1.

This property, together with P_3, follows from the Faddeev–Le-Verrier algorithm to calculate the coefficients of the characteristic polynomial of a square matrix, $\mathbf{A}$, as $c_1 = -\text{tr}\{\mathbf{A}\}$, $c_2 = -\frac{1}{2}(\text{tr}\{\mathbf{A}^2\} - (\text{tr}\{\mathbf{A}\})^2)$, and so on. Since $\text{tr}\{\mathbf{A}\} = 0$ and the diagonal elements of $\mathbf{A}^2$ are equal to the number of edges connected to each vertex (vertex degree), the total number of edges is equal to $\text{tr}\{\mathbf{A}^2\}/2 = -c_2$.

P_5: The degree of the minimal polynomial, N_m, is strictly larger than the graph diameter, d.

Example 4: Consider a connected graph with N vertices and only two distinct eigenvalues, λ_0 and λ_1. The order of minimal polynomial is then $N_m = 2$, while the diameter of this graph is $d = 1$, which indicates a complete graph.

Example 5: For the graph from Figure 2.1(a), the characteristic polynomial of its adjacency matrix, $\mathbf{A}$, defined in (2.1), is given by

$$P(\lambda) = \lambda^8 - 12\lambda^6 - 8\lambda^5 + 36\lambda^4 + 36\lambda^3 - 22\lambda^2 - 32\lambda - 8,$$

with the eigenvalues

$$\lambda \in \{-2, -1.741, -1.285, -0.677, -0.411, 1.114, 1.809, 3.190\}.$$

With all the eigenvalues different, the minimal polynomial is equal to the characteristic polynomial, $P_{\min}(\lambda) = P(\lambda)$.

Example 6: The adjacency matrix for the disconnected graph from Figure 2.7 is given in (2.19), and its characteristic polynomial has the form

$$P(\lambda) = \lambda^8 - 9\lambda^6 - 6\lambda^5 + 21\lambda^4 + 26\lambda^3 + 3\lambda^2 - 4\lambda$$

with the eigenvalues

$$\lambda \in \{-1.5616, -1.4812, -1, -1, 0, 0.3111, 2.1701, 2.5616\}.$$

Observe that the eigenvalue $\lambda = -1$ is of multiplicity higher than 1 (multiplicity of 2), so that the corresponding minimal polynomial becomes

$$P_{\min}(\lambda) = \lambda^7 - \lambda^6 - 8\lambda^5 + 2\lambda^4 + 19\lambda^3 + 7\lambda^2 - 4\lambda.$$

Although this graph is disconnected, the largest eigenvalue of its adjacency matrix, $\lambda_{\max} = 2.5616$, is of multiplicity 1. Relation between the graph connectivity and the multiplicity of eigenvalues will be discussed later.

3.2 Spectral Graph Theory

If all the eigenvalues of $\mathbf{A}$ are distinct (of algebraic multiplicity 1), then the N equations in the eigenvalue problem in (3.1), that is, $\mathbf{A}\mathbf{u}_k = \lambda_k \mathbf{u}_k$, $k = 0, 1, \ldots, N-1$, can be written in a compact form as one matrix equation with respect to the adjacency matrix, as

$$\mathbf{A}\mathbf{U} = \mathbf{U}\mathbf{\Lambda}$$

or

$$\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{-1}, \tag{3.3}$$

where $\mathbf{\Lambda} = \text{diag}(\lambda_0, \lambda_1, \ldots, \lambda_{N-1})$ is the diagonal matrix with the eigenvalues on its diagonal and $\mathbf{U}$ is a matrix composed of the eigenvectors, $\mathbf{u}_k$, as its columns. Since the eigenvectors, $\mathbf{u}$, are obtained by solving a homogeneous system of equations, defined by (3.1) and in the form $(\mathbf{A} - \lambda\mathbf{I})\mathbf{u} = \mathbf{0}$, one element of the eigenvector $\mathbf{u}$ can be arbitrarily

chosen. The common choice is to enforce unit energy, $\|\mathbf{u}_k\|_2^2 = 1$, for every $k = 0, 1, \ldots, N-1$.

Remark 9: For an undirected graph, the adjacency matrix $\mathbf{A}$ is symmetric, that is $\mathbf{A} = \mathbf{A}^T$. Any symmetric matrix (i) has real-valued eigenvalues; (ii) is diagonalizable; and (iii) has orthogonal eigenvectors, and hence

$$\mathbf{U}^{-1} = \mathbf{U}^T.$$

Remark 10: For directed graphs, in general, $\mathbf{A} \neq \mathbf{A}^T$.

Recall that a square matrix is diagonalizable if all its eigenvalues are distinct (this condition is sufficient, but not necessary) or if the algebraic multiplicity of each eigenvalue is equal to its geometrical multiplicity.

For some directed graphs, the eigenvalues of their adjacency matrix may be with algebraic multiplicity higher than one, and the matrix $\mathbf{A}$ may not be diagonalizable. In such cases, the algebraic multiplicity of the considered eigenvalue is higher than its geometric multiplicity and the Jordan normal form may be used in decomposition.

Definition: The set of the eigenvalues of an adjacency matrix is called the *graph adjacency spectrum*.

Remark 11: The spectral theory of graphs studies properties of graphs through the eigenvalues and eigenvectors of their associated adjacency and graph Laplacian matrices.

Example 7: For the graph presented in Figure 2.1(a), the graph adjacency spectrum is given by $\lambda \in \{-2, -1.741, -1.285, -0.677, -0.411, 1.114, 1.809, 3.190\}$, and is shown in Figure 3.1(top).

Example 8: The vertices of the graph presented in Figure 2.1(a) are randomly reordered, as shown in Figure 3.2. Observe that the graph adjacency spectrum, given in the same figure, retains the same values, with vertex indices of the eigenvectors reordered in the same way as the graph vertices, while the eigenvalues (spectra) retain the same order as in the original graph in Figure 3.1. By a simple inspection we see that, for example, the eigenvector elements at the vertex index position $n = 0$ in Figure 3.1 are now at the vertex index position $n = 3$ in all eigenvectors in Figure 3.2.

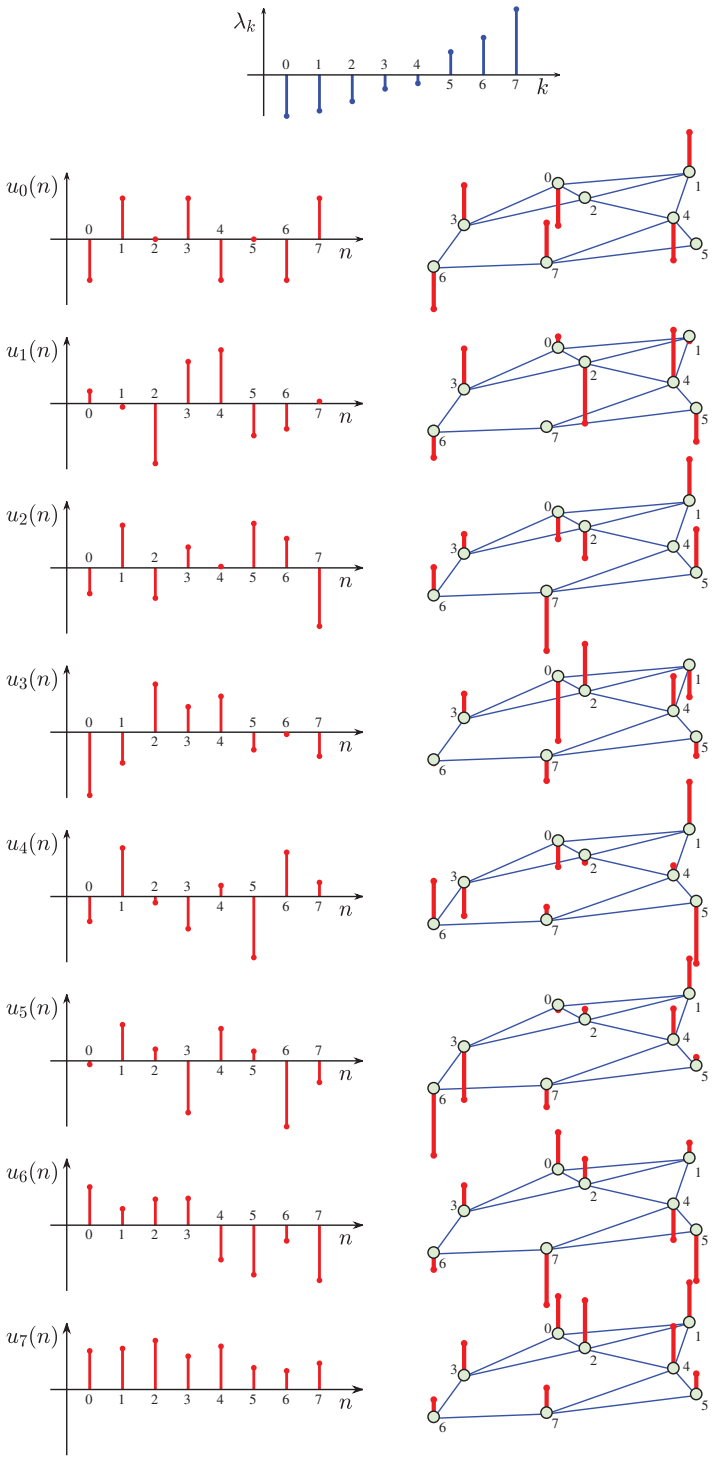


Figure 3.1: Eigenvalues, λ_k, for spectral indices (eigenvalue numbers) $k = 0, 1, \ldots, N-1$, and elements of the corresponding eigenvectors, $u_k(n)$, as a function of the vertex index $n = 0, 1, \ldots, N-1$, for the adjacency matrix, $\mathbf{A}$, of the undirected graph presented in Figure 2.1(a). The distinct eigenvectors are shown both on the vertex index axis, n, (left) and on the graph itself (right).

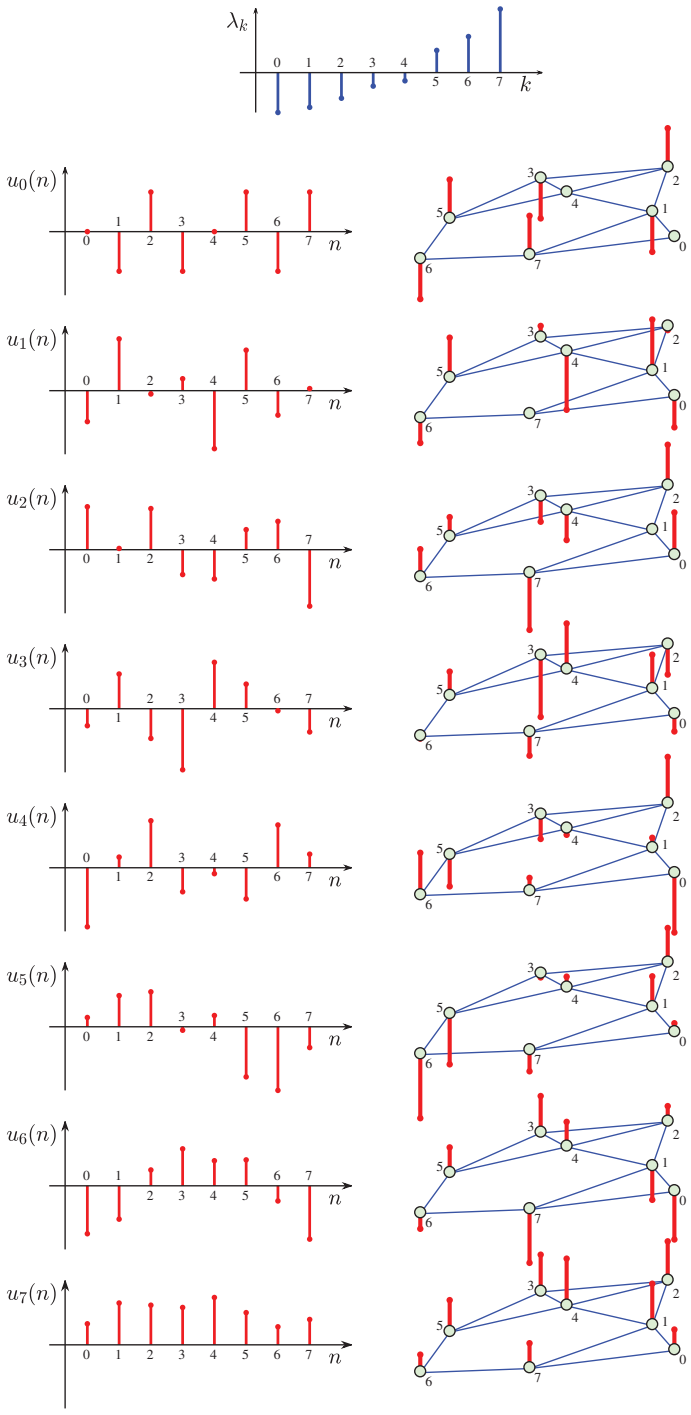


Figure 3.2: Eigenvalues, λ_k, for spectral indices (eigenvalue numbers) $k = 0, 1, \ldots, N-1$, and elements of the corresponding eigenvectors, $u_k(n)$, as a function of the vertex index $n = 0, 1, \ldots, N-1$, for the adjacency matrix, $\mathbf{A}$, of the undirected graph presented in Figure 2.1(a) with index reordering according to the scheme $[0, 1, 2, 3, 4, 5, 6, 7] \rightarrow [3, 2, 4, 5, 1, 0, 6, 7]$. The distinct eigenvectors are shown both on the vertex index axis, n, (left) and on the graph itself (right). Compare with the results for the original vertex ordering in Figure 3.1.

Remark 12: A unique feature of graphs is that vertex reindexing does not alter the eigenvalues of the adjacency matrix, while the corresponding eigenvectors of the reindexed adjacency matrix contain the same elements as the original eigenvectors, but reordered according to the vertex renumbering. This follows from the properties of the permutation matrix, as in Equation (2.3).

3.2.1 The DFT Basis Functions as a Special Case of Eigenvectors of the Adjacency Matrix

For continuity with standard spectral analysis, we shall first consider directed circular graphs, as this graph topology encodes the standard time and space domains.

Eigenvalue decomposition for the directed circular graph in Figure 2.3(g), assuming N vertices, follows from the definition $\mathbf{A}\mathbf{u}_k = \lambda_k \mathbf{u}_k$, and the form of the adjacency matrix in (2.14). Then, the elements of vector $\mathbf{A}\mathbf{u}_k$ are $u_k(n-1)$, as effectively matrix $\mathbf{A}$ here represents a shift operator, while the elements of vector $\lambda_k \mathbf{u}_k$ are $\lambda_k u_k(n)$, to give

$$u_k(n-1) = \lambda_k u_k(n), \tag{3.4}$$

where $u_k(n)$ are the elements of the eigenvector $\mathbf{u}_k$ for given vertex indices $n = 0, 1, \ldots, N-1$, and k is the index of an eigenvector, $k = 0, 1, \ldots, N-1$. This is a first-order linear difference equation, whose general form for a discrete signal $x(n)$ is $x(n) = ax(n-1)$, for which the solution is

$$u_k(n) = \frac{1}{\sqrt{N}} e^{j2\pi nk/N} \quad \text{and} \quad \lambda_k = e^{-j2\pi k/N}, \tag{3.5}$$

with $k = 0, 1, \ldots, N-1$. It is straightforward to verify that this solution satisfies the difference equation (3.4). Since the considered graph is circular, the eigenvectors also exhibit circular behavior, that is, $u_k(n) = u_k(n+N)$. For convenience, a unit energy condition is used to find the constants within the general solution of this first-order linear difference equation. Observe that the eigenvectors in (3.5) correspond exactly to the standard harmonic basis functions in DFT.

Remark 13: Classic DFT analysis may be obtained as a special case of the graph spectral analysis in (3.5), when considering directed circular

graphs. Observe that for circular graphs, the adjacency matrix plays the role of a shift operator, as seen in (3.4), with the elements of $\mathbf{A}\mathbf{u}_k$ equal to $u_k(n-1)$. This property will be used to define the shift operator on a graph in the following sections.

3.2.2 Decomposition of Graph Product Adjacency Matrices

We have already seen in Figures 2.8 and 2.9 that complex graphs, for example those with a three-dimensional vertex space, may be obtained as a Kronecker (tensor) product or a Cartesian (graph) product of two disjoint graphs $\mathcal{G}_1$ and $\mathcal{G}_2$. Their respective adjacency matrices, $\mathbf{A}_1$ and $\mathbf{A}_2$, are correspondingly combined into the adjacency matrices of the Kronecker graph product, $\mathbf{A}_{\otimes} = \mathbf{A}_1 \otimes \mathbf{A}_2$ and the Cartesian graph product, $\mathbf{A}_{\oplus} = \mathbf{A}_1 \oplus \mathbf{A}_2$, as described in properties M_{14} and M_{15}.

Graph Kronecker product. For the eigendecomposition of the Kronecker product of matrices $\mathbf{A}_1$ and $\mathbf{A}_2$, the following holds

$$\begin{aligned}\mathbf{A}_{\otimes} = \mathbf{A}_1 \otimes \mathbf{A}_2 &= (\mathbf{U}_1 \mathbf{\Lambda}_1 \mathbf{U}_1^H) \otimes (\mathbf{U}_2 \mathbf{\Lambda}_2 \mathbf{U}_2^H) \\ &= (\mathbf{U}_1 \otimes \mathbf{U}_2)(\mathbf{\Lambda}_1 \otimes \mathbf{\Lambda}_2)(\mathbf{U}_1 \otimes \mathbf{U}_2)^H,\end{aligned}$$

or in other words, the eigenvectors of the adjacency matrix of the Kronecker product of graphs are obtained by a Kronecker product of the eigenvectors of the adjacency matrices of individual graphs, as $\mathbf{u}_{k+lN_1} = \mathbf{u}_k^{(A_1)} \otimes \mathbf{u}_l^{(A_2)}$, $k = 0, 1, 2, \ldots, N_1 - 1$, $l = 0, 1, 2, \ldots, N_2 - 1$.

Remark 14: The eigenvectors of the individual graph adjacency matrices, $\mathbf{u}_k^{(A_1)}$ and $\mathbf{u}_k^{(A_2)}$, are of much lower dimensionality than those of the adjacency matrix of the resulting graph Kronecker product. This property can be used to reduce computational complexity when analyzing data observed on this kind of graph. Recall that the eigenvalues of the resulting graph adjacency matrix are equal to the product of the eigenvalues of adjacency matrices of the constituent graphs, $\mathcal{G}_2$ and $\mathcal{G}_2$, that is,

$$\lambda_{k+lN_1} = \lambda_k^{(A_1)} \lambda_l^{(A_2)}.$$

Graph Cartesian product. The eigendecomposition of the adjacency matrix of the Cartesian product of graphs, whose respective adjacency

matrices are $\mathbf{A}_1$ and $\mathbf{A}_2$, is of the form

$$\mathbf{A}_{\oplus} = \mathbf{A}_1 \oplus \mathbf{A}_2 = (\mathbf{U}_1 \otimes \mathbf{U}_2)(\mathbf{\Lambda}_1 \oplus \mathbf{\Lambda}_2)(\mathbf{U}_1 \otimes \mathbf{U}_2)^H. \tag{3.6}$$

with $\mathbf{u}_k = \mathbf{u}_k^{(A_1)} \otimes \mathbf{u}_k^{(A_2)}$ and $\lambda_{k+lN_1} = \lambda_k^{(A_1)} + \lambda_l^{(A_2)}$, $k = 0, 1, 2, \ldots, N_1 - 1$, $l = 0, 1, 2, \ldots, N_2 - 1$ (Barik *et al.*, 2015).

Remark 15: The Kronecker product and the Cartesian product of graphs share the same eigenvectors, while their spectra (eigenvalues) are different.

Example 9: The basis functions of classic two-dimensional (image) 2D-DFT follow from the spectral analysis of a Cartesian graph product which is obtained as a product the circular directed graph from Figure 2.3 with itself. Since from (3.5), the eigenvector elements of each graph are $u_k(n) = e^{j2\pi nk/N}/\sqrt{N}$, then the elements of the resulting basis functions are given by

$$u_{k+lN}(m + nN) = \frac{1}{N} e^{j2\pi mk/N} e^{j2\pi nl/N},$$

for $k = 0, 1, \ldots, N-1$, $l = 0, 1, \ldots, N-1$, $m = 0, 1, \ldots, N-1$, and $n = 0, 1, \ldots, N-1$. Figure 3.3 illustrates the Cartesian product of two circular undirected graphs with $N_1 = N_2 = 8$.

Remark 16: Cartesian products of graphs may be used for multidimensional extensions of vertex spaces and graph data domains, whereby

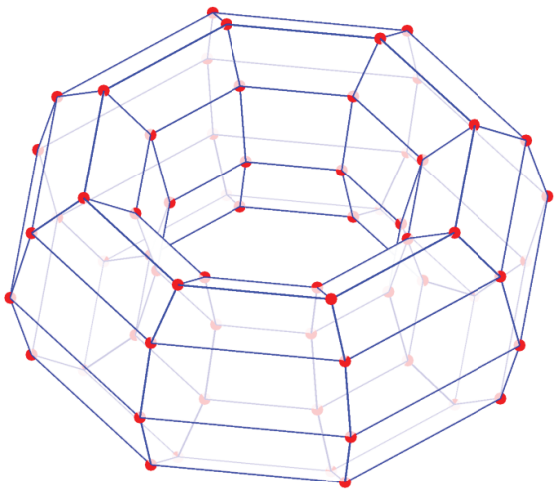


Figure 3.3: Graph Cartesian product of two planar circular unweighted graphs, with $N = 8$ vertices, produces a three-dimensional torus topology.

the resulting eigenvectors (basis functions) can be efficiently calculated using the eigenvectors of the original graphs, which are of lower dimensionality.

3.2.3 Decomposition of Matrix Powers and Polynomials

From the eigendecomposition of the adjacency matrix $\mathbf{A}$ in (3.3), eigenvalue decomposition of the squared adjacency matrix, $\mathbf{AA} = \mathbf{A}^2$, is given by

$$\mathbf{A}^2 = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{-1}\mathbf{U}\mathbf{\Lambda}\mathbf{U}^{-1} = \mathbf{U}\mathbf{\Lambda}^2\mathbf{U}^{-1},$$

under the assumption that $\mathbf{U}^{-1}$ exists. For an arbitrary natural number, m, the above result generalizes straightforwardly to

$$\mathbf{A}^m = \mathbf{U}\mathbf{\Lambda}^m\mathbf{U}^{-1}. \tag{3.7}$$

Further, for any matrix function, $f(\mathbf{A})$, that can be written in a polynomial form, given by

$$f(\mathbf{A}) = h_0\mathbf{A}^0 + h_1\mathbf{A}^1 + h_2\mathbf{A}^2 + \cdots + h_{N-1}\mathbf{A}^{N-1},$$

its eigenvalue decomposition is, in general, given by

$$f(\mathbf{A}) = \mathbf{U}f(\mathbf{\Lambda})\mathbf{U}^{-1}.$$

This is self-evident from the properties of eigendecomposition of matrix powers, defined in (3.7), and the linearity of the matrix multiplication operator, $\mathbf{U}(h_0\mathbf{A}^0 + h_1\mathbf{A}^1 + h_2\mathbf{A}^2 + \cdots + h_{N-1}\mathbf{A}^{N-1})\mathbf{U}^{-1}$.

3.3 Eigenvalue Decomposition of the Graph Laplacian

Spectral analysis for graphs can also be performed based on the graph Laplacian, $\mathbf{L}$, defined in (2.7). For convenience, we here adopt the same notation for the eigenvalues and eigenvectors of the graph Laplacian, as we did for the adjacency matrix $\mathbf{A}$, although the respective eigenvalues and eigenvectors are not directly related. The Laplacian of an undirected graph can be therefore written as

$$\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T \quad \text{or} \quad \mathbf{LU} = \mathbf{U}\mathbf{\Lambda},$$

where $\mathbf{\Lambda} = \text{diag}(\lambda_0, \lambda_1, \ldots, \lambda_{N-1})$ is a diagonal matrix of Laplacian eigenvalues and $\mathbf{U}$ is the orthonormal matrix of its eigenvectors (in columns), with $\mathbf{U}^{-1} = \mathbf{U}^T$. Note that the Laplacian of an undirected graph is always diagonalizable, since its matrix $\mathbf{L}$ is real and symmetric.

Then, every eigenvector, $\mathbf{u}_k$, $k = 0, 1, \ldots, N-1$, of a graph Laplacian, $\mathbf{L}$, satisfies

$$\mathbf{L}\mathbf{u}_k = \lambda_k \mathbf{u}_k. \tag{3.8}$$

Definition: The set of the eigenvalues, λ_k, $k = 0, 1, \ldots, N-1$, of the graph Laplacian is referred to as *the graph spectrum* or *graph Laplacian spectrum* (*cf.* graph adjacency spectrum based on $\mathbf{A}$).

Example 10: The Laplacian spectrum of the undirected graph from Figure 2.2, is given by

$$\lambda \in \{0, 0.29, 0.34, 0.79, 1.03, 1.31, 1.49, 2.21\},$$

and shown in Figure 3.4, along with the corresponding eigenvectors. The Laplacian spectrum of the disconnected graph from Figure 3.5, is given by

$$\lambda \in \{0, 0, 0.22, 0.53, 0.86, 1.07, 1.16, 2.03\},$$

and is illustrated in Figure 3.6. The disconnected nature of this graph is indicated by the zero-valued eigenvalue of algebraic multiplicity 2, that is, $\lambda_0 = \lambda_1 = 0$.

Remark 17: Observe that when graph-component (sub-graph) based vertex indexing is employed, then even though the respective graph spectra for the connected graph in Figure 3.4 and the disconnected graph Figure 3.6 are similar, for a given spectral index, the eigenvectors of a disconnected graph take nonzero values on only one of the individual disconnected graph components.

3.3.1 Properties of Laplacian Eigenvalue Decomposition

L_1: The graph Laplacian matrix is defined in (2.7) in such a way that the sum of elements in its each row (column) is zero. As a consequence, this enforces the inner products of every row of $\mathbf{L}$ with any constant vector, $\mathbf{u}$, to be zero-valued, that is,

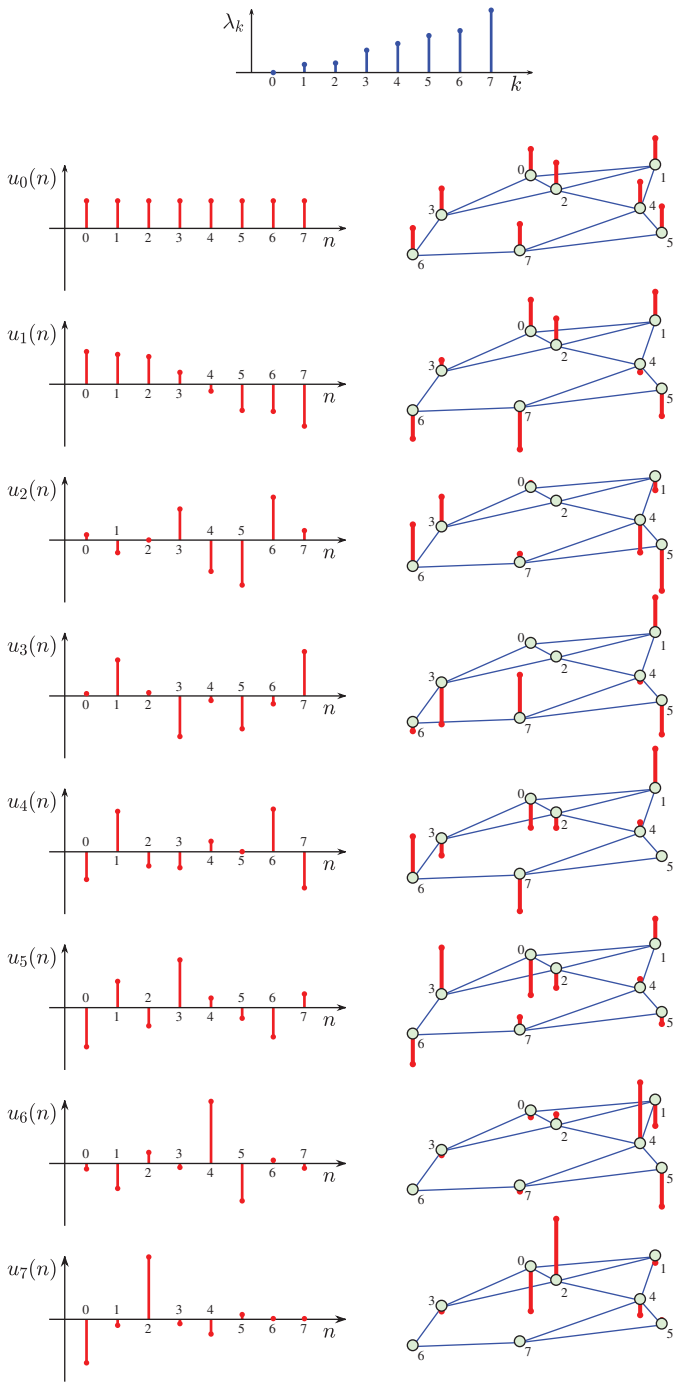


Figure 3.4: Eigenvalues, λ_k, for spectral indices (eigenvalue number) $k = 0, 1, \ldots, N-1$, and elements of the corresponding eigenvectors, $u_k(n)$, as a function of the vertex index $n = 0, 1, \ldots, N-1$, for the Laplacian matrix, $\mathbf{L}$, of the undirected graph presented in Figure 2.2. The distinct eigenvectors are shown both on the vertex index axis, n, (left) and on the graph itself (right). A comparison with the eigenvectors of the adjacency matrix in Figure 3.1, shows that for the adjacency matrix the smoothest eigenvector corresponds to the largest eigenvalue, while for the graph Laplacian the smoothest eigenvector corresponds to the smallest eigenvalue, λ_0.

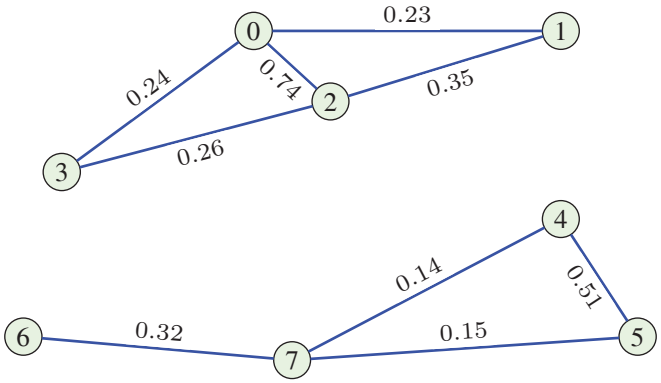


Figure 3.5: A disconnected weighted graph which consists of two sub-graphs.

$\mathbf{Lu} = \mathbf{0} = 0 \cdot \mathbf{u}$, for any constant vector $\mathbf{u}$. This means that at least one eigenvalue of the Laplacian is zero, $\lambda_0 = 0$, and its corresponding constant unit energy eigenvector is given by $\mathbf{u}_0 = [1, 1, \ldots, 1]^T/\sqrt{N} = \mathbf{1}/\sqrt{N}$.

L_2: The multiplicity of the eigenvalue $\lambda_0 = 0$ of the graph Laplacian is equal to the number of connected components (connected subgraphs) in the corresponding graph.

This property follows from the fact that the Laplacian matrix of disconnected graphs can be written in a block diagonal form, as in (2.18). *The set of eigenvectors of a block-diagonal matrix is obtained by grouping together the sets of eigenvectors of individual block submatrices.* Since each subgraph of a disconnected graph behaves as an independent graph, then for each subgraph $\lambda_0 = 0$ is the eigenvalue of the corresponding block Laplacian submatrix, according to property L_1. Therefore, the multiplicity of the eigenvalue $\lambda_0 = 0$ corresponds to the number of disjoint components (subgraphs) within a graph.

This property does not hold for the adjacency matrix, since *there are no common eigenvalues* in the adjacency matrices for the blocks (subgraphs) or arbitrary graphs, like in the case of $\lambda_0 = 0$ for the graph Laplacian matrix and any graph. *In this sense, the graph Laplacian matrix carries more physical meaning than the corresponding adjacency matrix.*

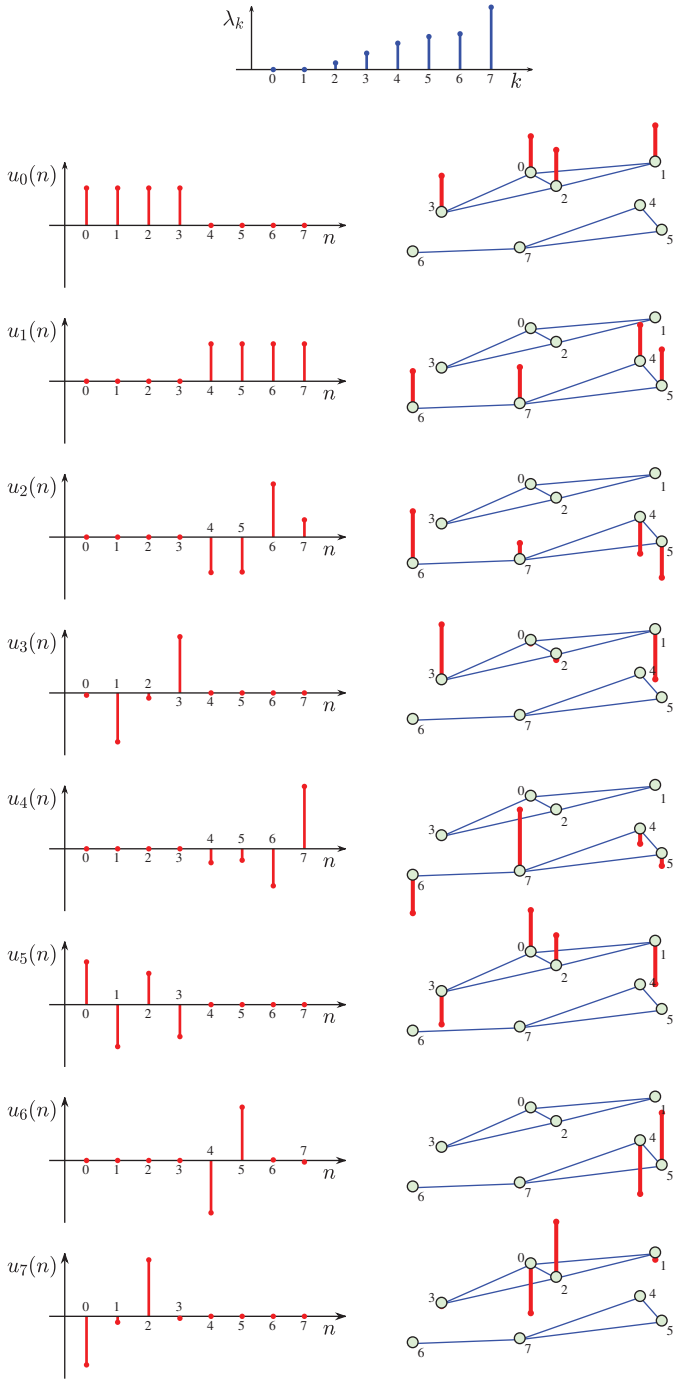


Figure 3.6: Eigenvalues, λ_k, for spectral indices (eigenvalue number) $k = 0, 1, \ldots, N-1$, and elements of the corresponding eigenvectors, $u_k(n)$, as a function of the vertex index $n = 0, 1, \ldots, N-1$, for the Laplacian matrix, $\mathbf{L}$, of the undirected graph presented in Figure 3.5. The distinct eigenvectors are shown both on the vertex index axis, n, (left) and on the graph itself (right). This graph is characterized with the zero eigenvalue of algebraic multiplicity 2, that is, $\lambda_0 = \lambda_1 = 0$. Observe that for every spectral index, k, the corresponding eigenvectors take nonzero values on only one of the disconnected graph components.

Remark 18: If $\lambda_0 = \lambda_1 = 0$, then the graph is not connected. If $\lambda_2 > 0$, then there are exactly two individually connected but globally disconnected components in this graph. If $\lambda_1 \neq 0$ then this eigenvalue may be used to describe the so called *algebraic connectivity of a graph*, whereby very small values of λ_1 indicate that the graph is weakly connected. This can be used as an indicator of the possibility of graph segmentation, as elaborated in Section 4.2.3.

L_3: As with any other matrix, the sum of the eigenvalues of the Laplacian matrix is equal to its trace. For the normalized Laplacian, the sum of its eigenvalues is equal to the number of vertices, N, if there are no isolated vertices.

L_4: The coefficient, c_N, in the characteristic polynomial of the graph Laplacian matrix

$$P(\lambda) = \det|\mathbf{L} - \lambda\mathbf{I}| = \lambda^N + c_1\lambda^{N-1} + \cdots + c_{N-1}\lambda + c_N$$

is equal to 0, since $\lambda = 0$ is an eigenvalue for the Laplacian matrix.

For unweighted graphs, the coefficient c_1 is equal to the number of edges multiplied by -2. This is straightforward to show following the relations from property P_4 which state that $c_1 = -\text{tr}\{\mathbf{L}\}$. For unweighted graphs, the diagonal elements of the Laplacian are equal to the corresponding vertex degrees (number of edges). Therefore, the number of edges in an unweighted graph is equal to $-c_1/2$.

Example 11: The characteristic polynomial of the Laplacian for the graph from Figure 2.1(a) is given by

$$P(\lambda) = \lambda^8 - 24\lambda^7 + 238\lambda^6 - 1256\lambda^5 + 3777\lambda^4 - 6400\lambda^3 + 5584\lambda^2 - 1920\lambda$$

with the eigenvalues $\lambda \in \{0, 1, 1.4384, 3, 4, 4, 5, 5.5616\}$. Observe that the eigenvalue $\lambda = 4$ is of multiplicity higher than one. The minimal polynomial therefore becomes $P_{\min}(\lambda) = \lambda^7 - 20\lambda^6 + 158\lambda^5 - 624\lambda^4 + 1281\lambda^3 - 1276\lambda^2 + 480\lambda$.

For the disconnected graph in Figure 2.7, the characteristic polynomial of the Laplacian is given by

$$P(\lambda) = \lambda^8 - 18\lambda^7 + 131\lambda^6 - 490\lambda^5 + 984\lambda^4 - 992\lambda^3 + 384\lambda^2,$$

with the eigenvalues $\lambda \in \{0, 0, 1, 2, 3, 4, 4, 4\}$. The eigenvalue $\lambda = 0$ is of algebraic multiplicity 2 and the eigenvalue $\lambda = 4$ of algebraic multiplicity 3, so that the minimal polynomial takes the form

$$P_{min}(\lambda) = \lambda^5 - 10\lambda^4 + 35\lambda^3 - 50\lambda^2 + 24\lambda.$$

Since the eigenvalue $\lambda = 0$ is of algebraic multiplicity 2, property L_2 indicates that this graph is disconnected, with two disjoint sub-graphs as its constituent components.

L_5: Graphs with identical spectra are called *isospectral* or *cospectral* graphs. However, *isospectral graphs are not necessary isomorphic, and construction of isospectral graphs that are not isomorphic* is an important topic in graph theory.

Remark 19: A complete graph is *uniquely determined by its Laplacian spectrum* (Van Dam and Haemers, 2003). The Laplacian spectrum of a complete unweighted graph, with N vertices, is $\lambda_k \in \{0, N, N, \ldots, N\}$. Therefore, two complete isospectral graphs are also isomorphic.

L_6: For a $\mathcal{J}$-regular graph, as in Figure 2.3(c), the eigenvectors of the graph Laplacian and the adjacency matrices are identical, with the following relation for the eigenvalues,

$$\lambda_k^{(L)} = \mathcal{J} - \lambda_k^{(A)},$$

where the superscript L designates the Laplacian and superscript A the corresponding adjacency matrix. This follows directly from $\mathbf{U}^T\mathbf{L}\mathbf{U} = \mathbf{U}^T(\mathcal{J}\mathbf{I} - \mathbf{A})\mathbf{U}$.

L_7: The eigenvalues of the normalized graph Laplacian, $\mathbf{L}_N = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$, are nonnegative and upper-bounded by

$$0 \leq \lambda \leq 2.$$

The equality for the upper bound holds if and only if the graph is a bipartite graph, as in Figure 2.3(b). This will be proven within the next property.

L_8: The eigenvalues and eigenvectors of the normalized Laplacian of a bipartite graph, with the disjoint sets of vertices $\mathcal{E}$ and $\mathcal{H}$, satisfy the relation, referred to as *the graph spectrum folding*, given by

$$\lambda_k = 2 - \lambda_{N-k} \tag{3.9}$$

$$\mathbf{u}_k = \begin{bmatrix} \mathbf{u}_{\mathcal{E}} \\ \mathbf{u}_{\mathcal{H}} \end{bmatrix} \quad \text{and} \quad \mathbf{u}_{N-k} = \begin{bmatrix} \mathbf{u}_{\mathcal{E}} \\ -\mathbf{u}_{\mathcal{H}} \end{bmatrix}, \tag{3.10}$$

where $\mathbf{u}_k$ designates the k-th eigenvector of a bipartite graph, $\mathbf{u}_{\mathcal{E}}$ is its part indexed on the first set of vertices, $\mathcal{E}$, while $\mathbf{u}_{\mathcal{H}}$ is the part of the eigenvector $\mathbf{u}_k$ indexed on the second set of vertices, $\mathcal{H}$.

In order to prove this property, we shall write the adjacency and the normalized Laplacian matrices of an undirected bipartite graph in their block forms

$$\mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{A}_{\mathcal{E}\mathcal{H}} \\ \mathbf{A}_{\mathcal{E}\mathcal{H}}^T & \mathbf{0} \end{bmatrix} \quad \text{and} \quad \mathbf{L}_N = \begin{bmatrix} \mathbf{I} & \mathbf{L}_{\mathcal{E}\mathcal{H}} \\ \mathbf{L}_{\mathcal{E}\mathcal{H}}^T & \mathbf{I} \end{bmatrix}.$$

The eigenvalue relation, $\mathbf{L}_N\mathbf{u}_k = \lambda_k\mathbf{u}_k$, can now be evaluated as

$$\mathbf{L}_N\mathbf{u}_k = \begin{bmatrix} \mathbf{u}_{\mathcal{E}} + \mathbf{L}_{\mathcal{E}\mathcal{H}}\mathbf{u}_{\mathcal{H}} \\ \mathbf{L}_{\mathcal{E}\mathcal{H}}^T\mathbf{u}_{\mathcal{E}} + \mathbf{u}_{\mathcal{H}} \end{bmatrix} = \lambda_k \begin{bmatrix} \mathbf{u}_{\mathcal{E}} \\ \mathbf{u}_{\mathcal{H}} \end{bmatrix}.$$

From there, we have $\mathbf{u}_{\mathcal{E}} + \mathbf{L}_{\mathcal{E}\mathcal{H}}\mathbf{u}_{\mathcal{H}} = \lambda_k\mathbf{u}_{\mathcal{E}}$ and $\mathbf{L}_{\mathcal{E}\mathcal{H}}^T\mathbf{u}_{\mathcal{E}} + \mathbf{u}_{\mathcal{H}} = \lambda_k\mathbf{u}_{\mathcal{H}}$, resulting in $\mathbf{L}_{\mathcal{E}\mathcal{H}}\mathbf{u}_{\mathcal{H}} = (\lambda_k - 1)\mathbf{u}_{\mathcal{E}}$ and $\mathbf{L}_{\mathcal{E}\mathcal{H}}^T\mathbf{u}_{\mathcal{E}} = (\lambda_k - 1)\mathbf{u}_{\mathcal{H}}$, to finally yield

$$\mathbf{L}_N \begin{bmatrix} \mathbf{u}_{\mathcal{E}} \\ -\mathbf{u}_{\mathcal{H}} \end{bmatrix} = (2 - \lambda_k) \begin{bmatrix} \mathbf{u}_{\mathcal{E}} \\ -\mathbf{u}_{\mathcal{H}} \end{bmatrix}.$$

This completes the proof.

Since for the graph Laplacian $\lambda_0 = 0$ always holds (see the property L_1), from $\lambda_k = 2 - \lambda_{N-k}$ in (3.9), it then follows that the largest eigenvalue is $\lambda_N = 2$, which also proves the property L_7 for a bipartite graph.

3.3.2 Fourier Analysis as a Special Case of the Laplacian Spectrum

Consider the undirected circular graph from Figure 2.3(e). Then, from the property L_1, the eigendecomposition relation for the Laplacian of this graph, $\mathbf{Lu} = \lambda\mathbf{u}$, admits a simple form

$$-u(n-1) + 2u(n) - u(n+1) = \lambda u(n). \tag{3.11}$$

This is straightforward to show by inspecting the Laplacian for the undirected circular graph from Figure 2.3(e), with $N = 8$ vertices for which the eigenvalue analysis is based on

$$\mathbf{Lu} = \begin{bmatrix} 2 & -1 & 0 & 0 & 0 & 0 & 0 & -1 \\ -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 \\ -1 & 0 & 0 & 0 & 0 & 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} u(0) \\ u(1) \\ u(2) \\ u(3) \\ u(4) \\ u(5) \\ u(6) \\ u(7) \end{bmatrix}. \tag{3.12}$$

This directly gives the term $-u(n-1)+2u(n)-u(n+1)$, while a simple inspection of the values $u(0)$ and $u(N)$ illustrates the circular nature of the eigenvectors; see also Remark 6. The solution to the second order difference equation in (3.11) is $u_k(n) = \cos(\frac{2\pi kn}{N} + \phi_k)$, with $\lambda_k = 2(1 - \cos(\frac{2\pi k}{N}))$. Obviously, for every eigenvalue, λ_k (except for λ_0 and for the last eigenvalue, λ_{N-1}, for an even N), we can choose to have two orthogonal eigenvectors with, for example, $\phi_k = 0$ and $\phi_k = \pi/2$. This means that most of the eigenvalues are of algebraic multiplicity 2, i.e., $\lambda_1 = \lambda_2$, $\lambda_3 = \lambda_4$, and so on. This eigenvalue multiplicity of two can be formally expressed as

$$\lambda_k = \begin{cases} 2 - 2\cos(\pi(k+1)/N), & \text{for odd } k = 1, 3, 5, \ldots, \\ 2 - 2\cos(\pi k/N), & \text{for even } k = 2, 4, 6, \ldots. \end{cases} \tag{3.13}$$

For an odd N, $\lambda_{N-2} = \lambda_{N-1}$, whereas for an even N we have $\lambda_{N-1} = 2$ which is of algebraic multiplicity 1.

The corresponding eigenvectors $\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_{N-1}$, then have the form

$$u_k(n) = \begin{cases} \sin(\pi(k+1)n/N), & \text{for odd } k,\ k < N-1 \\ \cos(\pi kn/N), & \text{for even } k \\ \cos(\pi n), & \text{for odd } k,\ k = N-1, \end{cases} \tag{3.14}$$

where $k = 0, 1, \ldots, N-1$ and $n = 0, 1, \ldots, N-1$.

Relation between graph Fourier analysis based on the graph adjacency matrix and graph Laplacian matrix. Recall that an arbitrary linear combination of eigenvectors $\mathbf{u}_{2k-1}$ and $\mathbf{u}_{2k}$, $1 \leq k < N/2$, is also an eigenvector since the corresponding eigenvalues are equal (in this case both their algebraic and geometric multiplicities are equal to 2, see Equation (3.13)). With this in mind, we can rewrite the full set of eigenvectors in an alternative compact form, given by

$$u_k(n) = \begin{cases} 1, & \text{for } k = 0 \\ e^{j\pi(k+1)n/N} = e^{j2\pi rn/N}, & \text{for odd } k = 2r-1,\ k < N-1 \\ e^{-j2\pi kn/N} = e^{-j2\pi rn/N}, & \text{for even } k = 2r,\ k > 0 \\ \cos(\pi n), & \text{for odd } k,\ k = N-1, \end{cases}$$

where $j^2 = -1$. The above eigenvectors are assumed to be normalized. It is now clear that, as desired, this set of eigenvectors is also orthonormal, so that the individual eigenvectors, $\mathbf{u}_k$, correspond to the harmonic basis functions within the standard temporal/spatial DFT obtained by the directed circular graph adjacency matrix decomposition.

If the vertices correspond to the pixels of a two-dimensional $N \times N$ image in a stacked-column representation, then the edge weights for a given vertex, n, are $w_{mn} = 1$, $m \in \{n-N, n-1, n+1, n+N\}$, while the degree of every vertex, n, is equal to 4. The corresponding graph Laplacian now becomes a discrete approximation of second-order partial derivatives, and is used as a standard tool in image processing for edge detection, while two-dimensional Fourier analysis can be defined using eigenvalue decomposition of this Laplacian. Notice that the Laplacians in the graph Cartesian product exhibit similar relations to those for the adjacency matrix in Equation (3.6) and Figure 3.3.

4

Vertex Clustering and Mapping

An important task for data analytics on graphs is to identify groups of vertices which exhibit similar behavior, referred to as *vertex clustering*. This is of particular importance in machine learning for data on irregular domains, while vertex clustering also represents a basis for collaborative data processing. Spectral domain analysis for vertex clustering may be performed based on several measures appropriate to the task at hand including the graph Laplacian, normalized graph Laplacian, generalized Laplacian eigenvectors, principal component analysis of the graph Laplacian, commute time (effective resistance) spectral vectors, the diffusion spectral vectors or other factors.

Definition: Vertex clustering is a type of graph learning which aims to group together vertices from the set $\mathcal{V}$ into multiple disjoint subsets, $\mathcal{V}_i$, called clusters. Vertices which are clustered into a subset of vertices, $\mathcal{V}_i$, are expected to exhibit a larger degree of within-cluster mutual similarity (in some sense) than with the vertices in other subsets, $\mathcal{V}_j$, $j \neq i$.

While the clustering of graph vertices refers to the process of identifying and arranging the vertices of a graph into nonoverlapping vertex subsets, with data in each subset expected to exhibit relative similarity

in some sense, the *segmentation* of a graph refers to its partitioning into nonoverlapping graph segments (components).

The notion of *vertex similarity metrics* and their use to accordingly cluster the vertices into sets, $\mathcal{V}_i$, of "related" vertices in graphs, has been a focus of significant research effort in machine learning and pattern recognition; this has resulted in a number of established vertex similarity measures and corresponding methods for graph clustering (Schaeffer, 2007). These can be considered within two main categories (i) *clustering based on graph topology* and (ii) *spectral (eigenvector-based) methods for graph clustering*.

Notice that in traditional clustering, a vertex is assigned to one cluster only. The type of clustering where a vertex may belong to more than one cluster is referred to as *fuzzy clustering* (Mordeson and Nair, 2012; Schaeffer, 2007), an approach that is not yet widely accepted in the context of graphs.

4.1 Clustering Based on Graph Topology

Among many such existing methods, the most popular ones are based on:

- Finding the minimum set of edges whose removal would disconnect a graph in some "optimal" or "least disturbance" way (*minimum cut* based clustering).
- Designing clusters within a graph based on the disconnection of vertices or edges which belong to the highest numbers of shortest paths in the graph (*vertex betweenness* and *edge betweenness* based clustering).
- The minimum spanning tree of a graph has been a basis for a number of widely used clustering methods (Kleinberg and Tardos, 2006; Morris *et al.*, 1986).
- Analysis of highly connected subgraphs (HCS) (Khuller, 1998) has also been used for graph clustering.
- Finally, *graph data analysis* may be used for **machine learned graph clustering**, like for example, the k-means based clustering methods (Dhillon *et al.*, 2004; Jain, 2010).

4.1.1 Minimum Graph Cut

We shall first briefly review the notion of graph cuts, as spectral methods for graph clustering may be introduced and interpreted based on the analysis and approximation of the (graph topology-based) minimum cut clustering.

Definition: Consider an undirected graph which is defined by a set of vertices, $\mathcal{V}$, and the corresponding set of edge weights, $\mathcal{W}$. Assume next that the vertices are grouped into $k = 2$ disjoint subsets of vertices, $\mathcal{E} \subset \mathcal{V}$ and $\mathcal{H} \subset \mathcal{V}$, with $\mathcal{E} \cup \mathcal{H} = \mathcal{V}$ and $\mathcal{E} \cap \mathcal{H} = \emptyset$. A cut of this graph, for the given subsets of vertices, $\mathcal{E}$ and $\mathcal{H}$, is equal to a sum of all weights that correspond to the edges which connect the vertices between the subsets, $\mathcal{E}$ and $\mathcal{H}$, that is

$$Cut(\mathcal{E}, \mathcal{H}) = \sum_{\substack{m \in \mathcal{E} \\ n \in \mathcal{H}}} W_{mn}.$$

Remark 20: For clarity, we shall focus on the case with $k = 2$ disjoint subsets of vertices. However, the analysis can be straightforwardly generalized to $k \geq 2$ disjoint subsets of vertices and the corresponding minimum k-cuts.

Example 12: Consider the graph in Figure 2.2, and the sets of vertices $\mathcal{E} = \{0, 1, 2, 3\}$ and $\mathcal{H} = \{4, 5, 6, 7\}$, shown in Figure 4.1. Its cut into the two components (sub-graphs), $\mathcal{E}$ and $\mathcal{H}$, involves the weights of all edges which exist between these two sets, that is, $Cut(\mathcal{E}, \mathcal{H}) = 0.32 + 0.24 + 0.23 = 0.79$. Such edges are shown by thin red lines in Figure 4.1.

Definition: A cut which exhibits the minimum value of the sum of weights between the disjoint subsets $\mathcal{E}$ and $\mathcal{H}$, considering all possible divisions of the set of vertices, $\mathcal{V}$, is referred to as *the minimum cut*. Finding the minimum cut of a graph in this way is a combinatorial problem.

Remark 21: The number of all possible combinations to split an even number of vertices, N, into two disjoint subsets is given by

$$C = \binom{N}{1} + \binom{N}{2} + \cdots + \binom{N}{N/2 - 1} + \binom{N}{N/2} \Big/ 2.$$

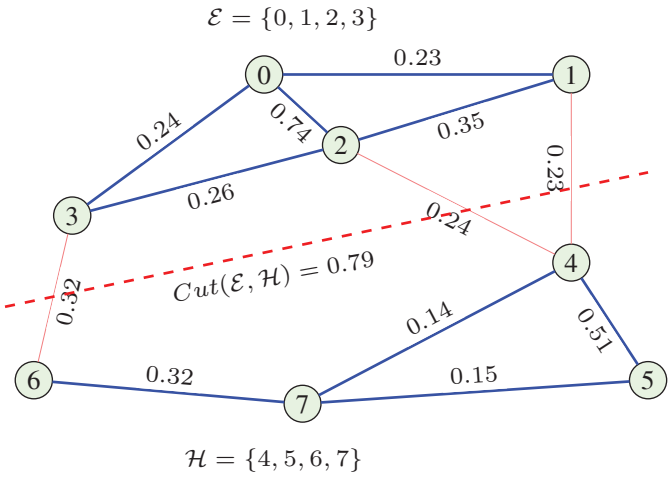


Figure 4.1: A cut for the weighted graph from Figure 2.2, with the disjoint subsets of vertices defined by $\mathcal{E} = \{0, 1, 2, 3\}$ and $\mathcal{H} = \{4, 5, 6, 7\}$. The edges between the sets $\mathcal{E}$ and $\mathcal{H}$ are designated by thin red lines. The cut, $Cut(\mathcal{E}, \mathcal{H})$, is equal to the sum of the weights that connect sets $\mathcal{E}$ and $\mathcal{H}$, and has the value $Cut(\mathcal{E}, \mathcal{H}) = 0.32 + 0.24 + 0.23 = 0.79$.

To depict the computational burden associated with this "brute force" graph cut approach, even for a relatively small graph with $N = 50$ vertices, the number of combinations to split the vertices into two subsets is $C = 5.6 \cdot 10^{14}$.

Example 13: The minimum cut for the graph from Figure 4.1 is

$$Cut(\mathcal{E}, \mathcal{H}) = 0.32 + 0.14 + 0.15 = 0.61$$

for $\mathcal{E} = \{0, 1, 2, 3, 4, 5\}$ and $\mathcal{H} = \{6, 7\}$. This can be confirmed by considering all $\binom{8}{1} + \binom{8}{2} + \binom{8}{3} + \binom{8}{4}/2 = 127$ possible cuts, that is, all combinations of the subsets $\mathcal{E}$ and $\mathcal{H}$ for this small size graph or by using, for example, the Stoer-Wagner algorithm (Stoer and Wagner, 1997).

4.1.2 Maximum-Flow Minimum-Cut Approach

This approach to the minimum cut problem employs the framework of *flow networks*.

Definition: A flow network is **a directed graph** with an arbitrary number of vertices, $N \geq 3$, but which involves **two given vertices** (nodes) called the source vertex, s, and the sink vertex, t, whereby the *capacity*

of edges (arcs) is defined by their weights. The flow (of information, water, traffic, ...) through an edge cannot exceed its capacity (the value of edge weight). For any vertex in the graph *the sum of all input flows is equal to the sum of all its output flows* (except for the source and sink vertices).

Problem formulation. The maximum-flow minimum-cut solution to the graph partitioning aims to find the maximum value of *flow* that can be passed through the graph (network flow) from the source vertex, s, to the sink vertex, t. The solution is based on the *max-flow min-cut theorem* which states that the maximum flow through a graph from a given source vertex, s, to a given sink vertex, t, is equal to the minimum cut, that is, the minimum sum of those edge weights (capacities) which, if removed, would disconnect the source, s from the sink, t (minimum cut capacity) (Kleinberg and Tardos, 2006; Kron, 1963). Physical interpretation of this theorem is obvious, since the maximum flow is naturally defined by the graph flow bottleneck between the source and sink vertices. The capacity of the bottleneck (maximum possible flow) will then be equal to the minimum capacity (weight values) of the edges which, if removed, would disconnect the graph into two parts, one containing vertex s and the other containing vertex t. Therefore, the problem of maximum flow is equivalent to the minimum cut (capacity) problem, under the assumption that the considered vertices, s and t, must belong to different disjoint subsets of vertices $\mathcal{E}$ and $\mathcal{H}$. *This kind of cut, with predefined vertices s and t, is called the (s,t) cut.*

Remark 22: In general, if the source and sink vertices are not given, the maximum flow algorithm should be repeated for all combinations of the source and sink vertices in order to find the minimum cut of a graph.

The most widely used approach to solve the minimum-cut maximum-flow problem is the *Ford–Fulkerson method* (Kleinberg and Tardos, 2006; Kron, 1963).

Example 14: Consider the weighted graph from Figure 2.2, with the assumed source and sink vertices, $s = 0$ and $t = 6$, as shown in Figure 4.2(a). The Ford–Fulkerson method is based on the analysis of paths and the corresponding flows between the source and sink vertex.

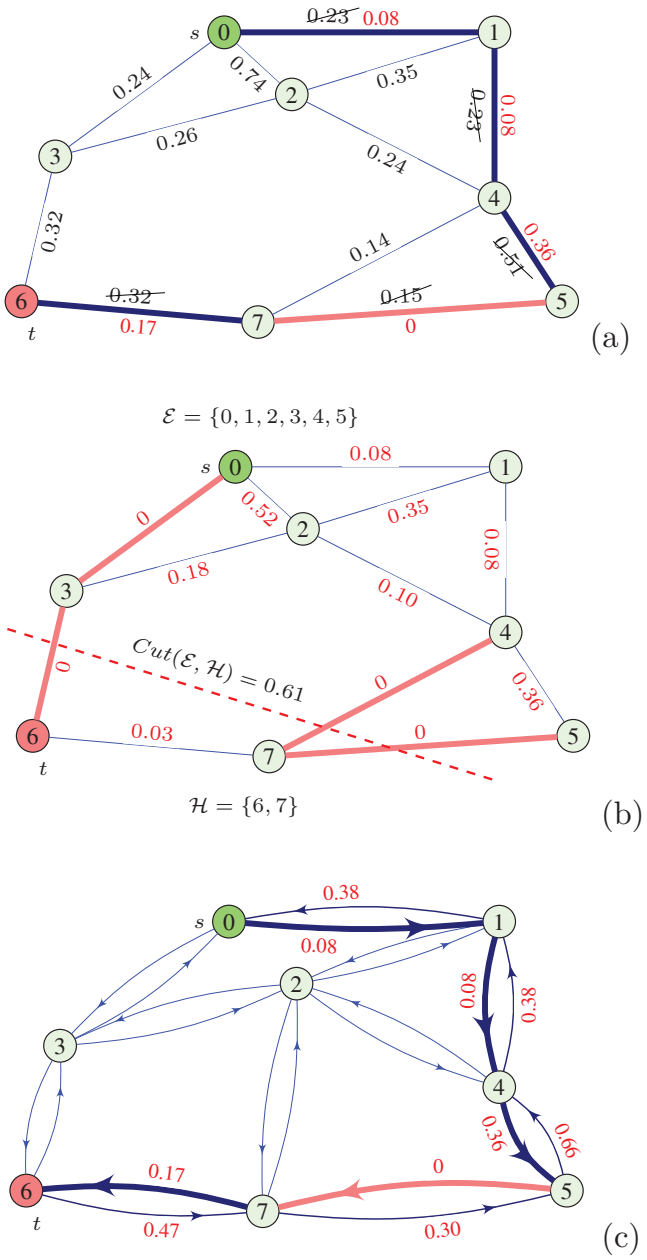


Figure 4.2: Principle of the maximum flow minimum cut method. (a) The weighted graph from Figure 2.2, with the assumed source vertex $s = 0$ and sink vertex $t = 6$, and a path between these two vertices for which the maximum flow is equal to the minimum capacity (weight) along this path, $W_{57} = 0.15$. This maximum flow value, $W_{57} = 0.15$, is then subtracted from all the original edge capacities (weights) to yield the new *residual edge capacities (weights)* which are shown in red. (b) The final edge capacities (weights) after the maximum flows are subtracted for all paths $0 \to 3 \to 6$, $0 \to 2 \to 4 \to 7 \to 6$, and $0 \to 2 \to 3 \to 6$, between vertices $s = 0$ and $t = 6$, with the resulting minimum cut now crossing only the zero-capacity (zero-weight) edges with its value equal to the sum of their initial capacities (weights), shown in Panel (a) in black. (c) A directed form of the undirected graph from (a), with the same path and the residual capacities (weights) given for both directions.

One such possible path between s and t, $0 \to 1 \to 4 \to 5 \to 7 \to 6$, is designated by the thick line in Figure 4.2(a). Recall that the *maximum flow*, for a path connecting the vertices $s = 0$ and $t = 6$, is restricted by the minimum capacity (equal to the minimum weight) along the considered path. For the considered path $0 \to 1 \to 4 \to 5 \to 7 \to 6$ the maximum flow from $s = 0$ to $t = 6$ is therefore equal to

$$\underset{0\to1\to4\to5\to7\to6}{\text{max-flow}} = \min\{0.23, 0.23, 0.51, 0.15, 0.32\} = 0.15,$$

since the minimum weight along this path is that connecting vertices 5 and 7, $W_{57} = 0.15$. The value of this maximum flow is then subtracted from each capacity (weight) in the considered path, with the new *residual edge capacities (weights)* designated in red in the *residual graph* in Figure 4.2(a). The same procedure is repeated for the remaining possible paths $0 \to 3 \to 6$, $0 \to 2 \to 4 \to 7 \to 6$, and $0 \to 2 \to 3 \to 6$, with appropriate corrections to the capacities (edge weights) after consideration of each path. The final residual form of the graph, after zero-capacity edges are obtained in such a way that no new path with nonzero flow from s to t can be defined, is given in Figure 4.2(b). For example, if we consider the path $0 \to 1 \to 2 \to 3 \to 6$ (or any other path), in the residual graph, then its maximum flow would be 0, since the residual weight in the edge $3 \to 6$ is equal to 0. The minimum cut has now been obtained as that which separates the sink vertex, $t = 6$, and its neighborhood from the the source vertex, $s = 0$, through the remaining zero-capacity (zero-weight) edges. This cut is shown in Figure 4.2(b), and separates the vertices $\mathcal{H} = \{6, 7\}$ from the rest of vertices by cutting the edges connecting vertices $3 \to 6$, $4 \to 7$, and $5 \to 7$. The original total weights of these edges are $Cut(\mathcal{E}, \mathcal{H}) = 0.32 + 0.14 + 0.15 = 0.61$.

We have so far considered an undirected graph, but since the Ford–Fulkerson algorithm is typically applied to directed graphs, notice that an undirected graph can be considered as a directed graph with every edge being split into a pair of edges having the same weight (capacity), but with opposite directions. After an edge is used in one direction (for example, edge 5–7 in Figure 4.2(a)) with a flow equal to its maximum capacity of 0.15 in the considered direction, the other flow direction (sister edge) becomes 0.30, as shown in Figure 4.2(c). The edge with opposite direction could be used (up the algebraic sum

of flows in both directions being equal to the total edge capacity) to form another path (if possible) from the source to the sink vertex. More specifically, the capacity of an edge (from the pair) in the assumed direction is reduced by the same value of the considered flow, while the capacity of the opposite-direction edge (from the same pair) is increased by the same flow, and can be used to send the flow in reverse direction if needed. All residual capacities for the path from Figure 4.2(a) are given in Figure 4.2(c). For clarity, the edge weights which had not been changed by this flow are not shown in Figure 4.2(c).

Ratio Minimum Cut

A number of optimization approaches may be employed to enforce some desired properties on graph clusters. One such approach is *the ratio minimum cut*, which is commonly used in graph theory, and is introduced by penalizing the value of $Cut(\mathcal{E}, \mathcal{H})$ by an additional term (cost) to enforce the subsets $\mathcal{E}$ and $\mathcal{H}$ to be *simultaneously as large as possible*. An obvious form of the ratio cut is given by Hagen and Kahng (1992)

$$CutN(\mathcal{E}, \mathcal{H}) = \left(\frac{1}{N_{\mathcal{E}}} + \frac{1}{N_{\mathcal{H}}}\right) \sum_{\substack{m \in \mathcal{E} \\ n \in \mathcal{H}}} W_{mn}, \tag{4.1}$$

where $N_{\mathcal{E}}$ and $N_{\mathcal{H}}$ are the respective numbers of vertices in the sets $\mathcal{E}$ and $\mathcal{H}$. Since $N_{\mathcal{E}} + N_{\mathcal{H}} = N$, the term $\frac{1}{N_{\mathcal{E}}} + \frac{1}{N_{\mathcal{H}}}$ reaches its minimum for $N_{\mathcal{E}} = N_{\mathcal{H}} = N/2$.

Example 15: Consider again Example 12, and the graph from Figure 4.1. For the sets of vertices, $\mathcal{E} = \{0, 1, 2, 3\}$ and $\mathcal{H} = \{4, 5, 6, 7\}$, the ratio cut is calculated as $CutN(\mathcal{E}, \mathcal{H}) = (1/4 + 1/4)0.79 = 0.395$. This cut also represents the minimum ratio cut for this graph; this can be confirmed by checking all possible cut combinations of $\mathcal{E}$ and $\mathcal{H}$ in this (small) graph. Figure 4.3 illustrates the clustering of vertices according to the minimum ratio cut. Notice, however, that in general the minimum cut and the minimum ratio cut do not produce the same vertex clustering into $\mathcal{E}$ and $\mathcal{H}$.

Graph separability. Relevant to this section, the minimum cut value admits a physical interpretation as a *measure of graph separability*.

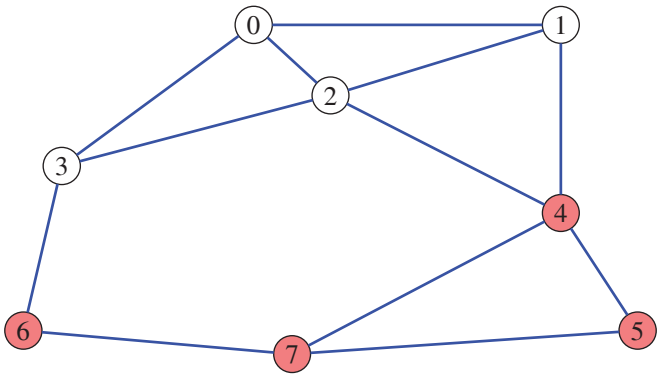


Figure 4.3: A clustering scheme based on the minimum ratio cut of the vertices in the graph from Figure 2.2 into two vertex clusters, $\mathcal{E} = \{0, 1, 2, 3\}$ and $\mathcal{H} = \{4, 5, 6, 7\}$. This cut corresponds to the arbitrarily chosen cut presented in Figure 4.1.

An ideal separability is possible if the minimum cut is equal to zero, meaning that there are no edges between subsets $\mathcal{E}$ and $\mathcal{H}$. In Example 15, the minimum cut value was $CutN(\mathcal{E}, \mathcal{H}) = 0.395$, which is not close to 0, and indicates that the segmentation of this graph into two subgraphs would not yield a close approximation of the original graph.

4.1.3 Volume Normalized Minimum Cut

A more general form of the normalized cut may also involve vertex weights when designing the size of subsets $\mathcal{E}$ and $\mathcal{H}$. By defining, respectively, *the volumes* of these sets as $V_{\mathcal{E}} = \sum_{n \in \mathcal{E}} D_{nn}$ and $V_{\mathcal{H}} = \sum_{n \in \mathcal{H}} D_{nn}$, and using these volumes instead of the numbers of vertices $N_{\mathcal{E}}$ and $N_{\mathcal{H}}$ in the definition of the ratio cut in (4.1), we arrive at Shi and Malik (2000)

$$CutV(\mathcal{E}, \mathcal{H}) = \left(\frac{1}{V_{\mathcal{E}}} + \frac{1}{V_{\mathcal{H}}} \right) \sum_{\substack{m \in \mathcal{E} \\ n \in \mathcal{H}}} W_{mn}, \tag{4.2}$$

where $D_{nn} = \sum_{m \in \mathcal{V}} W_{mn}$ is the degree of a vertex n. *The vertices with a higher degree, D_{nn}, are considered as structurally more important than the vertices with lower degrees.*

The above discussion shows that finding the normalized minimum cut is also a combinatorial problem, for which an approximative spectral-based solution will be discussed later in this section.

4.1.4 Other Forms of the Normalized Cut

In addition to the two presented forms of the normalized cut, based on the number of vertices and volume, other frequently used forms are:

1. The *sparsity of a cut* which is defined as

$$\rho(\mathcal{E}) = \frac{1}{N_{\mathcal{E}} N_{\mathcal{V}-\mathcal{E}}} \sum_{\substack{m\in\mathcal{E} \\ n\in\mathcal{V}-\mathcal{E}}} W_{mn}, \tag{4.3}$$

   where $\mathcal{V}-\mathcal{E}$ is the *set difference* of $\mathcal{V}$ and $\mathcal{E}$. The sparsity of a cut, $\rho(\mathcal{E})$, is related to the ratio cut as $N\rho(\mathcal{E}) = CutN(\mathcal{E},\mathcal{H})$, since $\mathcal{H} = \mathcal{V}-\mathcal{E}$ and $N_{\mathcal{E}} + N_{\mathcal{V}-\mathcal{E}} = N$. *The sparsity of a graph is then defined as the minimum sparsity of a cut.* It then follows that the cut which exhibits minimum sparsity and the minimum ratio cut in (4.1) produce the same set $\mathcal{E}$.

2. The *edge expansion of a subset,* $\mathcal{E} \subset \mathcal{V}$, is defined by

$$\alpha(\mathcal{E}) = \frac{1}{N_{\mathcal{E}}} \sum_{\substack{m\in\mathcal{E} \\ n\in\mathcal{V}-\mathcal{E}}} W_{mn}, \tag{4.4}$$

   with $N_{\mathcal{E}} \leq N/2$. Observe a close relation of edge expansion to the ratio cut in (4.1).

3. The *Cheeger ratio of a subset,* $\mathcal{E} \subset \mathcal{V}$, is defined as

$$\phi(\mathcal{E}) = \frac{1}{\min\{V_{\mathcal{E}}, V_{\mathcal{V}-\mathcal{E}}\}} \sum_{\substack{m\in\mathcal{E} \\ n\in\mathcal{V}-\mathcal{E}}} W_{mn}. \tag{4.5}$$

   The minimum value of $\phi(\mathcal{E})$ is denoted by $\phi(\mathcal{V})$ and called *the Cheeger constant* or *conductance* of a graph (Mohar, 1989). This form is closely related to the volume normalized cut in (4.2).

4.2 Spectral Methods for Graph Clustering

This class of methods is a modern alternative to the classical direct graph topology analysis, whereby vertex clustering is based on the eigenvectors of the graph Laplacian. Practical spectral methods for

graph clustering typically employ several smoothest eigenvectors of the graph Laplacian.

Simplified algorithms for vertex clustering may even employ only one eigenvector, namely the second (Fiedler, 1973) eigenvector of the graph Laplacian, $\mathbf{u}_1$, to yield a *quasi-optimal clustering* or partitioning scheme on a graph. These are proven to be efficient in a range of applications, including data processing on graphs, machine learning, and computer vision (Malik *et al.*, 2001). Despite their simplicity, such algorithms are typically quite accurate, and a number of studies show that *graph clustering and cuts based on the second eigenvector,* $\mathbf{u}_1$*, give a good approximation to the optimal cut* (Ng *et al.*, 2002; Spielman and Teng, 2007; Von Luxburg, 2007). Using more than one smooth eigenvector in graph clustering and partitioning will increase the number of degrees of freedom to consequently yield more physically meaningful clustering, when required for practical applications in data analytics.

For an enhanced insight we shall next review the smoothness index, before introducing the notions of graph spectral vectors and their distance, followed by the notions of similarity and clustering of vertices.

4.2.1 Smoothness of Eigenvectors on Graphs

Definition: The smoothness of an eigenvector, $\mathbf{u}_k$, is introduced through its quadratic Laplacian form, $\mathbf{u}_k^T\mathbf{L}\mathbf{u}_k$, with the smoothness index equal to the corresponding eigenvalue, λ_k, that is

$$\mathbf{u}_k^T(\mathbf{L}\mathbf{u}_k) = \mathbf{u}_k^T(\lambda_k\mathbf{u}_k) = \lambda_k. \tag{4.6}$$

To demonstrate physical intuition behind the use of quadratic form, $\mathbf{u}_k^T\mathbf{L}\mathbf{u}_k$, as a smoothness metric of $\mathbf{u}_k$, consider

$$\mathbf{u}_k^T\mathbf{L}\mathbf{u}_k = \mathbf{u}_k^T(\mathbf{D} - \mathbf{W})\mathbf{u}_k.$$

Then, an n-th element of the vector $\mathbf{L}\mathbf{u}_k$ is given by

$$\sum_{m=0}^{N-1} W_{nm}u_k(n) - \sum_{m=0}^{N-1} W_{nm}u_k(m),$$

since $D_{nn} = \sum_{m=0}^{N-1} W_{nm}$. Therefore,

$$\begin{aligned}\mathbf{u}_k^T \mathbf{L} \mathbf{u}_k &= \sum_{m=0}^{N-1} u_k(m) \sum_{n=0}^{N-1} W_{mn}(u_k(m) - u_k(n)) \\ &= \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} W_{mn}(u_k^2(m) - u_k(m)u_k(n)). \end{aligned} \tag{4.7}$$

Owing to the symmetry of the weight matrix, $\mathbf{W}$ (as shown in (2.5)), we can use $W_{nm} = W_{mn}$ to replace the full summation of $u_k^2(n)$ over m and n with a half of the summations for both $u_k^2(m)$ and $u_k^2(n)$, over all m and n. The same applies for the term $u(m)u(n)$. With that, we can write

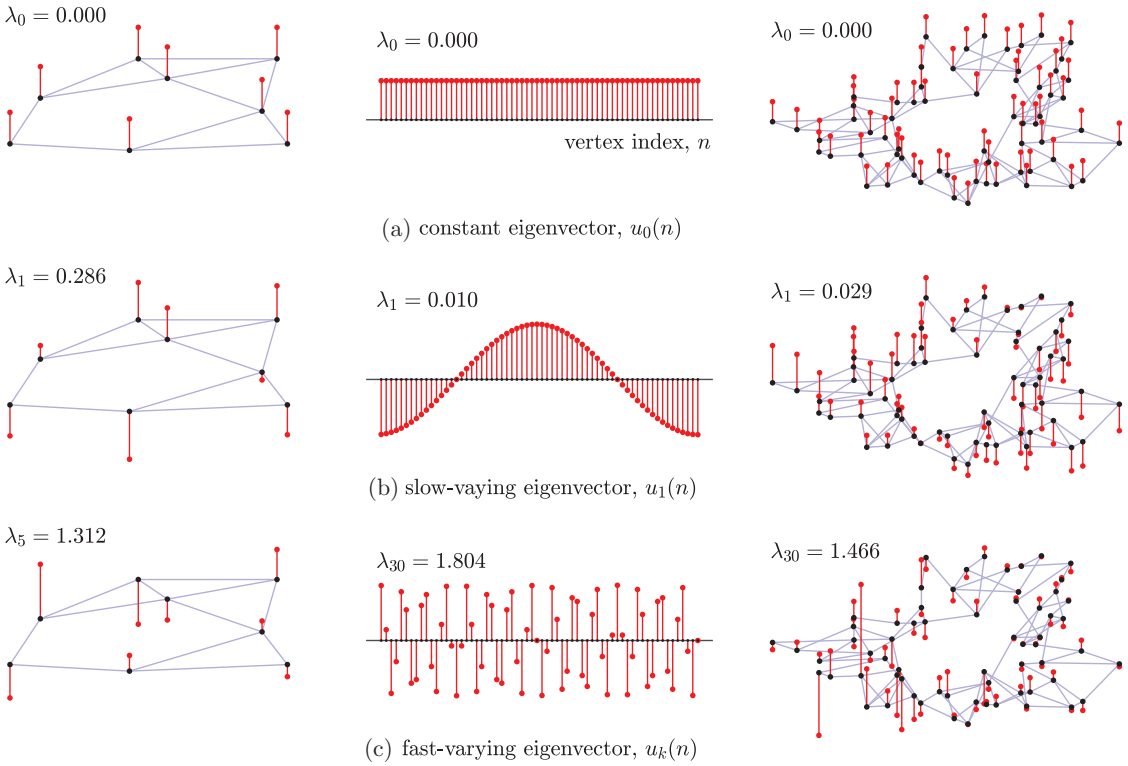
$$\begin{aligned}\mathbf{u}_k^T \mathbf{L} \mathbf{u}_k &= \frac{1}{2} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} W_{mn}(u_k^2(m) - u_k(m)u_k(n)) \\ &\quad + \frac{1}{2} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} W_{mn}(u_k^2(n) - u_k(n)u_k(m)) \\ &= \frac{1}{2} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} W_{mn}(u_k(n) - u_k(m))^2 \geq 0. \end{aligned} \tag{4.8}$$

Obviously, a small $\mathbf{u}_k^T \mathbf{L} \mathbf{u}_k = \lambda_k$ implies that all terms $W_{nm}(u_k(n) - u_k(m))^2 \leq 2\lambda_k$ are also small, thus indicating close values of $u_k(m)$ and $u_k(n)$ for vertices m and n with significant connections, W_{mn}. *The eigenvectors corresponding to a small λ_k are therefore slow-varying and smooth on a graph.*

Example 16: An exemplar of eigenvectors with a small, a moderate and a large smoothness index, λ_k, is given on the three graphs in Figure 4.4.

In order to illustrate the interpretation of the smoothness index in classical time-domain data processing, the time-domain form of the eigenvectors/basis functions in the real-valued Fourier analysis (3.14) is also shown in Figure 4.4 (middle). In this case, the basis functions can be considered as the eigenvectors of a directed circular graph, where the vertices assume the role of time instants.

Observe that in all three graphs the smooth eigenvectors, $\mathbf{u}_0$ and $\mathbf{u}_1$, have similar elements on the neighboring vertices (in the case of a path graph – time instants), and thus may be considered as smooth data on



(a) constant eigenvector, $u_0(n)$

(b) slow-vaying eigenvector, $u_1(n)$

(c) fast-varying eigenvector, $u_k(n)$

Figure 4.4: Illustration of the concept of smoothness of the graph Laplacian eigenvectors for three different graphs: The graph from Figure 2.2 (left), a path graph corresponding to classic temporal data analysis (middle), and an example of a more complex graph with $N = 64$ vertices (right). (a) Constant eigenvector, $u_0(n)$, shown on the three considered graphs. This is the smoothest possible eigenvector for which the smoothness index is $\lambda_0 = 0$. (b) Slow-varying Fiedler eigenvector (the smoothest eigenvector whose elements are not constant), $u_1(n)$, for the three graphs considered. (c) Fast-varying eigenvectors, for $k = 5$ (left), and $k = 30$ (middle and right). Graph vertices are denoted by black circles, and the values of elements of the eigenvectors, $u_k(n)$, by red lines, for $n = 0, 1, \ldots, N-1$. The smoothness index, λ_k, is also given for each case.

the corresponding graph domains. Such similarity does not hold for the fast-varying eigenvectors, $\mathbf{u}_5$ (left of Figure 4.4) and $\mathbf{u}_{30}$ (middle and right of Figure 4.4), which exhibit a much higher smoothness index.

Remark 23: The eigenvector of the graph Laplacian which corresponds to $\lambda_0 = 0$ is constant (maximally smooth for any vertex ordering) and is therefore not appropriate as a template for vertex ordering. The next smoothest eigenvector is $\mathbf{u}_1$, which corresponds to the eigenvalue λ_1.

Ordering of vertices for smoothest Fiedler vector. It is natural to order vertices within a graph in such a way so that the presentation of

the sequence of elements of the smoothest eigenvector, $\mathbf{u}_1$, as a function of the vertex index, n, is also maximally smooth. This can be achieved by sorting (rank ordering) the elements of the Fiedler vector, $\mathbf{u}_1$, in a nondecreasing order. Recall from Remark 12 that the isomorphic nature of graphs means that the reindexing of vertices does not change any graph property. The new order of graph vertices in the sorted $\mathbf{u}_1$ then corresponds to the smoothest sequence of elements of this vector along the vertex index line.

A unique feature of graphs, which renders them indispensable in modern data analytics on irregular domains, is that the ordering of vertices of a graph can be arbitrary, an important difference from classical data analytics where the ordering is inherently sequential and fixed (Stanković *et al.*, 2019). Therefore, in general, any change in data ordering (indexing) would cause significant changes in the results of classical methods, while when it comes to graphs, owing to their topological invariance (as shown in Figures 3.1 and 3.2 in the previous section), reordering of vertices would automatically imply the corresponding reordering of indices within each eigenvector, with no implication on the analysis results. However, the presentation of data sensed at the graph vertices, along a line of vertex indices, as in Figure 3.1(left), a common case for practical reasons, would benefit from an appropriate vertex ordering. Notice that vertex ordering in a graph is just a one-dimensional simplification of an important paradigm in graph analysis, known as graph clustering (Dong *et al.*, 2012; Horaud, 2009; Hamon *et al.*, 2016b; Lu *et al.*, 2014; Masoumi and Hamza, 2017; Masoumi *et al.*, 2016; Mejia *et al.*, 2017).

4.2.2 Spectral Space and Spectral Similarity of Vertices

For a graph with N vertices, the orthogonal eigenvectors of its graph Laplacian form the basis of an N-dimensional space, called the *spectral space*. In this way, the elements, $u_k(n)$, of every eigenvector $\mathbf{u}_k$, $k = 0, 1, 2, \ldots, N-1$, are assigned to the corresponding vertices, $n = 0, 1, 2, \ldots, N-1$, as shown in Figure 4.5(a). This, in turn, means that a set of elements, $u_0(n), u_1(n), u_2(n), \ldots, u_{N-1}(n)$, is assigned to every vertex n, as shown in Figure 4.5(b). For every vertex, n, we can

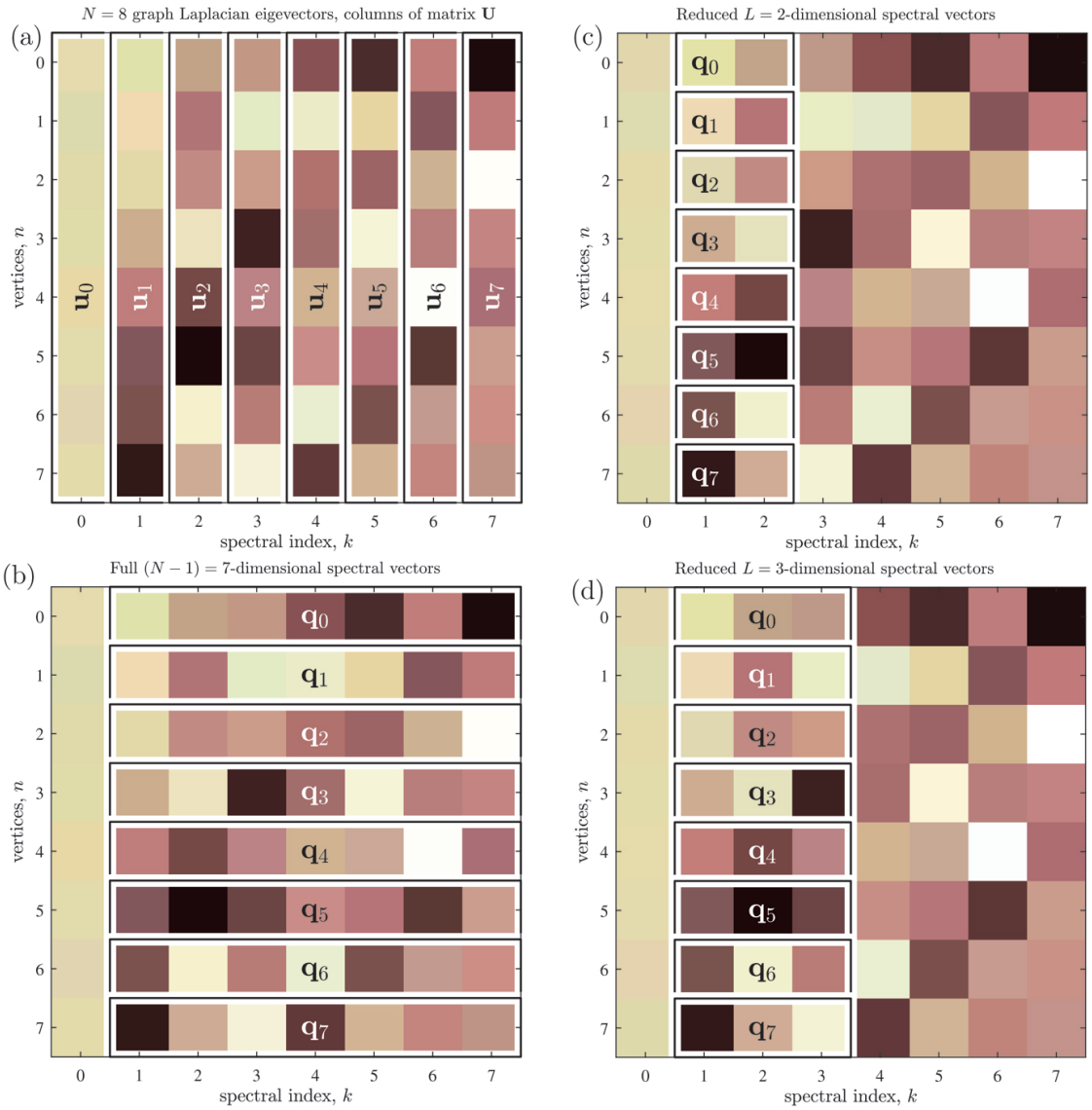


Figure 4.5: Illustration of spectral vectors for the graph from Figure 2.2, with $N = 8$ vertices. For an intuitive analogy with the classical Discrete Fourier Transform, notice that the complex harmonic basis functions within the DFT would play the role of eigenvectors in graph spectral representation, $\mathbf{u}_k$, $k = 0, 1, \ldots, 7$. Then, the spectral vectors, $\mathbf{q}_n$, $n = 0, 1, \ldots, 7$, would be analogous to the basis functions of the inverse Discrete Fourier transform (excluding the first constant element).

then group these elements into an N-dimensional *spectral vector*

$$\mathbf{q}_n \overset{def}{=} [u_0(n), u_1(n), \ldots, u_{N-1}(n)],$$

which is associated with the vertex n. Since the elements of the first eigenvector, $\mathbf{u}_0$, are constant, they do not convey any spectral difference to the graph vertices. Therefore, the elements of $\mathbf{u}_0$ are commonly

omitted from the spectral vector for vertex n, to yield

$$\mathbf{q}_n = [u_1(n), \ldots, u_{N-1}(n)], \tag{4.9}$$

as illustrated in Figure 4.5(b).

Vertex dimensionality in the spectral space. Now that we have associated a unique spectral vector $\mathbf{q}_n$ in (4.9), to every vertex $n = 0, 1, \ldots, N-1$, it is important to note that this $(N-1)$-dimensional representation of every vertex in a graph (whereby the orthogonal graph Laplacian eigenvectors, $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_{N-1}$, serve as a basis of that representation) does not affect the graph itself; this just means that the additional degrees of freedom introduced through spectral vectors facilitate more sophisticated and efficient graph analysis. For example, we may now talk about vertex similarity in the spectral space, or about the spectrum based graph cut, segmentation, and vertex clustering.

An analogy with classical signal processing would be to assign a vector of harmonic basis function values at a time instant (vertex) n, to "describe" this instant, that is, to assign the n-th column of the Discrete Fourier transform matrix to the instant n. This intuition is illustrated in Figures 4.5(a) and (b).

The spectral vectors shall next be used to define spectral similarity of vertices.

Definition: Two vertices, m and n, are called *spectrally similar* if their distance in the spectral space is within a small predefined threshold. The spectral similarity between vertices m and n is typically measured through the Euclidean norm of their spectral space distance, given by

$$d_{mn} \overset{def}{=} \|\mathbf{q}_m - \mathbf{q}_n\|_2.$$

Spectral manifold. Once graph is characterized by the original $(N-1)$-dimensional spectral vectors, the so obtained vertex positions in spectral vertex representation may reside near some well defined surface (commonly a hyperplane) called *a spectral manifold* which is of a reduced dimensionality $M < (N-1)$. The aim of spectral vertex mapping is then to map each spectral vertex representation from the original N-dimensional spectral vector space to a new spectral manifold which lies in a reduced M-dimensional spectral space, to a position closest

to its original $(N-1)$-dimensional spectral position. This principle is related to the Principal Component Analysis (PCA) method, and this relation will be discussed later in this section. *An analogy with classical Discrete Fourier Transform analysis would imply a restriction of the spectral analysis from the space of N harmonics to the reduced space of the M slowest-varying harmonics* (excluding the constant one).

These spectral dimensionality reduction considerations suggest to restrict the definition of spectral similarity to only a few lower-order (smooth) eigenvectors in the spectral space of reduced dimensionality. For example, if the spectral similarity is restricted to the two smoothest eigenvectors, $\mathbf{u}_1$ and $\mathbf{u}_2$ (omitting the constant $\mathbf{u}_0$), then the spectral vector for a vertex n would become

$$\mathbf{q}_n = [u_1(n), u_2(n)],$$

as illustrated in Figures 4.5(c) and 4.6(a). If for two vertices, m and n, the values of $u_1(m)$ are close to $u_1(n)$ and the values of $u_2(m)$ are close to $u_2(n)$, then these two vertices are said to be *spectrally similar*, that is, they exhibit a small spectral distance, $d_{mn} = \|\mathbf{q}_m - \mathbf{q}_n\|_2$.

Finally, the simplest spectral description uses only one (smoothest nonconstant) eigenvector to describe the spectral content of a vertex, so that the spectral vector reduces to a spectral scalar

$$\mathbf{q}_n = [q_n] = [u_1(n)],$$

whereby the so reduced spectral space is a one-dimensional line.

Example 17: The two-dimensional and three-dimensional spectral vectors, $\mathbf{q}_n = [u_1(n), u_2(n)]$ and $\mathbf{q}_n = [u_1(n), u_2(n), u_3(n)]$, of the graph from Figure 2.2 are shown in Figure 4.6, for $n = 2$ and $n = 6$.

Spectral embedding. The mapping from the reduced dimensionality spectral space back onto the original vertices is referred to as *Spectral embedding*.

We can proceed in two ways with the reduced spectral vertex space representation: (i) to assign the reduced dimension spectral vectors to the original vertex positions, for example, in the form of vertex coloring, as a basis for subsequent vertex clustering (Section 4.2.3), or (ii) to achieve new vertex positioning in the reduced dimensionality space of

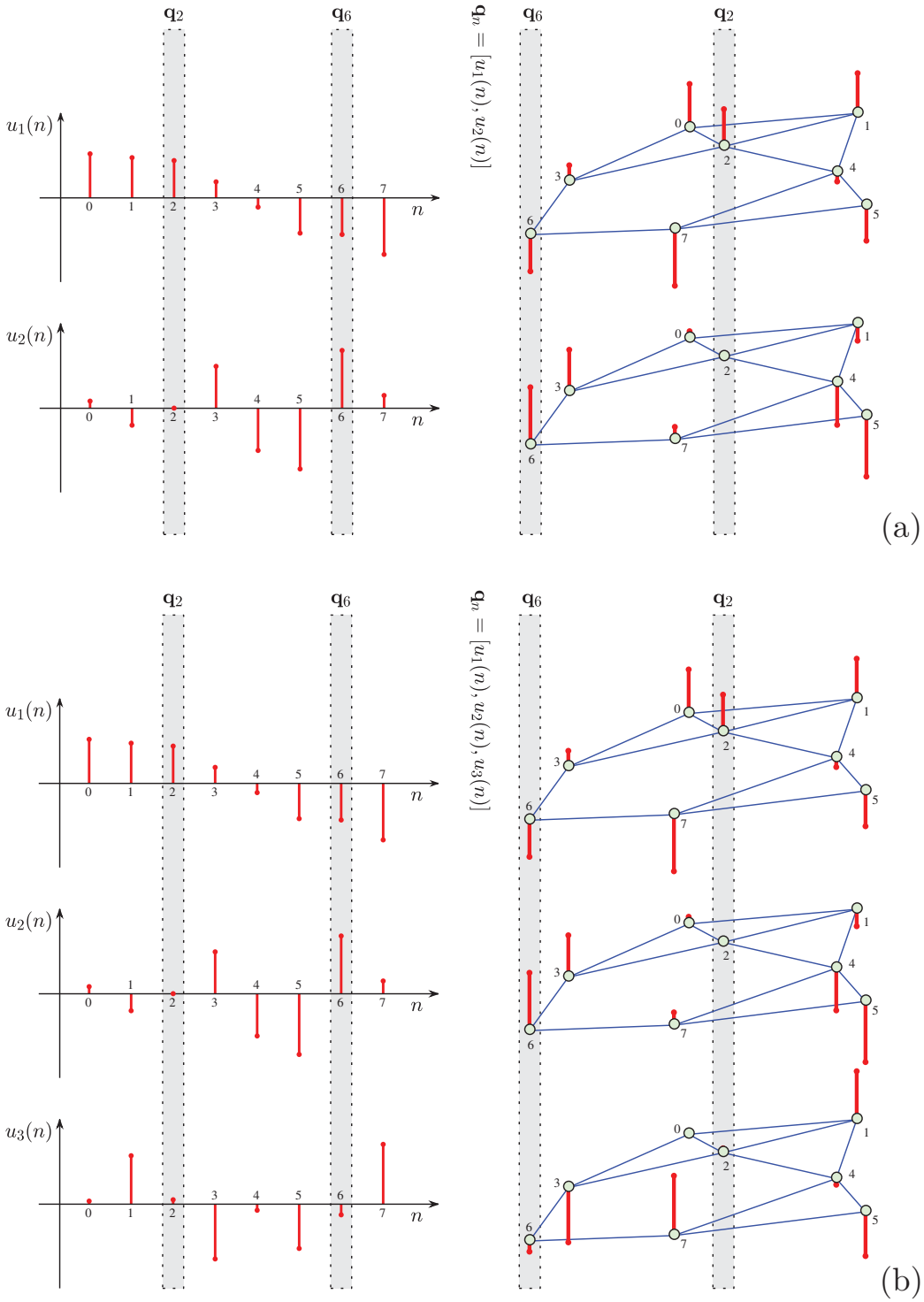


Figure 4.6: Illustration of the spectral vectors, $\mathbf{q}_n = [u_1(n), u_2(n)]$ and $\mathbf{q}_n = [u_1(n), u_2(n), u_3(n)]$, for the Laplacian matrix of the graph in Figure 2.2. (a) Two-dimensional spectral vectors, $\mathbf{q}_2 = [u_1(2), u_2(2)]$ and $\mathbf{q}_6 = [u_1(6), u_2(6)]$. (b) Three-dimensional spectral vectors, $\mathbf{q}_2 = [u_1(2), u_2(2), u_3(2)]$ and $\mathbf{q}_6 = [u_1(6), u_2(6), u_3(6)]$. For clarity, the spectral vectors are shown on both the vertex index axis and directly on graph.

eigenvectors (reduced spectral space), using eigenmaps (Section 4.4). Both yield similar information and can be considered as two sides of the same coin (Belkin and Niyogi, 2003). For visualization purposes, we will use colors of the RGB system to represent the spectral vector values in a reduced (one, two, or three) dimensional spectral space. Vertices at the original graph positions will be colored according to the spectral vector values.

4.2.3 Indicator Vector

Remark 21 shows that the combinatorial approach to minimum cut problem is computationally infeasible, as even for a graph with only 50 vertices we have $5.6 \cdot 10^{14}$ such potential cuts.

To break this *Curse of Dimensionality* it would be very convenient to relate the problem of the minimization of the ratio cut in (4.1) and (4.2) to that of eigenanalysis of graph Laplacian. To this end, we shall introduce the notion of *indicator vector* $\mathbf{x}$ on a graph, for which the elements take subgraph-wise constant values within each disjoint subset (cluster) of vertices, with these constants taking different values for different clusters of vertices (*subset-wise constant vector*). While this does not immediately reduce the computational burden (the same number of combinations remains as in the brute force method), the elements of $\mathbf{x}$ now uniquely reflect the assumed cut of the graph into disjoint subsets $\mathcal{E}, \mathcal{H} \subset \mathcal{V}$.

Establishing *a further link with only the smoothest eigenvector of the graph Laplacian will convert the original, computationally intractable, combinatorial minimum cut problem into a manageable algebraic eigenvalue problem, for which the computation complexity is of the* $\mathcal{O}(N^3)$ *order.* By casting the problem into the linear algebra framework, complexity of calculation can be additionally reduced through efficient eigenanalysis methods, such as the *Power Method* which sequentially computes the desired number of largest eigenvalues and the corresponding eigenvectors, at an affordable $\mathcal{O}(N^2)$ computations per iteration, as shown in the appendix.

However, unlike the indicator vector, $\mathbf{x}$, the smoothest eigenvector (corresponding to the smallest nonzero eigenvalue) of graph Laplacian

is not subset-wise constant, and such solution would be approximate, but computationally feasible.

Remark 24: The concept of indicator vector can be introduced through the analysis with an *ideal minimum cut* of a graph, given by

$$Cut(\mathcal{E},\mathcal{H}) = \sum_{\substack{m\in\mathcal{E}\\ n\in\mathcal{H}}} W_{mn} = 0,$$

that is, when considering an already disjoint graph for which $Cut(\mathcal{E},\mathcal{H}) = 0$ indicates that there exist no edges between the subsets $\mathcal{E}$ and $\mathcal{H}$, that is, $W_{mn} = 0$ for $m \in \mathcal{E}$, and $n \in \mathcal{H}$. Obviously, this ideal case can be solved without resorting to the combinatorial approach, since this graph is already in the form of two disconnected subgraphs, defined by the sets of vertices $\mathcal{E}$ and $\mathcal{H}$. For such a disconnected graph, the second eigenvalue of the graph Laplacian is $\lambda_1 = 0$, as established by the graph Laplacian property L_2. When $\lambda_1 = 0$, then

$$2\mathbf{u}_1^T\mathbf{L}\mathbf{u}_1 = \sum_{m=0}^{N-1}\sum_{n=0}^{N-1} W_{mn}(u_1(n) - u_1(m))^2 = 2\lambda_1 = 0,$$

which follows from (4.6) and (4.8). Since all terms in the last sum are nonnegative, this implies that they must be zero-valued, that is, the eigenvector $\mathbf{u}_1$ is subset-wise constant, with $u_1(n) = u_1(m) = c_1$ for $m, n \in \mathcal{E}$ and $u_1(n) = u_1(m) = c_2$ for $m, n, \in \mathcal{H}$. Since the eigenvector $\mathbf{u}_1$ is orthogonal to the constant eigenvector $\mathbf{u}_0$, then $\sum_{n=0}^{N-1} u_1(n) = 0$. A possible solution for $u_1(n)$, that satisfies the subset-wise constant form and has zero mean, is $u_1(n) = c_1 = 1/N_{\mathcal{E}}$ for $n \in \mathcal{E}$ and $u_1(n) = c_2 = -1/N_{\mathcal{H}}$ for $n \in \mathcal{H}$. We can conclude that the problem of finding an ideal minimum cut can indeed be solved by introducing an indicator vector $\mathbf{x} = \mathbf{u}_1$, such that $x(n) = 1/N_{\mathcal{E}}$ for $n \in \mathcal{E}$ and $x(n) = -1/N_{\mathcal{H}}$ for $n \in \mathcal{H}$. The membership of a vertex, n, to either the subset $\mathcal{E}$ or $\mathcal{H}$ of the ideal minimum cut is therefore uniquely defined by the sign of indicator vector $\mathbf{x} = \mathbf{u}_1$. This form of $\mathbf{x}$ is not normalized to unit energy, as its scaling by any constant would not influence solution for vertex clustering into subsets $\mathcal{E}$ or $\mathcal{H}$.

For a general graph, and following the above reasoning, we here consider two specific subset-wise constant forms of the indicator vector, $\mathbf{x}$, which are based on

(i) the number of vertices in disjoint subgraphs,

$$x(n) = \begin{cases} \dfrac{1}{N_{\mathcal{E}}}, & \text{for } n \in \mathcal{E} \\ -\dfrac{1}{N_{\mathcal{H}}}, & \text{for } n \in \mathcal{H}, \end{cases} \tag{4.10}$$

where $N_{\mathcal{E}}$ is the number of vertices in $\mathcal{E}$, and $N_{\mathcal{H}}$ is the number of vertices in $\mathcal{H}$, and
(ii) the volumes of the disjoint subgraphs,

$$x(n) = \begin{cases} \dfrac{1}{V_{\mathcal{E}}}, & \text{for } n \in \mathcal{E} \\ -\dfrac{1}{V_{\mathcal{H}}}, & \text{for } n \in \mathcal{H}, \end{cases} \tag{4.11}$$

where *the volumes of the sets*, $V_{\mathcal{E}}$ and $V_{\mathcal{H}}$, are defined as the sums of all vertex degrees, D_{nn}, in the corresponding subsets, $V_{\mathcal{E}} = \sum_{n\in\mathcal{E}} D_{nn}$ and $V_{\mathcal{H}} = \sum_{n\in\mathcal{H}} D_{nn}$.

Before proceeding further with the analysis of these two forms of indicator vector (in the next two remarks), it is important to note that if we can find the vector $\mathbf{x}$ which minimizes the ratio cut, $CutN(\mathcal{E},\mathcal{H})$ in (4.1), then the elements of vector $\mathbf{x}$ (their signs, $\text{sign}(x(n)) = 1$ for $n \in \mathcal{E}$ and $\text{sign}(x(n)) = -1$ for $n \in \mathcal{H}$) may be used to decide whether to associate a vertex, n, to either the set $\mathcal{E}$ or $\mathcal{H}$ of the minimum ratio cut.

Remark 25: The ratio cut, $CutN(\mathcal{E},\mathcal{H})$, defined in (4.1), for the indicator vector $\mathbf{x}$ with the elements $x(n) = 1/N_{\mathcal{E}}$ for $n \in \mathcal{E}$ and $x(n) = -1/N_{\mathcal{H}}$ for $n \in \mathcal{H}$, is equal to the *Rayleigh quotient* of the matrix $\mathbf{L}$ and vector $\mathbf{x}$, that is

$$CutN(\mathcal{E},\mathcal{H}) = \frac{\mathbf{x}^T\mathbf{L}\mathbf{x}}{\mathbf{x}^T\mathbf{x}}. \tag{4.12}$$

To prove this relation we rewrite (4.8) as

$$\mathbf{x}^T\mathbf{L}\mathbf{x} = \frac{1}{2}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1} W_{mn}(x(n) - x(m))^2. \tag{4.13}$$

For all vertices m and n within the same subgraph, that is, such that $m \in \mathcal{E}$ and $n \in \mathcal{E}$, the elements of vector $\mathbf{x}$ are therefore the same

and equal to $x(m) = x(n) = 1/N_{\mathcal{E}}$. In turn, this means that the terms $(x(n) - x(m))^2$ in (4.13) are zero-valued. The same holds for any two vertices belonging to the set $\mathcal{H}$, that is, for $m \in \mathcal{H}$ and $n \in \mathcal{H}$. Therefore, only the terms corresponding to the edges which define the cut, when $m \in \mathcal{E}$ and $n \in \mathcal{H}$, and vice versa, remain in the sum, and they are constant and equal to $(x(n) - x(m))^2 = (1/N_{\mathcal{E}} - (-1/N_{\mathcal{H}}))^2$, to yield

$$\begin{aligned} \mathbf{x}^T \mathbf{L} \mathbf{x} &= \left(\frac{1}{N_{\mathcal{E}}} + \frac{1}{N_{\mathcal{H}}} \right)^2 \sum_{\substack{m \in \mathcal{E} \\ n \in \mathcal{H}}} W_{mn} \\ &= \left(\frac{1}{N_{\mathcal{E}}} + \frac{1}{N_{\mathcal{H}}} \right) CutN(\mathcal{E}, \mathcal{H}), \end{aligned} \tag{4.14}$$

where the ratio cut, $CutN(\mathcal{E}, \mathcal{H})$, is defined in (4.1). Finally, from the energy of the indicator vector, $\mathbf{x}^T \mathbf{x} = e_x^2$,

$$\mathbf{x}^T \mathbf{x} = \|\mathbf{x}\|_2^2 = e_x^2 = \frac{N_{\mathcal{E}}}{N_{\mathcal{E}}^2} + \frac{N_{\mathcal{H}}}{N_{\mathcal{H}}^2} = \frac{1}{N_{\mathcal{E}}} + \frac{1}{N_{\mathcal{H}}}, \tag{4.15}$$

which proves (4.12).

The same analysis holds if *the indicator vector is normalized to unit energy*, whereby $x(n) = 1/(N_{\mathcal{E}} e_x)$ for $n \in \mathcal{E}$ and $x(n) = -1/(N_{\mathcal{H}} e_x)$ for $n \in \mathcal{H}$, with e_x defined in (4.15) as $e_x = \|\mathbf{x}\|_2$.

We can therefore conclude that the indicator vector, $\mathbf{x}$, which solves the problem of minimization of the ratio cut, is also a solution to (4.12). This minimization problem, for the unit energy form of the indicator vector the elements of which are the minimization variables, can also be written as

$$\min_{\mathbf{x}} \{\mathbf{x}^T \mathbf{L} \mathbf{x}\} \text{ subject to } \mathbf{x}^T \mathbf{x} = 1. \tag{4.16}$$

In general, this is again a combinatorial problem, since all possible combinations of subsets of vertices, $\mathcal{E}$ and $\mathcal{H}$, together with the corresponding indicator vectors, $\mathbf{x}$, must be considered.

For a moment we shall put aside the very specific (subset-wise constant) form of the indicator vector and consider the general minimization problem in (4.16). This problem can be solved using the method of Lagrange multipliers, with the corresponding cost function

$$\mathcal{L}(\mathbf{x}) = \mathbf{x}^T \mathbf{L} \mathbf{x} - \lambda (\mathbf{x}^T \mathbf{x} - 1).$$

From $\partial\mathcal{L}(\mathbf{x})/\partial\mathbf{x}^T = \mathbf{0}$, it follows that $\mathbf{L}\mathbf{x} = \lambda\mathbf{x}$, which is precisely the eigenvalue/eigenvector relation for the graph Laplacian $\mathbf{L}$, the solution of which is $\lambda = \lambda_k$ and $\mathbf{x} = \mathbf{u}_k$, for $k = 0, 1, \ldots, N-1$. In other words, upon replacing $\mathbf{x}$ by $\mathbf{u}_k$ in the term $\min\{\mathbf{x}^T\mathbf{L}\mathbf{x}\}$ above, we obtain $\min_k\{\mathbf{u}_k^T\mathbf{L}\mathbf{u}_k\} = \min_k\{\lambda_k\}$. After neglecting the trivial solution $\lambda_0 = 0$, since it produces a constant eigenvector $\mathbf{u}_0$, we next arrive at $\min_k\{\lambda_k\} = \lambda_1$ and $\mathbf{x} = \mathbf{u}_1$. Note that this solution yields a general form of vector $\mathbf{x}$ that minimizes (4.12). However, such a form does not necessarily correspond to a subset-wise constant indicator vector, $\mathbf{x}$. The fact that the trivial solution (constant vector $\mathbf{x}$) is neglected, is commonly written as an additional constraint in (4.16), of the form (see also Part III, Section 13.3),

$$\mathbf{x}^T\mathbf{1} = \mathbf{0}.$$

4.2.4 Bounds on the Minimum Cut

In general, the subset-wise constant indicator vector, $\mathbf{x}$, may be written as a linear combination of the graph Laplacian eigenvectors, $\mathbf{u}_k$, $k = 1, 2, \ldots, N-1$, to give

$$\mathbf{x} = \alpha_1\mathbf{u}_1 + \alpha_2\mathbf{u}_2 + \cdots + \alpha_{N-1}\mathbf{u}_{N-1}. \tag{4.17}$$

This kind of vector expansion onto the set of eigenvectors shall be addressed in more detail in Part II of this monograph. Note that the constant eigenvector $\mathbf{u}_0$ is omitted since the indicator vector is zero-mean by definition (orthogonal to a constant vector). The calculation of coefficients α_i would require the indicator vector (that is, the sets $\mathcal{E}$ and $\mathcal{H}$) to be known, leading again to the combinatorial problem of vertex set partitioning. It is interesting to note that the quadratic form of indicator vector, $\mathbf{x}$, given by (4.17) is equal to $\mathbf{x}^T\mathbf{L}\mathbf{x} = \alpha_1^2\lambda_1 + \alpha_2^2\lambda_2 + \cdots + \alpha_{N-1}^2\lambda_{N-1}$, and that it assumes the minimum value for $\alpha_1 = 1$, $\alpha_2 = \cdots = \alpha_{N-1} = 0$, that is, when $\mathbf{x} = \mathbf{u}_1$, which corresponds to imposing the normalized energy condition, $\mathbf{x}^T\mathbf{x} = \alpha_1^2 + \alpha_2^2 + \cdots + \alpha_{N-1}^2 = 1$. In other words, we now arrive at a physically meaningful bound

$$\lambda_1 \leq \mathbf{x}^T\mathbf{L}\mathbf{x} = CutN(\mathcal{E}, \mathcal{H}).$$

Observe that this inequality corresponds to the lower Cheeger bound for the minimum ratio cut in (4.1).

Remark 26: If the space of approximative solutions for the indicator vector, $\mathbf{x}$, is relaxed to allow for vectors that are not subset-wise constant (while omitting the constant eigenvector of the graph Laplacian, $\mathbf{u}_0$), the approximative solution becomes $\mathbf{x} = \mathbf{u}_1$ (as previously shown and illustrated in Example (18)). The above analysis indicates that this solution is *quasi-optimal*, however, despite its simplicity, the graph cut based on only the second graph Laplacian eigenvector, $\mathbf{u}_1$, typically produces a good approximation to the optimal (minimum ratio) cut.

It has been shown that the value of the true ratio minimum cut in (4.1), when the indicator vector $\mathbf{x}$ is subset-wise constant, is bounded on both sides (upper and lower) with the constants which are proportional to the smallest nonzero eigenvalue, $\mathbf{u}_1^T \mathbf{L} \mathbf{u}_1 = \lambda_1$, of the graph Laplacian. The simplest form of these bounds (Cheeger's bounds) for the cut defined by (4.5), has the form (Alon, 1986; Chung, 2005, 2007; Trevisan, 2013)

$$\frac{\lambda_1}{2} \leq \phi(\mathcal{V}) \overset{def}{=} \min_{\mathcal{E} \subset \mathcal{V}} \{\phi(\mathcal{E})\} \leq \sqrt{2\lambda_1}. \tag{4.18}$$

This shows that the eigenvalue λ_1 is also a good *measure of a graph separability and consequently the quality of spectral clustering* in the sense of a minimum Cheeger's ratio cut. The value of the minimum Cheeger's ratio cut of a graph (also referred to as *Cheeger's constant, conductivity, or isoperimetric number* of a graph) may also be considered as a numerical measure of the presence of a *"bottleneck"* in a graph.

4.2.5 Indicator Vector for Normalized Graph Laplacian

We shall now address the cut based on normalized graph Laplacian, in light of the above analysis.

Remark 27: The volume normalized cut, $CutV(\mathcal{E}, \mathcal{H})$, defined in (4.2), is equal to

$$CutV(\mathcal{E}, \mathcal{H}) = \frac{\mathbf{x}^T \mathbf{L} \mathbf{x}}{\mathbf{x}^T \mathbf{D} \mathbf{x}}, \tag{4.19}$$

where the corresponding, subset-wise constant, indicator vector has the values $x(n) = 1/V_{\mathcal{E}}$ for $n \in \mathcal{E}$ and $x(n) = -1/V_{\mathcal{H}}$ for $n \in \mathcal{H}$, while the volumes of the sets, $V_{\mathcal{E}}$ and $V_{\mathcal{H}}$, are defined in (4.2).

The proof is identical to that given in Remark 25. For the normalized indicator vector, we have $\mathbf{x}^T\mathbf{D}\mathbf{x} = 1$, so that the minimization problem in (4.19), for finding the elements of $\mathbf{x}$, reduces to

$$\min\{\mathbf{x}^T\mathbf{L}\mathbf{x}\} \text{ subject to } \mathbf{x}^T\mathbf{D}\mathbf{x} = 1. \tag{4.20}$$

If the solution space is restricted to the space of *generalized eigenvectors of the graph Laplacian*, defined by

$$\mathbf{L}\mathbf{u}_k = \lambda_k\mathbf{D}\mathbf{u}_k,$$

then the solution to (4.20) becomes

$$\mathbf{x} = \mathbf{u}_1,$$

where $\mathbf{u}_1$ is the generalized eigenvector of the graph Laplacian that corresponds to the lowest nonzero eigenvalue. The fact that the trivial solution (constant vector $\mathbf{x}$) is avoided as a solution, can be written in the form of an additional constraint, $\mathbf{x}^T\mathbf{1} = \mathbf{0}$, in (4.20).

The *eigenvectors of the normalized Laplacian*, $\mathbf{L}_N = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$, may also be used in optimal cut approximations since the minimization problem in (4.19) can be rewritten using the normalized Laplacian through a change of the variable, to yield

$$\mathbf{x} = \mathbf{D}^{-1/2}\mathbf{y},$$

which allows us to arrive at the following form Ng *et al.* (2002)

$$\begin{aligned}\min\{\mathbf{y}^T\mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}\mathbf{y}\} = \min\{\mathbf{y}^T\mathbf{L}_N\mathbf{y}\},\\ \text{subject to } \mathbf{y}^T\mathbf{y} = 1.\end{aligned} \tag{4.21}$$

If the space of solutions to this minimization problem is relaxed to the eigenvectors, $\mathbf{v}_k$, of the normalized graph Laplacian, $\mathbf{L}_N$, then $\mathbf{y} = \mathbf{v}_1$. For more detail on the various forms of the eigenvalues and eigenvectors of graph Laplacian, we refer to Table 4.1.

It is obvious now from (4.20) and (4.21) that the relation of the form $\mathbf{x} = \mathbf{D}^{-1/2}\mathbf{y}$ also holds for the corresponding eigenvectors of the normalized graph Laplacian, $\mathbf{v}_k$, and the generalized eigenvectors of the Laplacian, $\mathbf{v}_k$, that is,

$$\mathbf{u}_k = \mathbf{D}^{-1/2}\mathbf{v}_k.$$

Table 4.1: Summary of graph embedding mappings. The *Graph Laplacian mapping*, the *Generalized eigenvectors of the Laplacian mapping*, the *Normalized Laplacian mapping*, the *Commute time mapping*, the *Diffusion mapping*, and the *Cumulative diffusion mapping*.

Mapping	**Eigen-Analysis Relation**	**Reduced Dimensionality Spectral Vector**
Graph Laplacian mapping	$\mathbf{L}\mathbf{u}_k = \lambda_k \mathbf{u}_k$	$\mathbf{q}_n = [u_1(n), u(2), \ldots, u_M(n)]$
Generalized eigenvectors of Laplacian mapping	$\mathbf{L}\mathbf{u}_k = \lambda_k \mathbf{D}\mathbf{u}_k$	$\mathbf{q}_n = [u_1(n), u(2), \ldots, u_M(n)]$
Normalized Laplacian mapping	$(\mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2})\mathbf{u}_k = \lambda_k \mathbf{u}_k$	$\mathbf{q}_n = [u_1(n), u(2), \ldots, u_M(n)]$
Commute time mapping	$\mathbf{L}\mathbf{u}_k = \lambda_k \mathbf{u}_k$	$\mathbf{q}_n = [\frac{u_1(n)}{\sqrt{\lambda_1}}, \frac{u_2(n)}{\sqrt{\lambda_2}}, \ldots, \frac{u_M(n)}{\sqrt{\lambda_M}}]$
Diffusion (random walk) mapping	$\mathbf{L}\mathbf{u}_k = \lambda_k \mathbf{D}\mathbf{u}_k$	$\mathbf{q}_n = [u_1(n)(1-\lambda_1)^t, \ldots, u_M(n)(1-\lambda_M)^t]$
Cumulative diffusion mapping	$\mathbf{L}\mathbf{u}_k = \lambda_k \mathbf{D}\mathbf{u}_k$	$\mathbf{q}_n = [\frac{u_1(n)}{\lambda_1}, \frac{u_2(n)}{\lambda_2}, \ldots, \frac{u_M(n)}{\lambda_M}]$

It is important to note that, in general, clustering results based on the three forms of eigenvectors:

(i) the smoothest *graph Laplacian eigenvector*,

(ii) the smoothest *generalized eigenvector of the Laplacian*, and

(iii) the smoothest *eigenvector of the normalized Laplacian*,

are different. While the method (i) favors the clustering into subsets with (almost) equal number of vertices, the methods (ii) and (iii) favor subsets with (almost) equal volumes (defined as sums of the vertex degrees in the subsets). Also note that the methods (i) and (ii) approximate the indicator vector in different eigenvector subspaces. All three methods will produce the same clustering result for unweighted regular graphs, for which the volumes of subsets are proportional to the number of their corresponding vertices, while the eigenvectors for all the three Laplacian forms are the same in regular graphs, as shown in (2.13).

Generalized eigenvectors of the graph Laplacian and eigenvectors of the normalized Laplacian. Recall that the matrix $\mathbf{D}^{-1/2}$ is of a diagonal form, and with positive elements. Then, the solution to (4.20) which is equal to the generalized eigenvector of the graph Laplacian, and the solution to (4.21) which is equal to the eigenvector of the normalized Laplacian, are related as $\text{sign}(\mathbf{y}) = \text{sign}(\mathbf{x})$ or $\text{sign}(\mathbf{v}_1) = \text{sign}(\mathbf{u}_1)$. This indicates that if the sign of the corresponding eigenvector is used for the minimum cut approximation (clustering), both results are the same.

4.3 Spectral Clustering Implementation

Spectral clustering is most conveniently implemented using only low-dimensional spectral vectors, with the simplest case when only a one-dimensional spectral vector is used as indicator vector. More degrees of freedom can be achieved by clustering schemes which use two or three Laplacian eigenvectors, as discussed next.

4.3.1 Clustering Based on Only One (Fiedler) Eigenvector

From the analysis in the previous section, we can conclude that only the smoothest eigenvector, $\mathbf{u}_1$, can produce a good (quasi-optimal) approximation to the problem of minimum ratio cut graph clustering into two subsets of vertices, $\mathcal{E}$ and $\mathcal{H}$. Within the concept of spectral vectors, presented in Section 4.2.2, this indicates that the simplest form of spectral vector, $\mathbf{q}_n = u_1(n)$, based on just one (the smoothest) Fiedler eigenvector, $\mathbf{u}_1$, can be used for efficient *spectral vertex clustering*. Since the spectral vector $\mathbf{q}_n = u_1(n)$ is used as an approximative solution to the indicator vector within the minimum ratio cut definition, its values may be normalized. One such normalization

$$\mathbf{y}_n = \mathbf{q}_n / \|\mathbf{q}_n\|_2 \tag{4.22}$$

yields a two-level form of the spectral vector

$$\mathbf{y}_n = [u_1(n)/\|u_1(n)\|_2] = [\text{sign}(u_1(n))],$$

and represents a step before clustering, as proposed in Ng *et al.* (2002). This is justified based on the original form of the indicator vector, whose sign indicates the vertex association to either subset $\mathcal{E}$ or $\mathcal{H}$. For illustrative representation of the normalized spectral vector, we may use a simple two-level colormap and assign one of two colors to each vertex. Such a simple algorithm for clustering is given in Algorithm 1 (for an algorithm with more options for clustering and representation see the appendix (Algorithm 3) and Remarks 30 and 33).

Example 18: Consider the graph from Figure 2.2 and its Laplacian eigenvector, $\mathbf{u}_1$, from Figure 3.4. The elements of this single eigenvector, $\mathbf{u}_1$, are used to encode the vertex colormap, as shown in Figure 4.7(a). Here, the minimum element of $\mathbf{u}_1$ was used to select the red color (vertex 7), while the white color at vertex 0 was designated by the maximum value of this eigenvector. Despite its simplicity, this scheme immediately allows us to threshold $\mathbf{u}_1$ and identify two possible *graph clusters*, $\{0, 1, 2, 3\}$, and $\{4, 5, 6, 7\}$, as illustrated in Figure 4.7(b). The same result would be obtained if the sign of $\mathbf{u}_1$ was used to color the vertices, and this would correspond to the minimum ratio cut clustering in Figure 4.3.

Algorithm 1. Clustering using the graph Laplacian.

Input:

- Graph vertices $\mathcal{V} = \{0, 1, \ldots, N-1\}$
- Graph Laplacian $\mathbf{L}$

1: $[\mathbf{U}, \mathbf{\Lambda}] \leftarrow \text{eig}(\mathbf{L})$
2: $y_n \leftarrow U(2, n)$
3: $\mathcal{E} \leftarrow \{n \,|\, y_n > 0\}, \quad \mathcal{H} \leftarrow \{n \,|\, y_n \leq 0\}$

Output:

- Vertex clusters $\mathcal{E}$ and $\mathcal{H}$

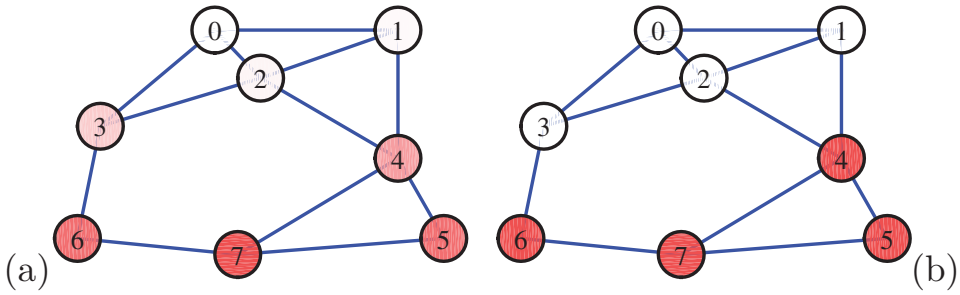


Figure 4.7: Vertex coloring for the graph from Figure 2.2, with its spectrum shown in Figure 3.4. (a) The eigenvector, $\mathbf{u}_1$, of the Laplacian matrix of this graph, given in (2.8), is normalized and is used to define the red color intensity levels within the colormap for every vertex. For this example, $\mathbf{u}_1 = [0.42, 0.38, 0.35, 0.15, -0.088, -0.34, -0.35, -0.54]^T$. The largest element of this eigenvector is $u_1(0) = 0.42$ at vertex 0, which indicates that this vertex should be colored by the lowest red intensity (white), while the smallest element is $u_1(7) = -0.54$, so that vertex 7 is colored with the strongest red color intensity. (b) Simplified two-level coloring based on the sign of the elements of eigenvector $\mathbf{u}_1$.

The true indicator vector, $\mathbf{x}$, for the minimum ratio cut of this graph is presented in Figure 4.8(a). This vector is obtained by checking all the 127 possible cut combinations of $\mathcal{E}$ and $\mathcal{H}$ in this small graph, together with the corresponding $x(n)$. The signs of the elements of this vector indicate the way for optimal clustering into the subsets $\mathcal{E} = \{0, 1, 2, 3\}$ and $\mathcal{H} = \{4, 5, 6, 7\}$, while the minimum cut value is

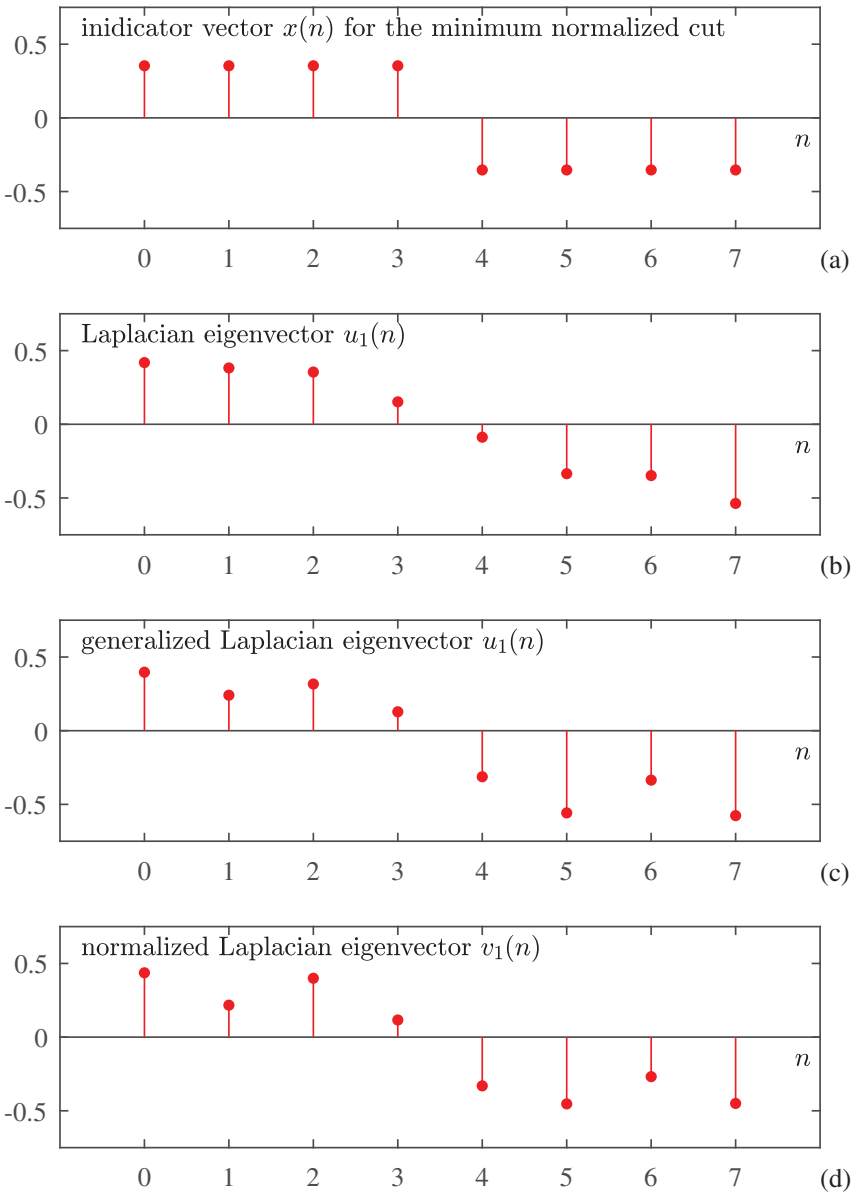


Figure 4.8: Principle of the minimum ratio cut based clustering and its spectral (graph Laplacian eigenvector) based approximation; all vectors are plotted against the vertex index n. (a) The ideal indicator vector for a minimum ratio cut, $CutN(\mathcal{E}, \mathcal{H})$, normalized to unit energy. (b) The graph Laplacian eigenvector, $\mathbf{u}_1$. (c) The generalized eigenvector of the Laplacian, $\mathbf{u}_1$. (d) The eigenvector of the normalized Laplacian, $\mathbf{v}_1$. The eigenvectors in (c) and (d) are related as $\mathbf{u}_1 = \mathbf{D}^{-1/2}\mathbf{v}_1$. In this case, the signs of the indicator vector and the eigenvectors, sign($\mathbf{x}$), sign($\mathbf{u}_1$), and sign($\mathbf{v}_1$) are the same in all the four vectors. The signs of these vectors then all may be used to define the minimum ratio cut based clustering into $\mathcal{E}$ and $\mathcal{H}$, that is, the association of a vertex, n, to either the subset $\mathcal{E}$ or subset $\mathcal{H}$.

$CutN(\mathcal{E},\mathcal{H}) = \mathbf{x}^T\mathbf{L}\mathbf{x} = 0.395$. Figure 4.8(b) shows an approximation of the indicator vector within the space of the graph Laplacian eigenvector, $\mathbf{u}_1$. The quadratic form of the eigenvector, $\mathbf{u}_1$, is equal to $\mathbf{u}_1^T\mathbf{L}\mathbf{u}_1 = \lambda_1 = 0.286$. As shown in (4.17), note that the true indicator vector, $\mathbf{x}$, can be decomposed into the set of all graph Laplacian eigenvectors, $\mathbf{u}_k$, and written as their linear combination.

The generalized Laplacian eigenvector, $\mathbf{u}_1 = [0.37, 0.24, 0.32, 0.13, -0.31, -0.56, -0.34, -0.58]$, which is an approximation of the indicator vector for the minimum volume normalized cut in (4.2), is presented in Figure 4.8(c). In this case, the generalized eigenvector indicates the same clustering subsets, $\mathcal{E} = \{0, 1, 2, 3\}$ and $\mathcal{H} = \{4, 5, 6, 7\}$. The eigenvector of the normalized Laplacian, $\mathbf{v}_1$, is shown in Figure 4.8(d).

Example 19: Consider the graph from Figure 2.2, with the weight matrix, $\mathbf{W}$, in (2.4), and the graph Laplacian eigenvector $\mathbf{u}_1$ (shown in Figure 3.4, Figure 4.4(b) (left), and Figure 4.8(b)). When this eigenvector is thresholded to only two intensity levels, $\text{sign}(\mathbf{u}_1)$, two graph clusters are obtained, as shown in Figure 4.7(b). In an ideal case, these clusters may even be considered as independent graphs (graph segmentation being the strongest form of clustering); this can be achieved by redefining the weights as $W_{nm} = 0$, if m and n are in different clusters, and $W_{nm} = W_{nm}$ otherwise (Ng *et al.*, 2002), for the corresponding disconnected (segmented) graph, whose weight matrix, $\hat{\mathbf{W}}$, is given by

$$\hat{\mathbf{W}} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{cccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \left[\begin{array}{cccccccc} 0 & 0.23 & 0.74 & 0.24 & 0 & 0 & 0 & 0 \\ 0.23 & 0 & 0.35 & 0 & 0 & 0 & 0 & 0 \\ 0.74 & 0.35 & 0 & 0.26 & 0 & 0 & 0 & 0 \\ 0.24 & 0 & 0.26 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.51 & 0 & 0.14 \\ 0 & 0 & 0 & 0 & 0.51 & 0 & 0 & 0.15 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.32 \\ 0 & 0 & 0 & 0 & 0.14 & 0.15 & 0.32 & 0 \end{array}\right] \end{array}. \tag{4.23}$$

4.3.2 "Closeness" of the Segmented and Original Graphs

The issue of how "close" the behavior of the weight matrix of the segmented graph, $\hat{\mathbf{W}}$, in (4.23) (and the corresponding $\hat{\mathbf{L}}$) is to the

original $\mathbf{W}$ and $\mathbf{L}$, in (2.4) and (2.8), is usually considered within matrix perturbation theory.

It can be shown that a good measure of the "closeness" is the so-called *eigenvalue gap*, $\delta = \lambda_2 - \lambda_1$, Ng *et al.* (2002), that is the difference between the eigenvalue λ_1 associated with the eigenvector $\mathbf{u}_1$, which is used for segmentation, and the next eigenvalue, λ_2, in the graph spectrum of the normalized graph Laplacian (for additional explanation see Example 23). For the obvious reason of analyzing the eigenvalue gap at an appropriate scale, we suggest to consider *the relative eigenvalue gap*

$$\delta_r = \frac{\lambda_2 - \lambda_1}{\lambda_2} = 1 - \frac{\lambda_1}{\lambda_2}. \tag{4.24}$$

The relative eigenvalue gap value range is within the interval $0 \leq \delta_r \leq 1$, since the eigenvalues are nonnegative real-valued numbers sorted into a nondecreasing order. The value of this gap may be considered as large if it is close to the maximum eigengap value, $\delta_r = 1$.

Example 20: The Laplacian eigenvalues for the graph in Figure 4.7 are $\lambda \in \{0, 0.29, 0.34, 0.79, 1.03, 1.31, 1.49, 2.21\}$, with the relative eigenvalue gap, $\delta_r = (\lambda_2 - \lambda_1)/\lambda_2 = 0.15$, which is not large and indicates that the segmentation in Example 19 is not "close".

As an illustration, consider three hypothetical but practically relevant scenarios: (i) $\lambda_2 = 0$ and $\lambda_3 = 1$, (ii) $\lambda_2 = 0$ and $\lambda_3 = \varepsilon$, (iii) $\lambda_2 = 1$ and $\lambda_3 = 1 + \varepsilon$, where ε is small positive number and close to 0. According to Remark 18, the graph in case (i) consists of exactly two disconnected components, and the subsequent clustering and segmentation is appropriate, with $\delta_r = 1$. For the case (ii), the graph consists of more than two almost disconnected components and the clustering in two sets can be performed in various ways, with $\delta_r = 1/\varepsilon$. Finally, in the last scenario the relative gap is very small, $\delta_r = \varepsilon$, thus indicating that the behavior of the segmented graph is not "close" to the original graph, that is, $\hat{\mathbf{L}}$ is not "close" to $\mathbf{L}$, and thus any segmentation into two disconnected subgraphs would produce inadequate results.

Remark 28: The thresholding of elements of the Fiedler vector, $\mathbf{u}_1$, of the normalized graph Laplacian, $\mathbf{L}_N = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$, performed in order to cluster the graph is referred to as the *Shi–Malik algorithm* (Shi and Malik, 2000; Weiss, 1999). Note that similar results would have been

obtained if clustering was based on the thresholding of elements of the smoothest eigenvector corresponding to the second largest eigenvalue of the normalized weight matrix, $\mathbf{W}_N = \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$ (*Perona–Freeman algorithm* (Perona and Freeman, 1998; Weiss, 1999)). This becomes clear after recalling that the relation between the normalized weight and graph Laplacian matrices is given by

$$\begin{aligned} \mathbf{L}_N &= \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}, \\ \mathbf{L}_N &= \mathbf{I} - \mathbf{W}_N. \end{aligned} \tag{4.25}$$

The eigenvalues of these two matrices are therefore related as $\lambda_k^{(L_N)} = 1 - \lambda_k^{(W_N)}$, while they share the same eigenvectors.

Clustering Based on More Than One Eigenvector

More complex clustering schemes can be achieved when using more than one Laplacian eigenvector. In turn, vertices with similar values of several slow-varying eigenvectors, $\mathbf{u}_k$, would exhibit high spectral similarity.

The concept of **using more than one eigenvector in vertex clustering** and possible subsequent graph segmentation was first introduced by Scott and Longuet-Higgins (1990). They used k eigenvectors of the weight matrix $\mathbf{W}$ to form a new $N \times k$ matrix $\mathbf{V}$, for which a further row normalization was performed. Vertex clustering is then performed based on the elements of the matrix $\mathbf{V}\mathbf{V}^T$.

For the normalized weight matrix, $\mathbf{W}_N$, the Scott and Longuet-Higgins algorithm reduces to the corresponding analysis with k eigenvectors of the normalized graph Laplacian, $\mathbf{L}_N$. Since $\mathbf{W}_N$ and $\mathbf{L}_N$ are related by (4.25), they thus have the same eigenvectors.

Example 21: Consider two independent ratio cuts of a graph, where the first cut splits the graph into the sets of vertices $\mathcal{E}_1$ and $\mathcal{H}_1$, and the second cut further splits all vertices into the sets $\mathcal{E}_2$ and $\mathcal{H}_2$, and define this two-level cut as

$$CutN2(\mathcal{E}_1, \mathcal{H}_1, \mathcal{E}_2, \mathcal{H}_2) = CutN(\mathcal{E}_1, \mathcal{H}_1) + CutN(\mathcal{E}_2, \mathcal{H}_2) \tag{4.26}$$

where both $CutN(\mathcal{E}_i, \mathcal{H}_i)$, $i = 1, 2$, are defined by (4.1).

If we now introduce two indicator vectors, $\mathbf{x}_1$ and $\mathbf{x}_2$, for the two respective cuts, then, from (4.12) we may write

$$CutN2(\mathcal{E}_1, \mathcal{H}_1, \mathcal{E}_2, \mathcal{H}_2) = \frac{\mathbf{x}_1^T \mathbf{L} \mathbf{x}_1}{\mathbf{x}_1^T \mathbf{x}_1} + \frac{\mathbf{x}_2^T \mathbf{L} \mathbf{x}_2}{\mathbf{x}_2^T \mathbf{x}_2}. \tag{4.27}$$

As mentioned earlier, finding the indicator vectors, $\mathbf{x}_1$ and $\mathbf{x}_2$, which minimize (4.27) is a combinatorial problem. However, if the space of solutions for the indicator vectors is now relaxed from the subset-wise constant form to the space spanned by the eigenvectors of the graph Laplacian, then the approximative minimum value of the two cuts, $CutN2(\mathcal{E}_1, \mathcal{H}_1, \mathcal{E}_2, \mathcal{H}_2)$, is obtained for $\mathbf{x}_1 = \mathbf{u}_1$ and $\mathbf{x}_2 = \mathbf{u}_2$, since $\mathbf{u}_1$ and $\mathbf{u}_2$ are maximally smooth but not constant (for the proof see (4.31)–(4.32) and for the illustration see Example 22).

For the case of two independent cuts, for convenience, we may form the indicator $N \times 2$ matrix $\mathbf{Y} = [\mathbf{x}_1, \mathbf{x}_2]$, so that the corresponding matrix of the solution (within the graph Laplacian eigenvectors space) to the two ratio cuts minimization problem, has the form

$$\mathbf{Q} = [\mathbf{u}_1, \mathbf{u}_2].$$

The rows of this matrix, $\mathbf{q}_n = [u_1(n), u_2(n)]$, are the spectral vectors which are assigned to each vertex, n.

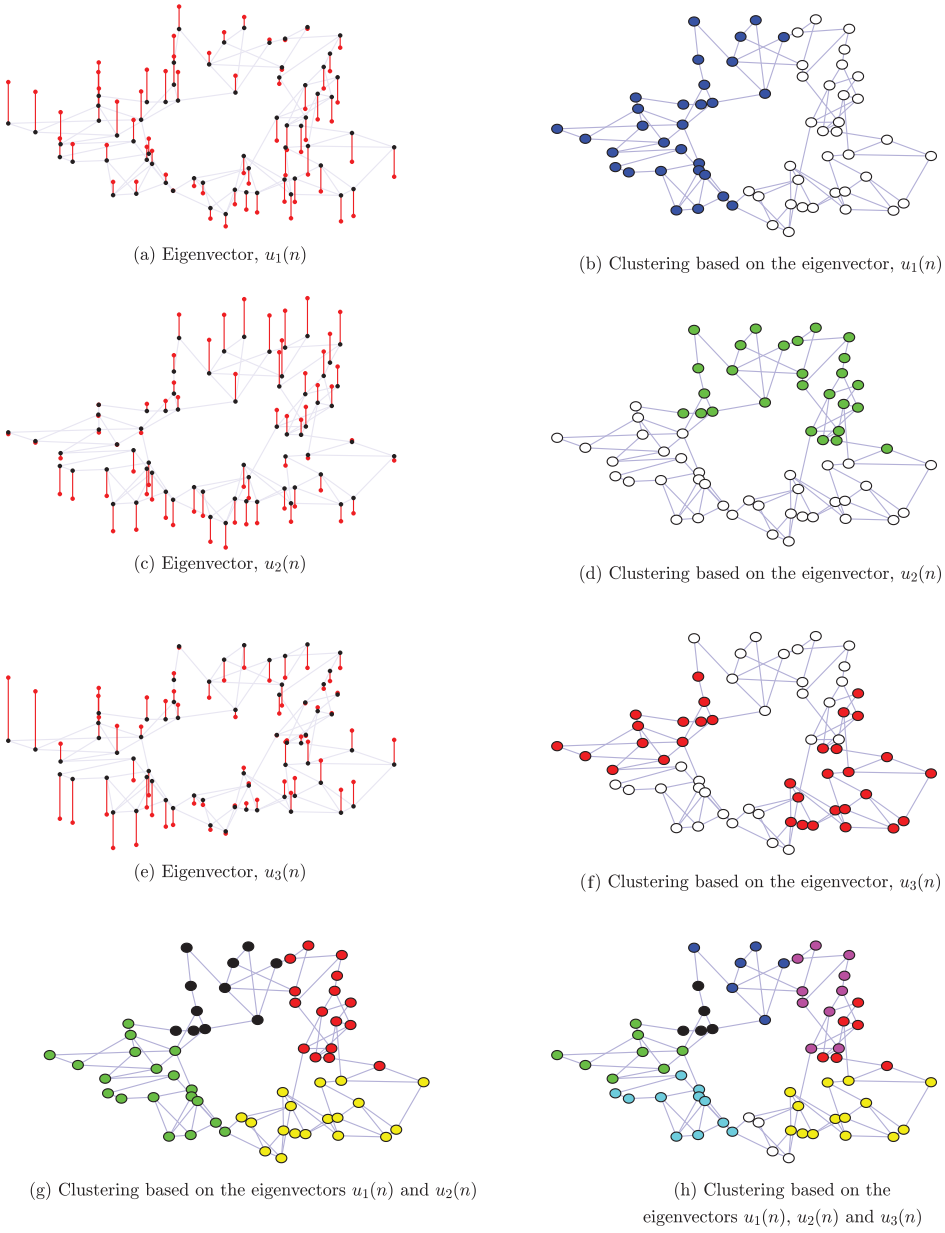
The same reasoning can be followed for the cases of three or more independent cuts, to obtain an $N \times M$ indicator matrix $\mathbf{Y} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M]$ with the corresponding eigenvector approximation, $\mathbf{Q}$, the rows of which are the spectral vectors $\mathbf{q}_n = [u_1(n), u_2(n), \ldots, u_M(n)]$.

Remark 29: *Graph clustering* in the spectral domain may be performed by assigning the spectral vector,

$$\mathbf{q}_n = [u_1(n), \ldots, u_M(n)]$$

in (4.9), to each vertex, n, and subsequently by grouping the vertices with similar spectral vectors into the corresponding clusters (Belkin and Niyogi, 2003; Ng *et al.*, 2002).

Low dimensional spectral vectors (up to $M = 3$) can be represented by color coordinates of, for example, standard RGB coloring system. To this



(a) Eigenvector, $u_1(n)$

(b) Clustering based on the eigenvector, $u_1(n)$

(c) Eigenvector, $u_2(n)$

(d) Clustering based on the eigenvector, $u_2(n)$

(e) Eigenvector, $u_3(n)$

(f) Clustering based on the eigenvector, $u_3(n)$

(g) Clustering based on the eigenvectors $u_1(n)$ and $u_2(n)$

(h) Clustering based on the eigenvectors $u_1(n)$, $u_2(n)$ and $u_3(n)$

Figure 4.9: Spectral vertex clustering schemes for the graph from Figure 4.4. (a) The eigenvector, $\mathbf{u}_1$, of the Laplacian matrix (plotted in red lines on vertices designated by black dots) is first normalized and is then used to designate (b) a two-level blue colormap intensity (through its signs) for every vertex (blue-white circles). (c) The eigenvector, $\mathbf{u}_2$, of the Laplacian matrix is normalized and is then used to provide (d) a two-level green colormap intensity for every vertex. (e) The eigenvector, $\mathbf{u}_3$, of the Laplacian matrix is normalized and used as (f) a two-level red colormap intensity for every vertex. (g) Clustering based on the combination of the eigenvectors $\mathbf{u}_1$ and $\mathbf{u}_2$. (h) Clustering based on the combination of the eigenvectors $\mathbf{u}_1$, $\mathbf{u}_2$, and $\mathbf{u}_3$. Observe an increase in degrees of freedom with the number of eigenvectors used; this is reflected in the number of detected clusters, starting from two clusters in (b) and (d), via four clusters in (g), to 8 clusters in (h).

end, it is common to use different vertex colors, which represent different spectral vectors, for the visualization of spectral domain clustering.

Example 22: Figure 4.9 illustrates several spectral vector clustering schemes for the graph in Figure 4.4 (right), based on the three smoothest eigenvectors $\mathbf{u}_1$, $\mathbf{u}_2$, and $\mathbf{u}_3$. Clustering based on the eigenvector $\mathbf{u}_1$, with $\mathbf{q}_n = [u_1(n)]$, is shown in Figure 4.9(b), clustering using the eigenvector $\mathbf{u}_2$ only, with $\mathbf{q}_n = [u_2(n)]$, is shown in Figure 4.9(d), while Figure 4.9(e) illustrates the clustering based on the eigenvectors $\mathbf{u}_3$, when $\mathbf{q}_n = [u_3(n)]$. Clustering based on the combination of the two smoothest eigenvectors $\mathbf{u}_1$, and $\mathbf{u}_2$, with spectral vectors $\mathbf{q}_n = [u_1(n), u_2(n)]$, is shown in Figure 4.9(g), while Figure 4.9(h) illustrates clustering based on the three smoothest eigenvectors, $\mathbf{u}_1$, $\mathbf{u}_2$, and $\mathbf{u}_3$, whereby the spectral vector $\mathbf{q}_n = [u_1(n), u_2(n), u_3(n)]$. In all cases, two-level colormaps were used for each eigenvector. The smallest eigenvalues were $\lambda_0 = 0$, $\lambda_1 = 0.0286$, $\lambda_2 = 0.0358$, $\lambda_3 = 0.0899$, $\lambda_4 = 0.104$, and $\lambda_5 = 0.167$, so that the largest relative gap was obtained when $\mathbf{u}_1$ and $\mathbf{u}_2$ were used for clustering, with the corresponding eigenvalue gap of $\delta_r = 1 - \lambda_2/\lambda_3 = 0.6$.

Remark 30: ***k*-means algorithm.** The above clustering schemes are based on the quantized levels of spectral vectors. These can be refined using *the k-means algorithm*, that is, through postprocessing in the form of unsupervised learning and in the following way.

(i) After an initial vertex clustering is performed by grouping the vertices into $\mathcal{V}_i$, $i = 1, 2, \ldots, k$ nonoverlapping vertex subsets, a new spectral vector centroid, $\mathbf{c}_i$, is calculated as

$$\mathbf{c}_i = \text{mean}_{n \in \mathcal{V}_i}\{\mathbf{q}_n\},$$

for each cluster of vertices $\mathcal{V}_i$.

(ii) Every vertex, n, is then reassigned to its nearest (most similar) spectral domain centroid, i, where the spectral distance (spectral similarity) is calculated as $\|\mathbf{q}_n - \mathbf{c}_i\|_2$.

This two-step algorithm is iterated until no vertex changes clusters. Finally, all vertices in one cluster are colored based on the corresponding common spectral vector $\mathbf{c}_i$ (or visually, a color representing $\mathbf{c}_i$).

Clustering refinement using the k-means algorithm is illustrated later in Example 29.

Example 23: Graphs represent quite a general mathematical formalism, and we will here provide only one possible physical interpretation of graph clustering. Assume that each vertex represents one out of the set of N images, which exhibit both common elements and individual differences. If the edge weights are calculated so as to represent mutual similarities between these images, then spectral vertex analysis can be interpreted as follows. If the set is complete and with very high similarity among all vertices, then $W_{mn} = 1$, and $\lambda_0 = 0, \lambda_1 = N, \lambda_2 = N, \ldots, \lambda_{N-1} = N$, as shown in Remark 19. The relative eigenvalue gap is then $\delta_r = (\lambda_2 - \lambda_1)/\lambda_2 = 0$ and the segmentation is not possible.

Assume now that the considered set of images consists of two connected subsets with the respective numbers of N_1 and $N_2 \geq N_1$ of very similar photos within each subset. In this case, the graph consists of two complete components (sub-graphs). According to Remarks 18 and 19, the graph Laplacian eigenvalues are now $\lambda_0 = 0, \lambda_1 = 0, \lambda_2 = N_1, \ldots, \lambda_{N_1} = N_1, \lambda_{N_1+1} = N_2, \ldots, \lambda_{N-1} = N_2$. Then, this graph may be well segmented into two components (sub-graphs) since the relative eigenvalue gap is now large, $\delta_r = (\lambda_2 - \lambda_1)/\lambda_2 = 1$. Therefore, this case can be used for *collaborative data processing* within each of these subsets. The analysis can be continued and further refined for cases with more than one eigenvector and more than two subsets of vertices. Note that segmentation represents a "hard-thresholding" operation of cutting the connections between vertices in different subsets, while clustering refers to just a grouping of vertices, which exhibit some similarity, into subsets, while keeping their mutual connections.

Example 24: For enhanced intuition, we next consider a real-world dataset with eight images, shown in Figure 4.10. The connectivity weights were calculated using the structural similarity index (SSIM), with an appropriate threshold (Wang *et al.*, 2003). The so obtained

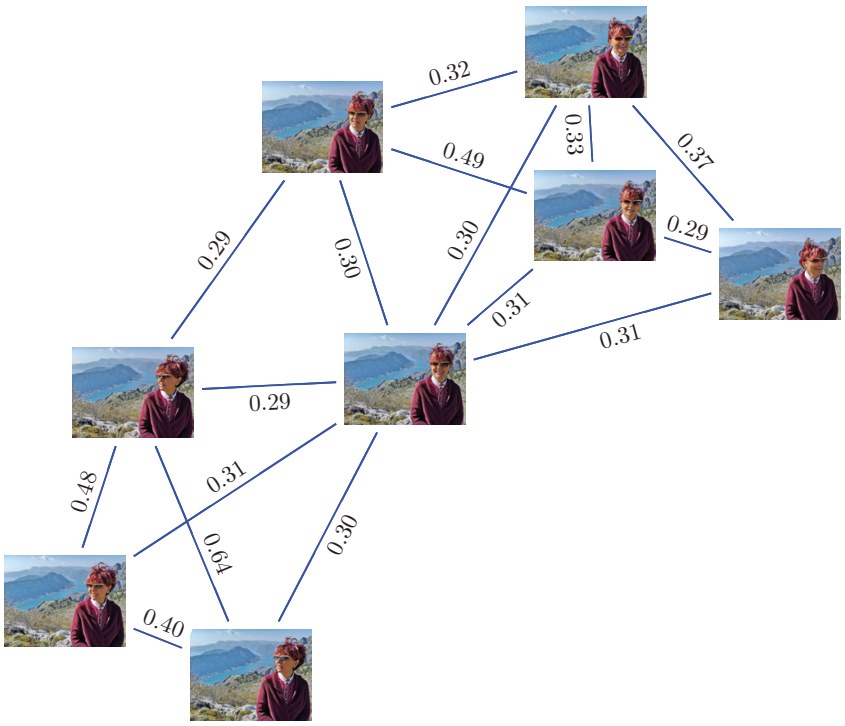


Figure 4.10: A graph representation of a set of the real-world images which exhibit an almost constant background but different head orientation, which moves gradually from the left profile (bottom left) to the right profile (top right). The images serve as vertices, while the edges and the corresponding weight matrix are defined through the squared structural similarity index (SSIM) between images, with $W_{mn} = \text{SSIM}_T^2(m,n)$, and hard thresholded at 0.28 to account for the contribution of the background to the similarity index, that is, $\text{SSIM}_T(m,n) = \text{hard}(\text{SSIM}(m,n), 0.53)$.

weight matrix, $\mathbf{W}$, is given by

$$\mathbf{W} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{cccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{bmatrix} 0 & 0.49 & 0.33 & 0.29 & 0.31 & 0 & 0 & 0 \\ 0.49 & 0 & 0.32 & 0 & 0.30 & 0 & 0 & 0.29 \\ 0.33 & 0.32 & 0 & 0.37 & 0.30 & 0 & 0 & 0 \\ 0.29 & 0 & 0.37 & 0 & 0.31 & 0 & 0 & 0 \\ 0.31 & 0.30 & 0.30 & 0.31 & 0 & 0.31 & 0.30 & 0.29 \\ 0 & 0 & 0 & 0 & 0.31 & 0 & 0.40 & 0.48 \\ 0 & 0 & 0 & 0 & 0.30 & 0.40 & 0 & 0.64 \\ 0 & 0.29 & 0 & 0 & 0.29 & 0.48 & 0.64 & 0 \end{bmatrix} \end{array}. \tag{4.28}$$

The standard graph form for this real-world scenario in Figure 4.10 is shown in Figure 4.11, together with the corresponding image/vertex indexing. Notice the almost constant background in all eight images

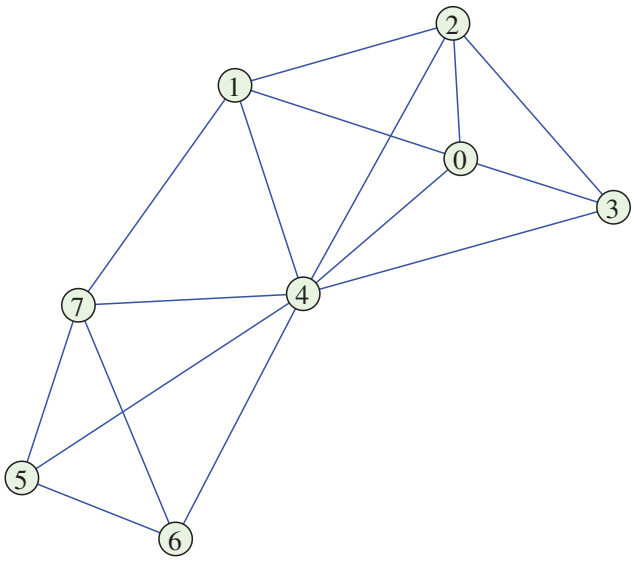


Figure 4.11: Graph topology for the real-world images from Figure 4.10.

(the photos were taken in the wild by a "hand-held device"), and that the only differences between the images are in that the model gradually moved her head position from the left profile (bottom left) to the right profile (top right). Therefore, the two frontal face positions, at vertices $n = 4$ and $n = 0$, exhibit higher vertex degrees than the other head orientations, which exemplifies physical meaningfulness of graph representations. The normalized spectral vectors for this graph, $\mathbf{q}_n = [u_1(n)]/\|[u_1(n)]\|_2$ and $\mathbf{q}_n = [u_1(n), u_2(n)]/\|[u_1(n), u_2(n)]\|_2$ were obtained as the generalized eigenvectors of the graph Laplacian, and were used to define the coloring scheme for the graph clustering in Figure 4.12. Recall that similar vertex colors indicate spectral similarity of the images assigned to the corresponding vertices.

The eigenvalues of the graph Laplacian for this example are $\lambda_k \in \{0, 0.42, 1.12, 1.63, 1.68, 1.89, 2.31, 2.42\}$. The largest relative eigenvalue gap is therefore between the eigenvalues $\lambda_1 = 0.42$ and $\lambda_2 = 1.12$, and indicates that the best clustering will be obtained in a one-dimensional spectral space (with clusters shown in Figure 4.12(a)). However, the value of such cut would be large, $Cut(\{0, 1, 2, 3, 4\}, \{5, 6, 7\}) = 1.19$, while the value of the ratio cut,

$$CutN(\{0, 1, 2, 3, 4\}, \{5, 6, 7\}) = 0.63 \sim \lambda_1 = 0.42,$$

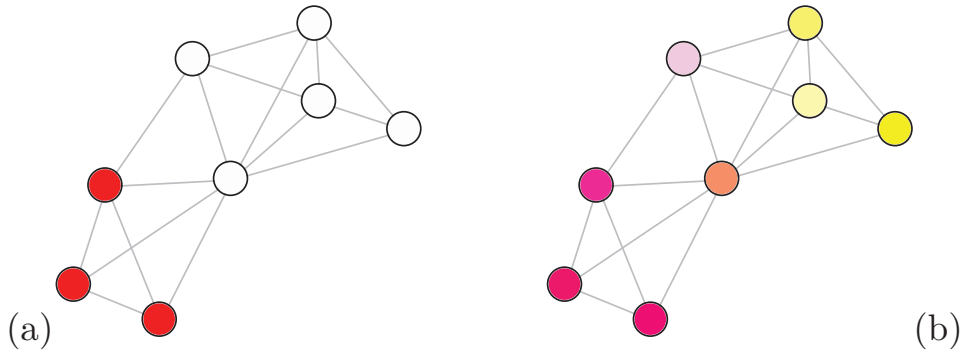


Figure 4.12: Graph clustering structure for the images from Figure 4.10. (a) Vertices are clustered (colored) using the row-normalized spectral Fiedler eigenvector to give the spectral vector $\mathbf{u}_1$, $\mathbf{q}_n = [u_1(n)]/\|[u_1(n)]\|_2$. (b) Clustering scheme whereby spectral values of vertices are calculated using the two smoothest eigenvectors, $\mathbf{q}_n = [u_1(n), u_2(n)]$, which are then employed to designate the colormap for the vertices. Recall that the so obtained similar vertex colors indicate spectral similarity of the images from Figure 4.10.

indicates that the connections between these two clusters are too significant for a segmented graph to produce a "close" approximation of the original graph with only two components (disconnected subgraphs). Given the gradual change in head orientation, this again conforms with physical intuition, and the subsequent clustering based on two smoothest eigenvectors, $\mathbf{u}_1$ and $\mathbf{u}_2$, yields three meaningful clusters of vertices corresponding to the "left head orientation" (red), "frontal head orientation" (two shades of pink), and "right head orientation" (yellow).

Example 25: **Minnesota roadmap graph.** Three eigenvectors of the graph Laplacian matrix, $\mathbf{u}_2$, $\mathbf{u}_3$, and $\mathbf{u}_4$, were used as the coloring templates to represent the spectral similarity and clustering in the benchmark *Minnesota roadmap graph*, shown in Figure 4.13. The eigenvectors $\mathbf{u}_0$ and $\mathbf{u}_1$ were omitted, since their corresponding eigenvalues are $\lambda_0 = \lambda_1 = 0$ (due to an isolated vertex in the graph data which behaves as a graph component, see Remark 18). The full (nonquantized) colormap scale was used to color the vertices (that is, represent three-dimensional spectral vectors). As elaborated above, regions where the vertices visually assume similar colors are also spectrally similar, and with similar behavior of the corresponding slow-varying eigenvectors.

Figure 4.13: Vertex coloring in the benchmark Minnesota road-map graph using the three smoothest Laplacian eigenvectors $\{\mathbf{u}_2, \mathbf{u}_3, \mathbf{u}_4\}$, as coordinates in the standard RGB coloring system (a three-dimensional spectral space with the spectral vector $\mathbf{q}_n = [u_2(n), u_3(n), u_4(n)]$ for every vertex, n). The vertices with similar colors are therefore also considered spectrally similar. Observe three different clusters, characterized by the shades of predominantly red, green, and blue color, that correspond to intensities defined by the eigenvectors $u_2(n), u_3(n)$, and $u_4(n)$.

Example 26: **Brain connectivity graph.** Figure 4.14 shows the benchmark Brain Atlas connectivity graph (Mijalkov *et al.*, 2017, Rubinov and Sporns, 2010), for which the data is given in two matrices: "Coactivation matrix", $\hat{\mathbf{W}}$, and "Coordinate matrix". The "Coordinate matrix" contains the vertex coordinates in a three-dimensional Euclidean space, whereby the coordinate of a vertex n is defined by the n-th row of the "Coordinate matrix", that is, $[x_n, y_n, z_n]$.

In our analysis, the graph weight matrix, $\mathbf{W}$, was empirically formed by:

(i) thresholding the "Coactivation matrix", $\hat{\mathbf{W}}$, to preserve only the strongest connections within this brain atlas, for example, those greater than $0.1 \max\{\hat{W}_{mn}\}$, as recommended in Rubinov and Sporns (2010);

(ii) only the edges between the vertices m and n, whose Euclidean distance satisfies $d_{mn} \leq 20$ are kept in the graph representation.

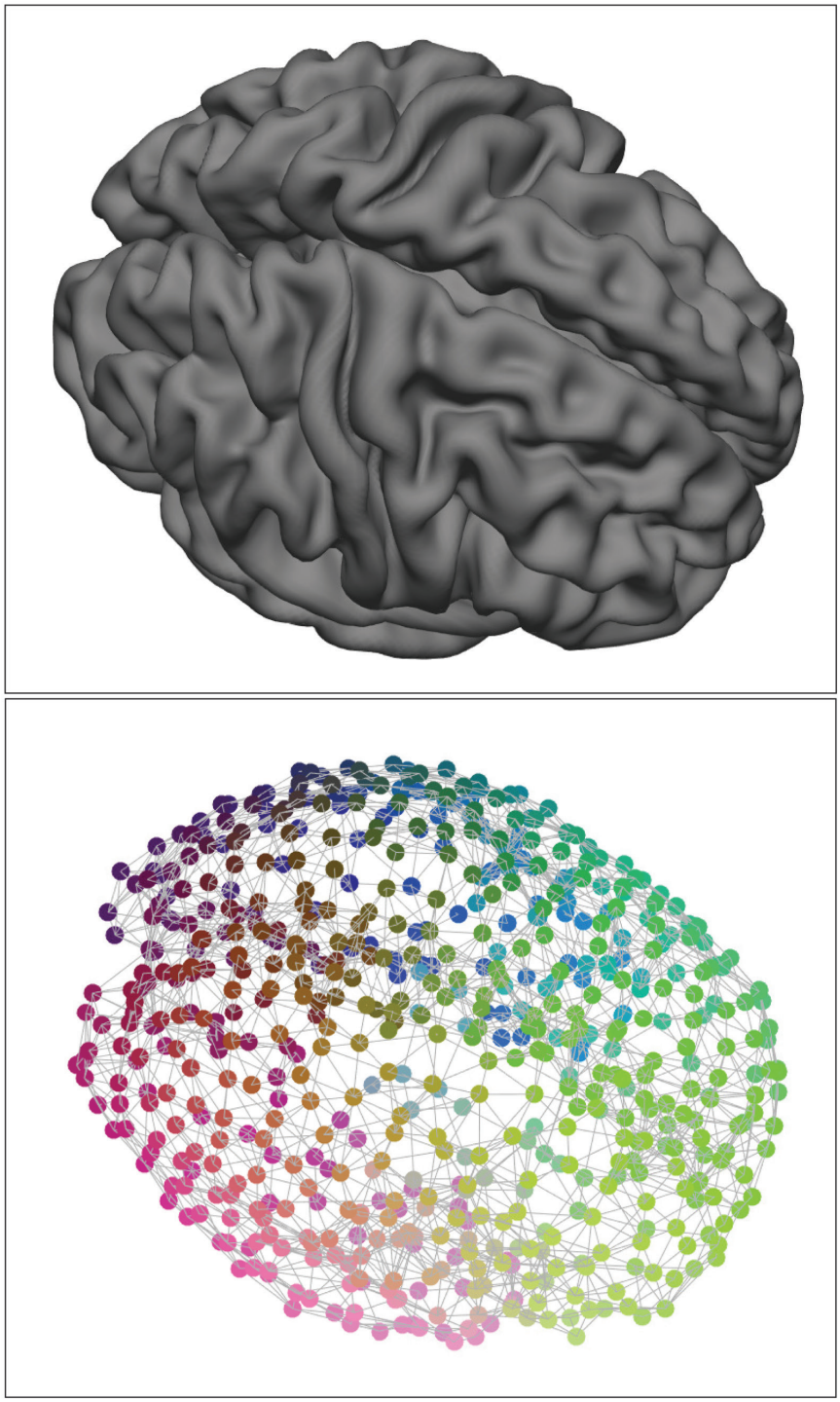


Figure 4.14: Brain atlas (top) and its graph (bottom), with vertex coloring based on the three smoothest generalized eigenvectors, $\mathbf{u}_1$, $\mathbf{u}_2$, and $\mathbf{u}_3$, of graph Laplacian. The spectral vector, $\mathbf{q}_n = [u_1(n), u_2(n), u_3(n)]$ is employed as the coordinates in the RGB coloring scheme (Mijalkov *et al.*, 2017; Rubinov and Sporns, 2010).

The elements, W_{mn}, of the brain graph weight matrix, $\mathbf{W}$, are therefore obtained from the corresponding elements, $\hat{W}_{mn}$, of the "Coactivation matrix" as

$$W_{mn} = \begin{cases} \hat{W}_{mn}, & \text{if } \hat{W}_{mn} > 0.1 \max\{\hat{W}_{mn}\} \text{ and } d_{mn} \leq 20 \\ 0, & \text{elsewhere.} \end{cases} \tag{4.29}$$

The brain connectivity graph with the so defined weight matrix, $\mathbf{W}$, is shown in Figure 4.14(bottom).

The three smoothest generalized eigenvectors, $\mathbf{u}_1$, $\mathbf{u}_2$ and $\mathbf{u}_3$, of the corresponding graph Laplacian matrix, $\mathbf{L} = \mathbf{D} - \mathbf{W}$, were next used to define the spectral vectors

$$\mathbf{q}_n = [u_1(n), u_2(n), u_3(n)]$$

for each vertex, $n = 0, 1, \ldots, N-1$. The elements of this spectral vector, $\mathbf{q}_n$, were then used to designate the corresponding RGB coordinates for the coloring of the vertices of the brain graph, as shown in Figure 4.14.

4.4 Vertex Dimensionality Reduction Using the Laplacian Eigenmaps

We have seen that graph clustering can be used for collaborative processing on the set of data which is represented by the vertices within a cluster. In general, *any form of the presentation of a graph and its corresponding vertices, that employs the eigenvectors of the graph Laplacian may be considered as a Laplacian eigenmap*. The idea which underpins eigenmap-based approaches presented here is to employ spectral vectors, $\mathbf{q}_n$, to define the new positions of the original vertices in such a "transform-domain" space so that spectrally similar vertices appear spatially closer than in the original vertex space.

Remark 31: The Laplacian eigenmaps may be employed for *vertex dimensionality reduction*, while at the same time preserving the local properties and natural connections within the original graph (Belkin and Niyogi, 2003).

Consider a vertex n, $n = 0, 1, \ldots, N-1$, which resides in an L-dimensional space $\mathbb{R}^L$, at the position defined by the L-dimensional

vector $\mathbf{r}_n$. A spectral vector for vertex n is then defined in a new lower-dimensional (M-dimensional) space, with $M < N$, by keeping the M smoothest eigenvectors of graph Laplacian, $\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_M$. Upon omitting the constant eigenvector, $\mathbf{u}_0$, this gives the new basis designated by the spectral vector

$$\mathbf{q}_n = [u_1(n), \ldots, u_M(n)]. \tag{4.30}$$

Since $M < L$, this provides the desired dimensionality reduction of the vertex space. The concepts of spectral vector-based vertex dimensionality reduction, and physical meaning associated with the spectral vector space representation are illustrated in the next example.

Example 27: **Vertex dimensionality reduction.** Consider a set of $N = 70$ students and their marks in 40 lecture courses. Every student can be considered as a vertex located in the original $L = 40$ dimensional space at the position $\mathbf{r}_n$, where $r_n(k)$ is a mark for the n-th student at k-th course. Assume that the marks are within the set $\{2, 3, 4, 5\}$ and that some students have affinity to certain subsets of courses (for example, social sciences, natural sciences and skills). This set-up can be represented in a tabular (70×40) compact matrix form as in Figure 4.15(a), where the columns contain the marks for every student (the marks are color coded).

The average marks per student and per course are shown in Figures 4.15(b) and (c). Observe the limitations of this representation, as for example, the average marks cannot be used to determine student affinities to the subsets of their courses.

We can now create a graph representation by connecting with edges students with similar marks. In our example, the edge weights were determined through a distance in the 40-dimensional feature (marks) space, as

$$W_{mn} = \begin{cases} e^{-\|\mathbf{r}_m - \mathbf{r}_n\|_2^2/70}, & \text{for } \|\mathbf{r}_m - \mathbf{r}_n\|_2 \geq 7 \\ 0, & \text{otherwise.} \end{cases}$$

With the so obtained connectivity, this graph is presented in Figure 4.15(d), whereby the vertices (students) are randomly positioned in a plane and connected with edges. We shall now calculate the

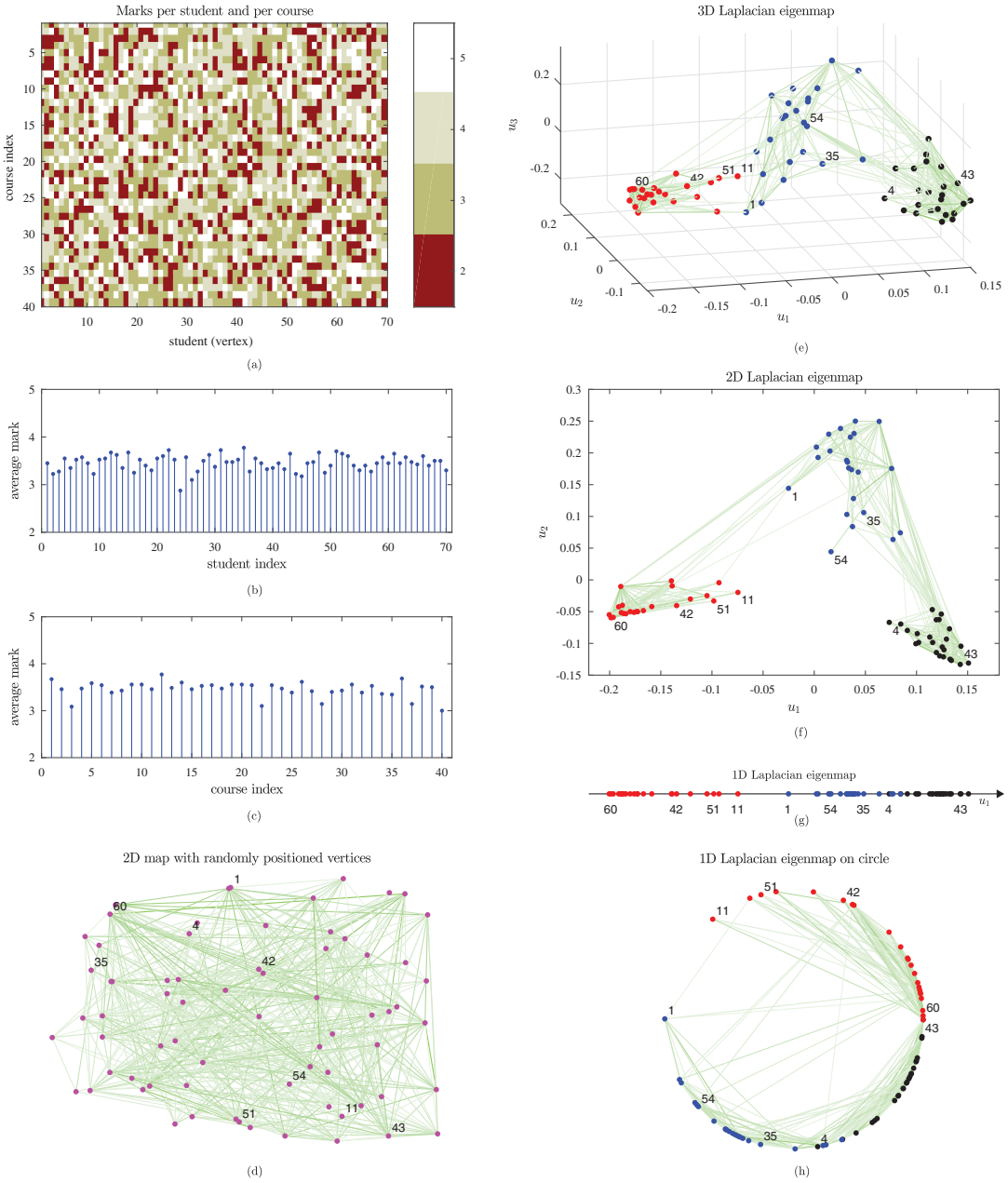


Figure 4.15: Illustration of spectral dimensionality reduction through an example of exam marks for a cohort of students. (a) Each of the 70 columns (students) represents a 40-dimensional vector with student marks. Therefore the dimensionality of the original representation space is $L = 40$. (b) Average mark per student. (c) Average mark per course. (d) Two-dimensional graph representation of the matrix in (a), where the individual students are represented by randomly positioned vertices in the plane. To perform vertex (student) dimensionality reduction we can use spectral vectors to reduce their original $L = 40$ dimensional representation space to (e) $M = 3$, (f) $M = 2$, and (g) $M = 1$ dimensional spectral representation spaces. (h) Vertices from path graph (g) positioned on a circle (by connecting the ends of the line) which allows us to also show the edges.

normalized Laplacian eigenvectors and remap the vertices according to the three-dimensional, two-dimensional and one-dimensional spectral vectors, $\mathbf{q}_n$, defined by (4.30) that is, for $M = 3$, $M = 2$, and $M = 1$. In this way, the vertex dimensionality is reduced from the original $L = 40$ to a much lower $M \ll L$. The corresponding graph representations are respectively shown in Figures 4.15(e)–(g). For $M = 2$ and $M = 3$ we can now clearly divide students into the three affinity groups (designated by the red, blue, and black). Although the obtained groups (clusters) are logically ordered even in the one-dimensional case in Figure 4.15(g), observe that we cannot use $M = 1$ for precise grouping since there is no enough gap between the groups. However, even in this case, if we re-cast the vertices on a circle instead on a line (by connecting two ends of a line), and draw the connecting edges (the same edges as in Figures 4.15(d)–(f)) we can see the benefit of a graph representation even after such a radical dimensionality reduction.

The dimensionality reduction principle can also be demonstrated based on Example 24, whereby each vertex is a 640×480 RGB color image which can be represented as a vector in the $L = 640 \times 480 \times 3 = 921600$ dimensional space. Indeed, using spectral vectors with $M = 2$, this graph can be presented in a two-dimensional space as in Figure 4.10.

Within the Laplacian eigenmaps method, we may use any of the three forms of graph Laplacian eigenvectors introduced in Section 4.2.3. The relations among these three presentations are explained in Section 4.2.3 and Table 4.1. A unified algorithm for all three variants of the Laplacian eigenmaps, and the corresponding clustering methods, is given in Algorithm 3 in the appendix.

Remark 32: The Laplacian eigenmaps are optimal in the sense that they minimize an objective function which penalizes for the distance between the neighboring vertices in the spectral space. This ensures that if the vertices at the positions $\mathbf{r}_m$ and $\mathbf{r}_n$ in the original high-dimensional L-dimensional space are *"close" in the sense of some data association metric*, then they will also be *close in the Euclidean sense* in the reduced M-dimensional spectral space, where their positions are defined by the corresponding spectral vectors, $\mathbf{q}_m$ and $\mathbf{q}_n$.

4.4.1 Euclidean Distances in the Space of Spectral Vectors

We shall prove the "distance preserving" property of the above spectral mapping in an inductive way. Assume that a graph is connected, i.e., $\lambda_1 \neq 0$. The derivation is based on the quadratic form in (4.8)

$$\mathbf{u}_k^T \mathbf{L} \mathbf{u}_k = \frac{1}{2} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} (u_k(m) - u_k(n))^2 W_{mn}$$

which states that $\mathbf{u}_k^T \mathbf{L} \mathbf{u}_k$ is equal to the weighted sum of squared Euclidean distances between the elements of the m-th and n-th eigenvector at vertices m and n, for all m and n. Recall that $\mathbf{u}_k^T \mathbf{L} \mathbf{u}_k$ is also equal to λ_k, by definition (see the elaboration after (4.6)).

Single-dimensional case. To reduce the original L-dimensional vertex space to a single-dimensional path graph with vertex coordinates $\mathbf{q}_n = u_k(n)$, the minimum sum of the weighted squared distances between the vertices m and n, that is

$$\begin{aligned} &\frac{1}{2} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} \|\mathbf{q}(m) - \mathbf{q}(n)\|_2^2 W_{mn} \\ &\quad = \frac{1}{2} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} (u_k(m) - u_k(n))^2 W_{mn} = \lambda_k \end{aligned}$$

will be obtained with the new positions of vertices, designated by $\mathbf{q}_n = [u_1(n)]$, and for $k = 1$, since $\min_{k, \lambda_k \neq 0}\{\lambda_k\} = \lambda_1$ is the smallest nonzero eigenvalue.

Two-dimensional case. If we desire to reduce the original L-dimensional vertex representation space to a two-dimensional spectral space, designated by $\mathbf{q}_n = [u_k(n), u_l(n)]$ and defined through any two eigenvectors of the graph Laplacian, $\mathbf{u}_k$ and $\mathbf{u}_l$, then the minimum sum of the weighted squared distances between all vertices, m and n, given by

$$\begin{aligned} &\frac{1}{2} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} \|\mathbf{q}_m - \mathbf{q}_n\|_2^2 W_{mn} \\ &\quad = \frac{1}{2} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} (u_k(m) - u_k(n))^2 W_{mn} \end{aligned}$$

$$+\frac{1}{2}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1}(u_l(m)-u_l(n))^2 W_{mn}$$
$$=\mathbf{u}_k^T\mathbf{L}\mathbf{u}_k+\mathbf{u}_l^T\mathbf{L}\mathbf{u}_l=\lambda_k+\lambda_l \tag{4.31}$$

will be obtained with the new spectral positions, $\mathbf{q}_n=[u_k(n),u_l(n)]$, such that $\mathbf{q}_n=[u_1(n),u_2(n)]$, since

$$\min_{k,l,k\neq l,kl\neq 0}\{\lambda_k+\lambda_l\}=\lambda_1+\lambda_2 \tag{4.32}$$

for nonzero k and l, and keeping in mind that $\lambda_1\leq\lambda_2\leq\lambda_3\leq\cdots\leq\lambda_{N-1}$. The same reasoning holds for new three- and higher-dimensional spectral representation spaces for the vertices, which yields (4.30) as the optimal vertex positions in the reduced M-dimensional vertex space.

The same relations hold for both the generalized eigenvectors of the Laplacian, defined by $\mathbf{L}\mathbf{u}_k=\lambda_k\mathbf{D}\mathbf{u}_k$, and the eigenvectors of the normalized Laplacian, defined by $\mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}\mathbf{v}_k=\lambda_k\mathbf{v}_k$. The only difference is in their respective normalization conditions, $\mathbf{u}_k^T\mathbf{D}\mathbf{u}_k$ and $\mathbf{v}_k^T\mathbf{v}_k$. The relation between the eigenvectors of the normalized graph Laplacian, $\mathbf{v}_k$, and the generalized eigenvectors of the graph Laplacian, $\mathbf{u}_k$, in the form $\mathbf{u}_k=\mathbf{D}^{-1/2}\mathbf{v}_k$, follows from their definitions (see Remark 27). Since the elements $u_1(n)$ and $u_2(n)$ are obtained by multiplying the elements $v_1(n)$ and $v_2(n)$ by the same value, $1/D_{nn}$, that is, $[u_1(n),u_2(n)]=[v_1(n),v_2(n)]/D_{nn}$, their normalized forms of $\mathbf{u}_k$ and $\mathbf{v}_k$ are identical,

$$\frac{\mathbf{q}_n}{\|\mathbf{q}_n\|_2}=\frac{[u_1(n),u_2(n)]}{\|[u_1(n),u_2(n)]\|_2}=\frac{[v_1(n),v_2(n)]}{\|[v_1(n),v_2(n)]\|_2}.$$

4.4.2 Examples of Graph Analysis in the Spectral Space

Example 28: The graph from Figure 2.2, where the vertices reside in a two-dimensional plane, is shown in Figure 4.16(a), while Figure 4.16(b) illustrates the same graph but represented in a reduced single-dimensional vertex space (a line). The vertex positions on the line are defined by the spectral vector, $\mathbf{q}_n=[u_1(n)]$, with $\mathbf{u}_1=[0.42,0.38,0.35,0.15,-0.088,-0.34,-0.35,-0.54]^T$.

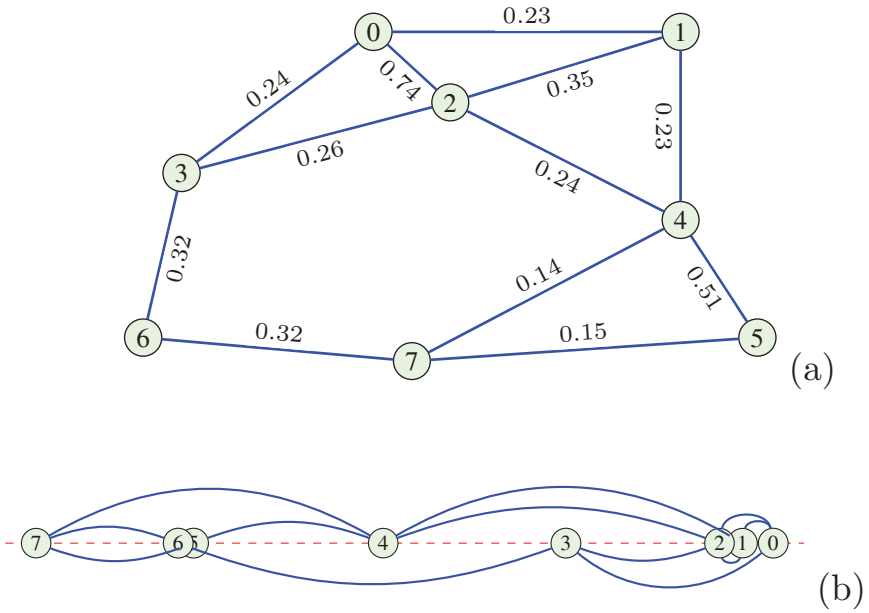


Figure 4.16: Principle of vertex dimensionality reduction based on the spectral vectors. (a) The weighted graph from Figure 2.2 with its vertices in a two-dimensional space. (b) The graph from (a) with its vertices located along a line (one-dimensional vertex space), whereby the positions on the line are defined by the one-dimensional spectral vector, $\mathbf{q}_n = [u_1(n)]$, with $\mathbf{u}_1 = [0.42, 0.38, 0.35, 0.15, -0.088, -0.34, -0.35, -0.54]^T$. Observe that this dimensionality reduction method may be used for clustering, based on the vertex position on the line.

Remark 33: After the vertices are reordered according to the Fiedler eigenvector, $\mathbf{u}_1$, Example 28 indicates the possibility of clustering refinement through a recalculation of ratio cuts. For the set of vertices $\mathcal{V} = \{0, 1, 2, \ldots, N-1\}$, Figure 4.16(b) illustrates their ordering along a line, with the new order $\{v_1, v_2, \ldots, v_N\} = \{7, 6, 5, 4, 3, 2, 1, 0\}$. Instead of using the sign of $\mathbf{u}_1$ to cluster the vertices, we can recalculate the ratio cuts, $CutN(\mathcal{E}_p, \mathcal{H}_p)$, with this sequential vertex order, where $\mathcal{E}_p = \{v_1, v_2, \ldots, v_p\}$ and $\mathcal{H}_p = \{v_{p+1}, v_{p+2}, \ldots, v_N\}$, for $p = 1, 2, \ldots, N-1$. The estimation of the minimum ratio cut then becomes

$$(\mathcal{E}_p, \mathcal{H}_p) = \arg\min_p \{CutN(\mathcal{E}_p, \mathcal{H}_p)\}.$$

This approximation of the Cheeger's cut can also be written using the thresholding of the eigenvector $\mathbf{u}_1$, by a threshold t, as

$$\phi^*(\mathcal{V}) = \min_t \left\{ \frac{1}{\min\{N_{\mathcal{E}_t}, N_{\mathcal{V}-\mathcal{E}_t}\}} \sum_{\substack{m\in\mathcal{E}_t \\ n\in\mathcal{V}-\mathcal{E}_t}} W_{mn} \right\}, \tag{4.33}$$

where the vertex n belongs to $\mathcal{E}_t$ if $u_1(n) > t$.

This method is computationally efficient since only $(N-1)$ cuts, $CutN(\mathcal{E}_p, \mathcal{H}_p)$, need to be calculated. In addition, the cuts $CutN(\mathcal{E}_p, \mathcal{H}_p)$ can be calculated recursively, using the previous $CutN(\mathcal{E}_{p-1}, \mathcal{H}_{p-1})$ and the connectivity parameters (degree, D_{pp}, and weights, W_{pm}) of vertex p. Any normalized cut form presented in Section 4.1 can also be used instead of $CutN(\mathcal{E}_p, \mathcal{H}_p)$. When the Cheeger ratio, defined in (4.5), is used in this minimization, then an upper bound on the cut can be obtained as Trevisan (2013)

$$\min_p\{\phi(\mathcal{E}_p)\} \leq \sqrt{2\lambda_1} \leq 2\sqrt{\phi(\mathcal{V})}, \tag{4.34}$$

where $\phi(\mathcal{V})$ denotes the combinatorial (true) minimum cut, with bounds given in (4.18).

Example 29: We shall now revisit the graph in Figure 4.9 and examine the clustering schemes based on: (i) standard Laplacian eigenvectors (Figure 4.17), (ii) generalized eigenvectors of graph Laplacian (Figure 4.18), and (iii) eigenvectors of the normalized Laplacian (Figure 4.19). Figure 4.17(b) illustrates Laplacian eigenmaps based dimensionality reduction for the graph from Figure 4.9(g), with the two eigenvectors, $\mathbf{u}_1$ and $\mathbf{u}_2$, serving as new vertex coordinates, and using the same vertex coloring scheme as in Figure 4.9(g). While both the original and the new vertex space are two-dimensional, we can clearly see that in the new vertex space the vertices belonging to the same clusters are also spatially closer, which is both physically meaningful and exemplifies the practical value of the eigenmaps. Figure 4.17(c) is similar to Figure 4.17(b) but is presented using the normalized spectral space coordinates, $\mathbf{q}_n = [u_1(n), u_2(n)]/\|[u_1(n), u_2(n)]\|_2$. In Figure 4.17(d) the clusters are refined using the k-means algorithm, as per Remark 30. The same representations are repeated and shown in

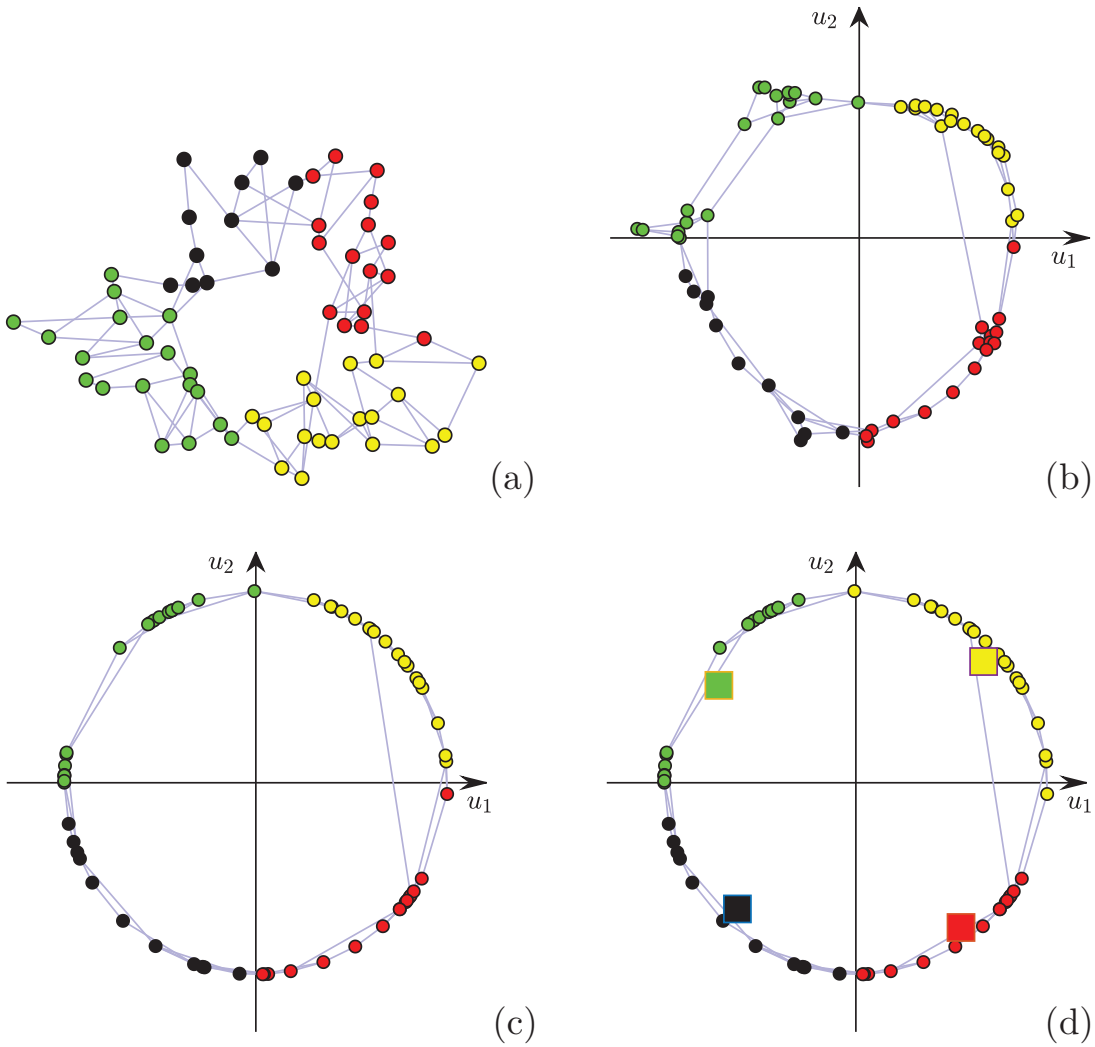


Figure 4.17: Principle of Laplacian eigenmaps and clustering based on the eigenvectors of the graph Laplacian, $\mathbf{L}$. (a) The original graph from Figure 4.9, with the spectral vector $\mathbf{q}_n = [u_1(n), u_2(n)]$, defined by the graph Laplacian eigenvectors $\{\mathbf{u}_1, \mathbf{u}_2\}$, which is used to cluster (color) the vertices. (b) Two-dimensional vertex positions obtained through Laplacian eigenmaps, with the spectral vector $\mathbf{q}_n = [u_1(n), u_2(n)]$ serving as the vertex coordinates (the 2D Laplacian eigenmap). While both the original and this new vertex space are two-dimensional, the new eigenmaps-based space is advantageous in that it emphasizes vertex spectral similarity in a spatial way (physical closeness of spectrally similar vertices). (c) The graph from (b) but produced using normalized spectral space coordinates $\mathbf{q}_n = [u_1(n), u_2(n)]/\|[u_1(n), u_2(n)]\|_2$, as in (4.22). (d) The graph from (c) with clusters refined using the k-means algorithm, as per Remark 30. The centroids of clusters are designated by squares of the same color. The complexity of graph presentation is also significantly reduced through eigenmaps, with most of the edges between strongly connected vertices being very short and located along a circle.

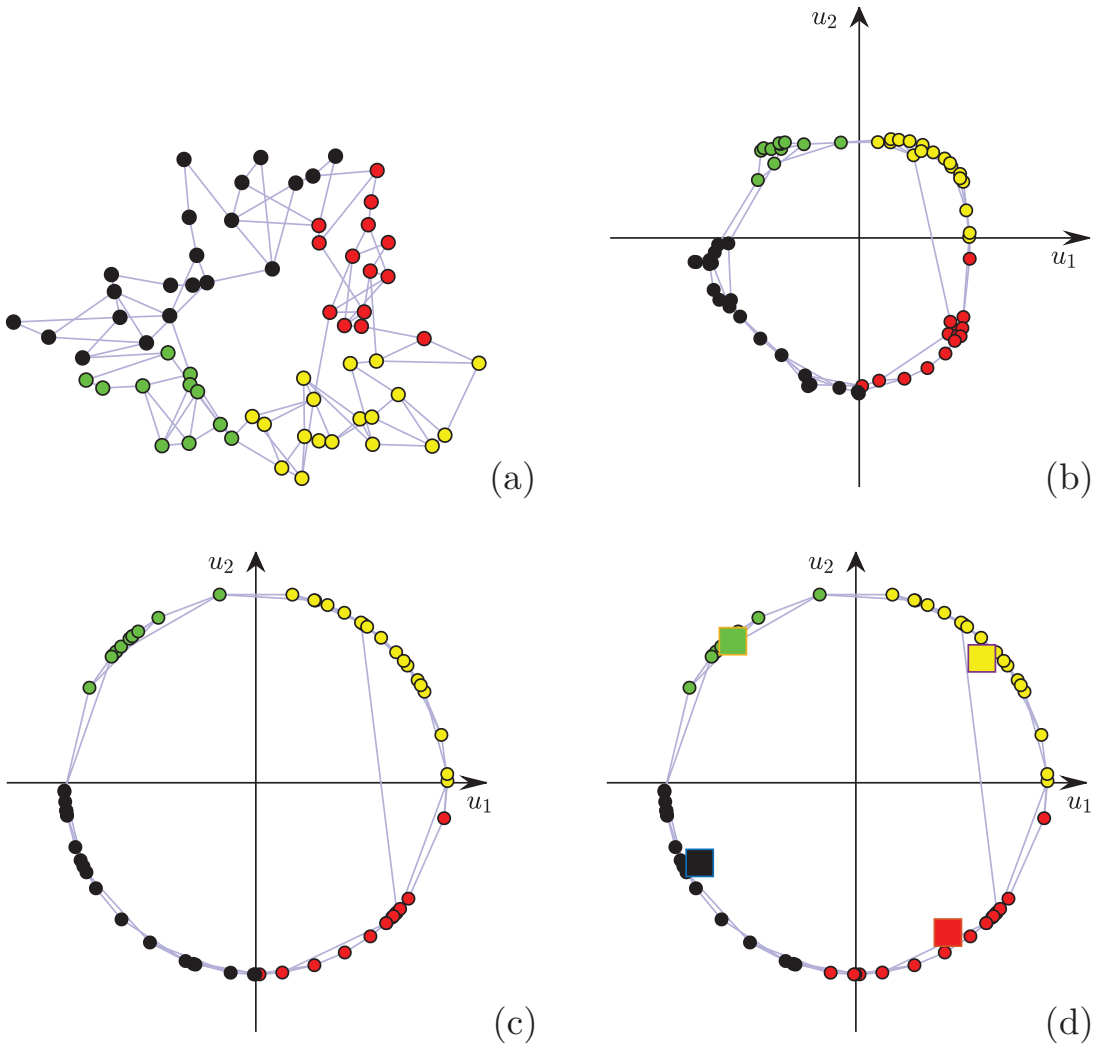


Figure 4.18: Principle of Laplacian eigenmaps and clustering based on the *generalized eigenvectors of the graph Laplacian*, obtained as a solution to $\mathbf{L}\mathbf{u}_k = \lambda_k \mathbf{D}\mathbf{u}_k$. Vertex coloring was produced using the same procedure as in Figure 4.17.

Figures 4.18(a)–(d) for the representation based on the generalized eigenvectors of the graph Laplacian, obtained as a solution to $\mathbf{L}\mathbf{u}_k = \lambda_k \mathbf{D}\mathbf{u}_k$. Finally, in Figures 4.19(a)–(d), the Laplacian eigenmaps and clustering are produced based on the eigenvectors of the normalized graph Laplacian, $\mathbf{L}_N = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$. As expected, the eigenmaps obtained using the generalized Laplacian eigenvectors, in Figure 4.19(b), and the eigenvectors of the normalized Laplacian, in Figure 4.18(b), are different; however, they reduce to the same eigenmaps after spectral

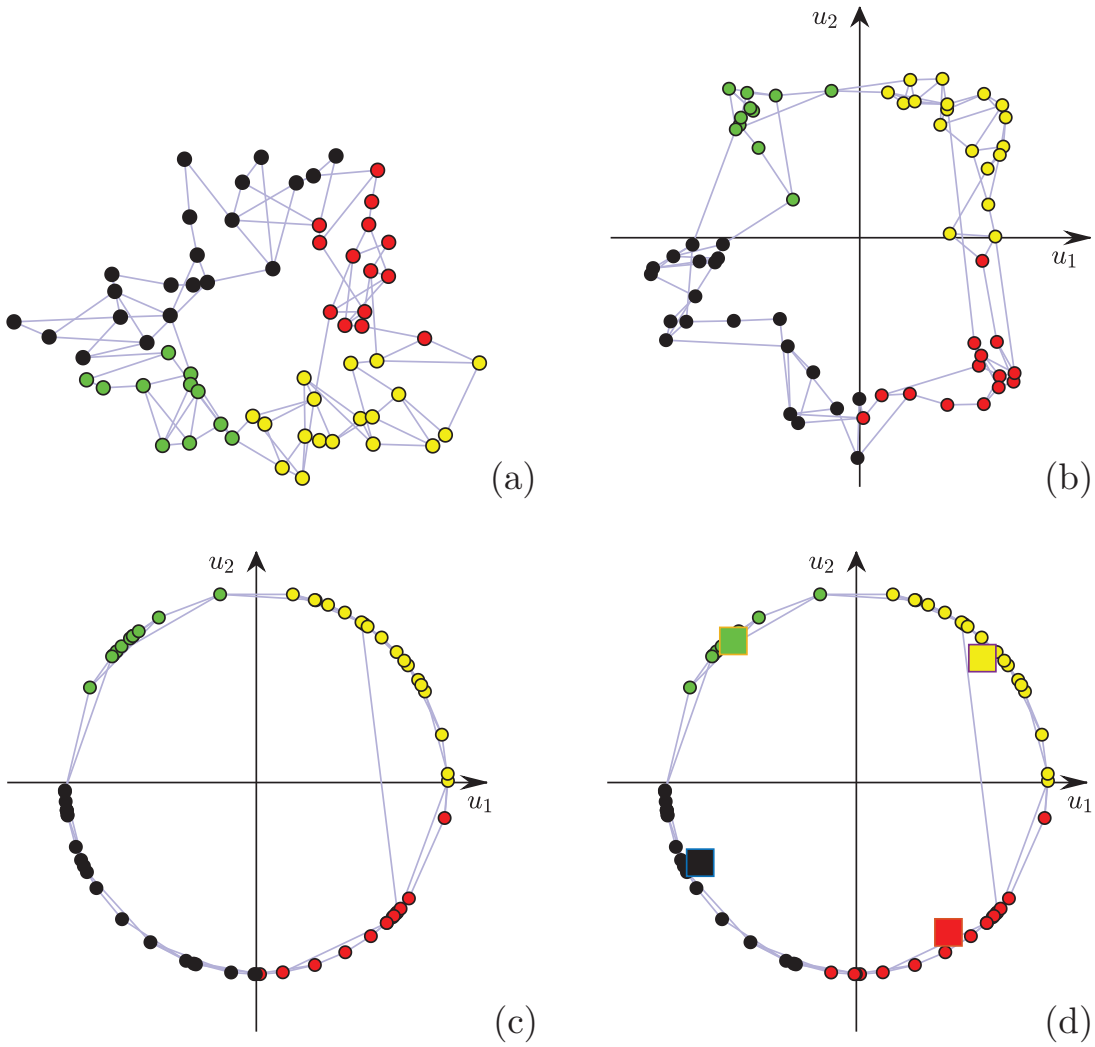


Figure 4.19: Principle of Laplacian eigenmaps and clustering based on the *eigenvectors of the normalized graph Laplacian*, $\mathbf{L}_N = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$. Vertex coloring was performed using the same procedure as in Figure 4.17. The eigenvectors of the normalized graph Laplacian, $\mathbf{v}_k$, are related to the generalized eigenvectors of the graph Laplacian, $\mathbf{u}_k$, through $\mathbf{u}_k = \mathbf{D}^{-1/2}\mathbf{v}_k$, as stated in Remark 27. This means that the signs of these two eigenvectors are the same, $\text{sign}(\mathbf{u}_k) = \text{sign}(\mathbf{v}_k)$. Since in order to obtain $u_1(n)$ and $u_2(n)$, the elements $v_1(n)$ and $v_2(n)$ are multiplied by the same value, $1/D_{nn}$, then $[u_1(n), u_2(n)]/\|[u_1(n), u_2(n)]\|_2 = [v_1(n), v_2(n)]/\|[v_1(n), v_2(n)]\|_2$, thus yielding the same graph forms in (c) and (d) in both this figure and in Figure 4.18.

vector normalization, as shown Figure 4.19(c) and Figure 4.18(c). After the k-means based clustering refinement was applied, in all three cases two vertices switched their initial color (cluster), as shown in Figures 4.17(d), 4.18(d), and 4.19(d).

Observe that the eigenmaps obtained with the normalized forms of the generalized eigenvectors of the Laplacian and the eigenvectors of the normalized Laplacian are the same, and in this case their clustering performances are similar to those based on the eigenmaps produced with the eigenvectors of the original Laplacian.

Remark 34: In general, an independent quantization of two smoothest eigenvectors of the graph Laplacian, $\mathbf{u}_1$ and $\mathbf{u}_2$, will produce four clusters. However, that will not be the case if we analyze the graph with an almost ideal eigenvalue gap (unit value) between λ_2 and λ_3. In other words, when the gap $\delta_r = 1 - \lambda_2/\lambda_3$ tends to 1, that is, $\lambda_2 \to 0$ and $\lambda_1 < \lambda_2 \to 0$, then this case corresponds to a graph with exactly three disjoint subgraph components, with vertices belonging to the disjoint sets $\mathcal{E}$, $\mathcal{H}$, and $\mathcal{K}$. Without loss of generality, assume $N_{\mathcal{E}} > N_{\mathcal{H}} > N_{\mathcal{K}}$. The minimum ratio cut, $CutN(\mathcal{E}, \mathcal{H} \cup \mathcal{K})$ is then obtained with the first indicator vector $x_1(n) = c_{11}$ for $n \in \mathcal{E}$ and $x_1(n) = c_{12}$ for $n \in \mathcal{H} \cup \mathcal{K}$. The second indicator vector will produce the next minimum ratio cut, $CutN(\mathcal{E} \cup \mathcal{K}, \mathcal{H})$ with $x_2(n) = c_{21}$ for $n \in \mathcal{E} \cup \mathcal{K}$ and $x_2(n) = c_{22}$ for $n \in \mathcal{H}$. Following the same analysis as in the case of one indicator vector and the cut of graph into two disjoint subsets of vertices, we can immediately conclude that the two smoothest eigenvectors, $\mathbf{u}_1$ and $\mathbf{u}_2$, which correspond to $\lambda_2 \to 0$ and $\lambda_1 \to 0$, can be used to form an indicator matrix $\mathbf{Y} = [\mathbf{x}_1, \mathbf{x}_2]$, so that the corresponding matrix of the solution (within the graph Laplacian eigenvector space) to the minimization problem of two ratio cuts, has the form $[\text{sign}(\mathbf{u}_1), \text{sign}(\mathbf{u}_2)]$. The elements of these indicator vectors, $[\text{sign}(u_1(n)), \text{sign}(u_2(n))]$, have therefore a subset-wise constant vector form, assuming exactly three different vector values that correspond to individual disjoint sets $\mathcal{E}$, $\mathcal{H}$, and $\mathcal{K}$.

This procedure can be generalized up to every individual vertex becoming a cluster (no clustering). To characterize N independent disjoint sets we will need $(N-1)$ spectral vectors, if the constant eigenvector, $\mathbf{u}_0$, is omitted.

Example 30: The two-dimensional Laplacian eigenmap for the benchmark Minnesota roadmap graph (with $M = 2$) is given in Figure 4.20. In this new space, the spectral vectors $\mathbf{q}_n = [u_2(n), u_3(n)]$, are used as

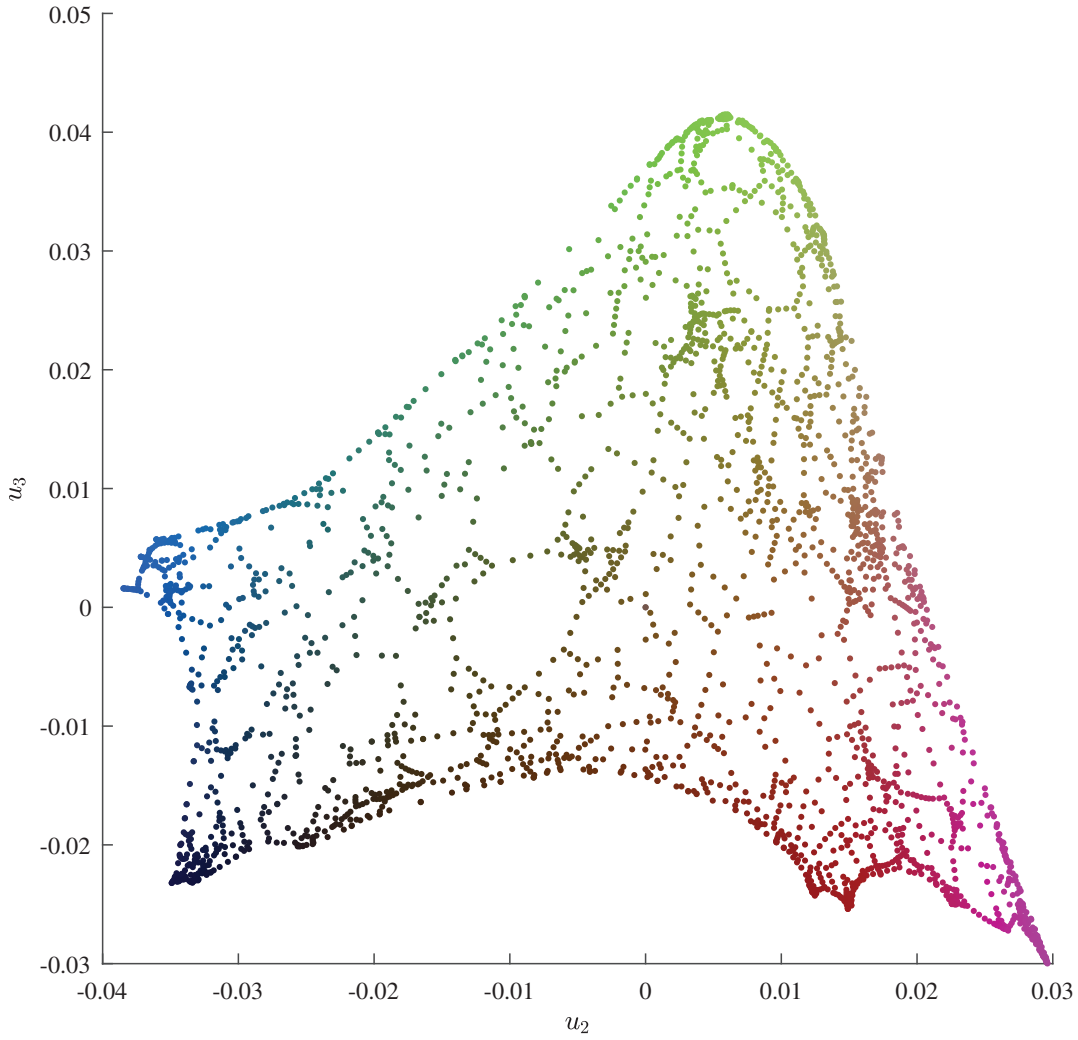


Figure 4.20: Laplacian eigenmaps for the Minnesota road-map graph, produced based on the new two-dimensional spectral vertex positions defined by the Laplacian eigenvectors $\{\mathbf{u}_2,\mathbf{u}_3\}$ as the vertex coordinates (the 2D Laplacian eigenmap).

the coordinates of the new vertex positions. Here, two vertices with similar slow-varying eigenvectors are located close to one another in the new coordinate system defined by $\mathbf{u}_2$ and $\mathbf{u}_3$. This illustrates that the eigenmaps can be considered as a basis for "scale-wise" graph representation.

Example 31: The Laplacian eigenmaps of the Brain Atlas graph from Figure 4.14, whose original vertex locations reside in an $L = 3$ dimensional space, is presented in a new reduced $M = 2$ dimensional space which is defined based on the two smoothest eigenvectors, $\mathbf{u}_1$ and $\mathbf{u}_2$.

This example of vertex dimensionality reduction, with new vertex locations but with the original edges kept, is shown in Figure 4.21.

The generalized eigenvectors of the graph Laplacian, $\mathbf{u}_k$, for $k = 1, 2, 3, 4, 5, 6$, are shown in Figure 4.22(a) using the standard colormap in both the original three-dimensional and the reduced two-dimensional space, as shown in Figure 4.22(b).

Example 32: Vertices of a three-dimensional Swiss roll graph are shown in Figure 4.24(a). The vertex locations in this original $L = 3$ dimensional space are calculated as $x_n = \alpha_n \cos(\alpha_n)/(4\pi)$, $y_n = \beta_n$, and $z_n = \alpha_n \sin(\alpha_n)/(4\pi)$, $n = 0, 1, 2, \ldots, N-1$, with α_n randomly taking values between π and 4π, and β_n from -1.5 to 1.5. The edge weights are calculated using $W_{mn} = \exp(-d_{mn}^2/(2\kappa^2))$, where d_{mn} is the square Euclidean distance between the vertices m and n, and $W_{mn} = 0$ if $d_{mn} \geq 0.15$ with $\kappa = 0.1$. The resulting three-dimensional Swiss roll graph is shown in Figure 4.24(b), while Figure 4.24(c) shows the same graph but with vertices colored (clustered) using the normalized graph Laplacian eigenvectors, $u_1(n)$ and $u_2(n)$, as a colormap. The same vectors are then used in Figure 4.24(d) as the new coordinates in the reduced two-dimensional Laplacian eigenmap vertex space ($M = 2$) for the Swiss roll graph.

4.5 Pseudo-Inverse of Graph Laplacian-Based Mappings

The graph Laplacian is a singular matrix (since $\lambda_0 = 0$) for which an inverse does not exist. To deal with this issue, the pseudo-inverse of the graph Laplacian, $\mathbf{L}^+$, is defined as a matrix that satisfies the property

$$\mathbf{L}\mathbf{L}^+ = \begin{bmatrix} 0 & \mathbf{0}_{1\times(N-1)} \\ \mathbf{0}_{(N-1)\times 1} & \mathbf{I}_{(N-1)\times(N-1)} \end{bmatrix}, \tag{4.35}$$

where we assumed that the graph is connected. The eigenvalues of the graph Laplacian pseudo-inverse are therefore the inverses of the original eigenvalues, $\{0, 1/\lambda_1, \ldots, 1/\lambda_{N-1}\}$, while it shares the same eigenvectors with the original graph Laplacian, $\mathbf{u}_0$, $\mathbf{u}_1$, $\ldots$, $\mathbf{u}_{N-1}$. *The eigenmaps for which the spectral coordinates are scaled based on the eigenvalues of the pseudo-inverse of graph Laplacian can be interpreted within the Principal Component Analysis (PCA) framework in the following way.*

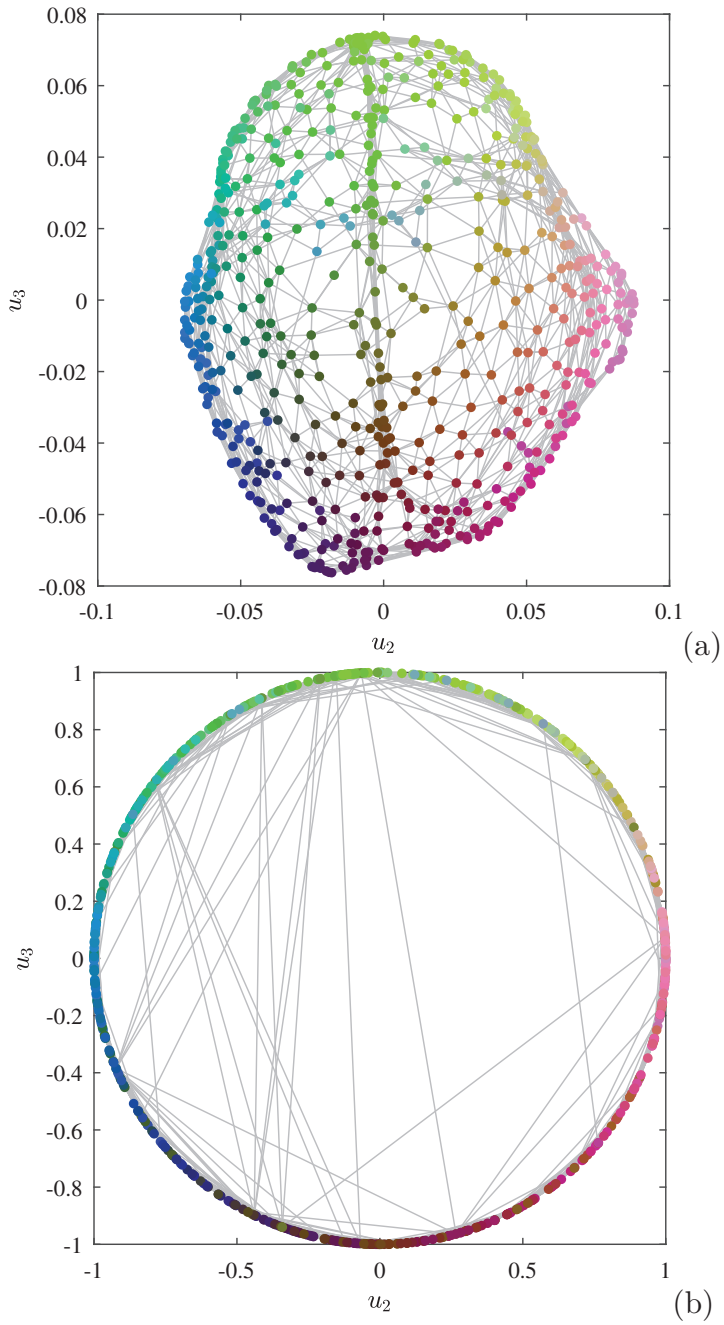


Figure 4.21: Brain atlas representation based on normalized spectral vectors. (a) A two-dimensional Laplacian eigenmap based on the generalized Laplacian eigenvectors. The original $L = 3$ dimensional graph from Figure 4.14 is reduced to a two-dimensional representation based on the two smoothest eigenvectors, $u_1(n)$ and $u_2(n)$, which both serve as spectral coordinates and define color templates in the colormap, as in Figure 4.14. (b) Eigenmaps from (a) but in the space of normalized spectral space coordinates, $\mathbf{q}_n = [u_2(n), u_3(n)]/\|[u_2(n), u_3(n)]\|_2$, with the complexity of graph representation now significantly reduced. Observe that most edges only exists between strongly connected vertices located along the circle.

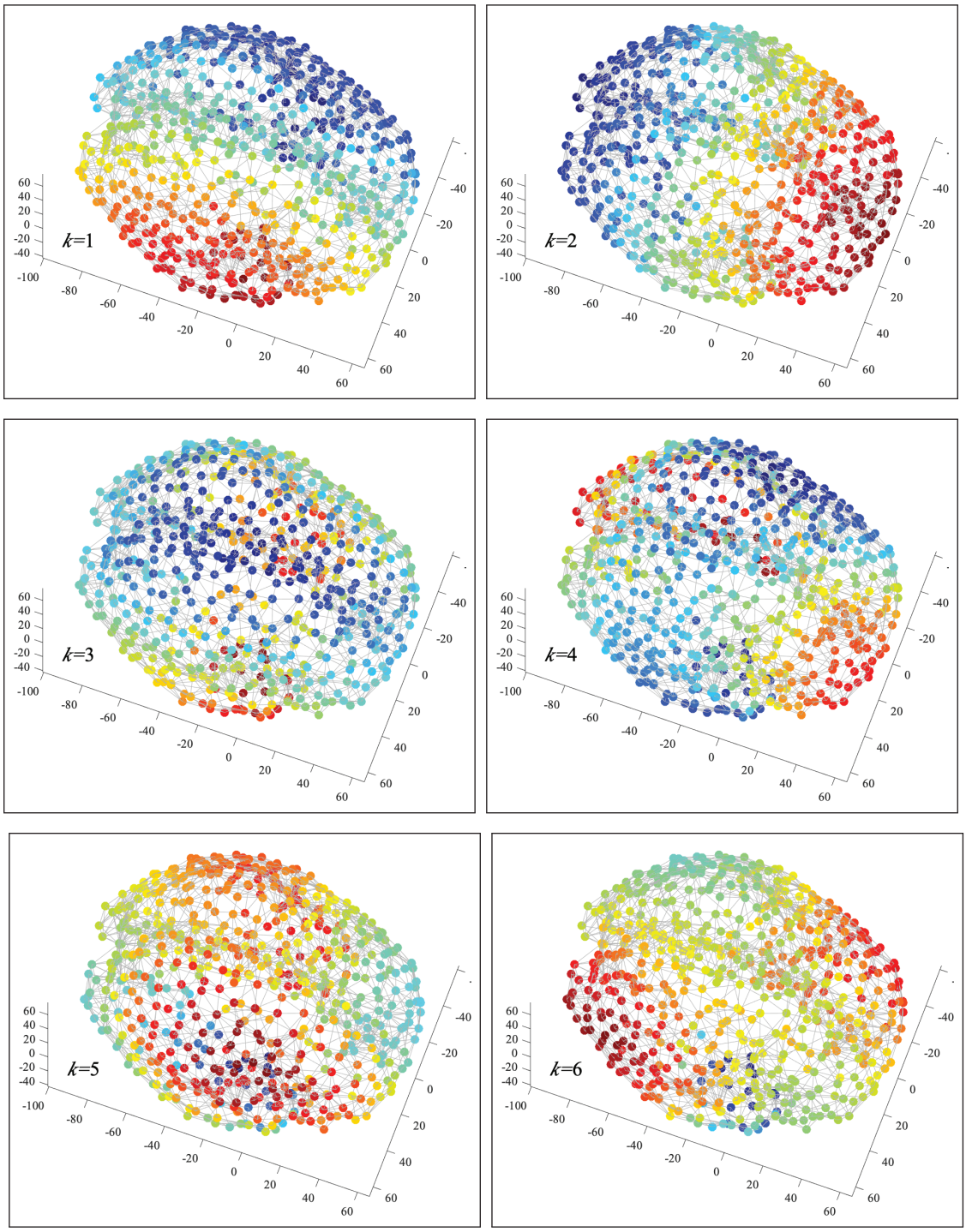


Figure 4.22: Generalized eigenvectors, $\mathbf{u}_k$, $k = 1, 2, 3, 4, 5, 6$, of the graph Laplacian of the Brain Atlas graph, shown using vertex coloring in the original three-dimensional vertex space. Each panel visualizes a different $\mathbf{u}_k$, $k = 1, 2, 3, 4, 5, 6$.

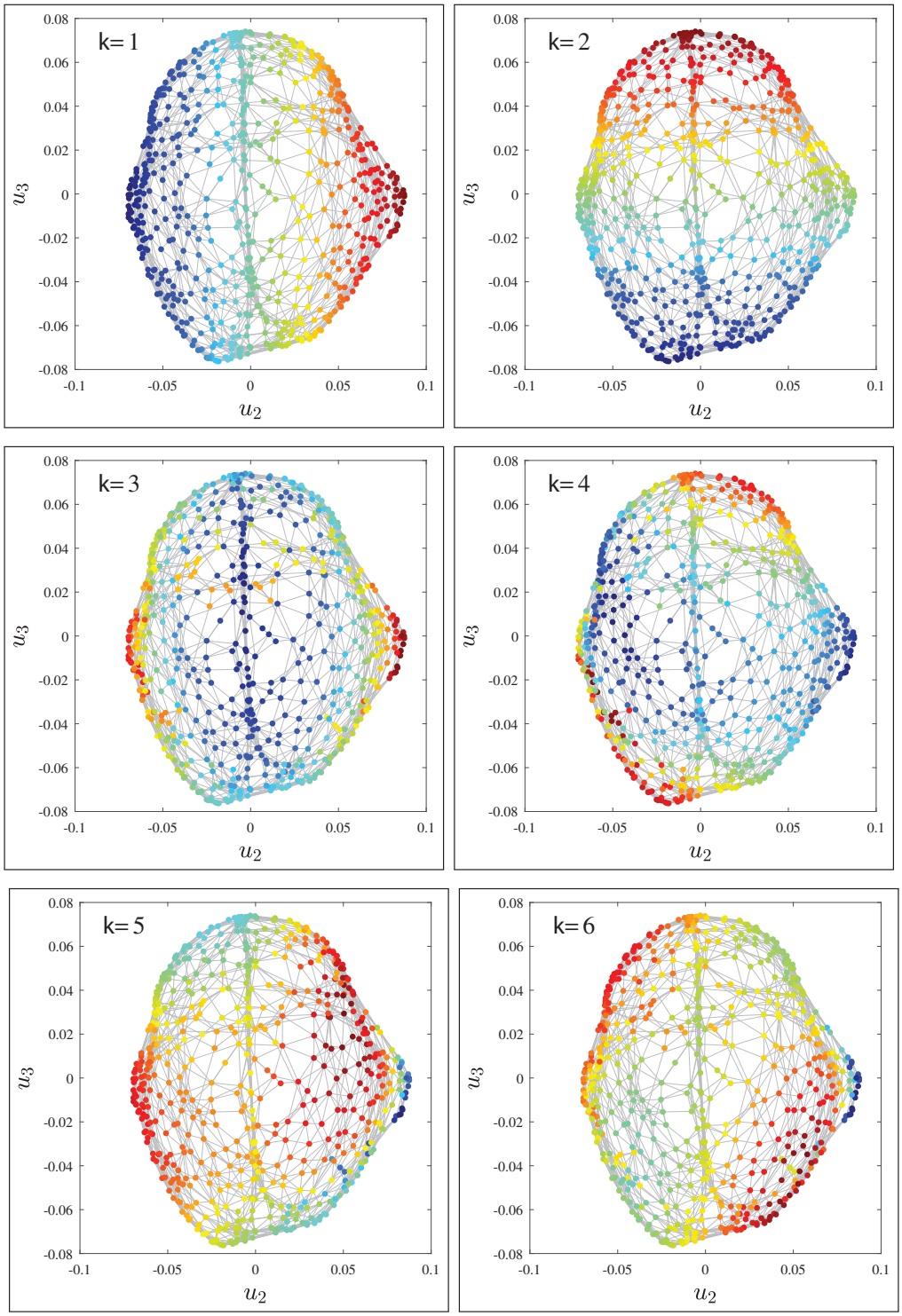


Figure 4.23: Laplacian eigenmaps of the Brain Atlas graph in the reduced two-dimensional space defined by the two smoothest generalized eigenvectors of the graph Laplacian, $\mathbf{u}_1$ and $\mathbf{u}_2$. The panels each visualize a different generalized eigenvector, $\mathbf{u}_k$, $k = 1, 2, 3, 4, 5, 6$.

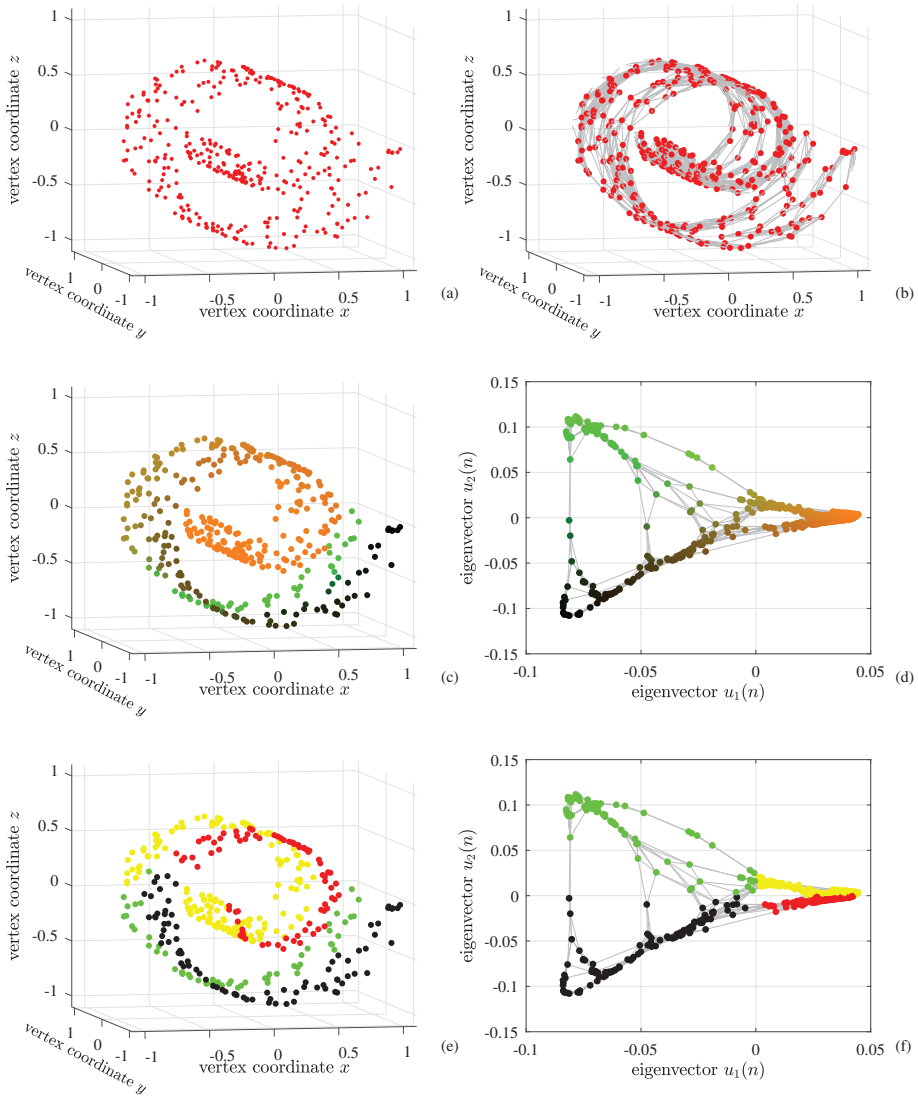


Figure 4.24: Laplacian eigenmaps based dimensionality reduction for the Swiss roll graph. (a) Vertex locations for the Swiss roll graph in the original $L = 3$ dimensional space with $N = 500$ points (vertices). (b) The Swiss roll graph with edges whose weights are calculated based on the Euclidean distances between vertices. (c) The Swiss roll graph with vertices colored using the normalized graph Laplacian eigenvectors, $u_1(n)$ and $u_2(n)$, as a colormap. (d) The same vectors are used as the new coordinates (spectral vectors) in a reduced two-dimensional Laplacian eigenmap vertex space ($M = 2$). The vertices with high similarity (similar values of the smoothest eigenvectors) are located close to one another, thus visually indicating the expected similarity of data observed at these vertices. (e) Clustering of the Swiss roll graph, in the original $L = 3$ dimensional space, using the two smoothest eigenvectors, $u_1(n)$ and $u_2(n)$. (f) Clustering of the Swiss roll graph using the two smoothest eigenvectors, $u_1(n)$ and $u_2(n)$, presented in the $M = 2$ eigenmap space, where for every vertex its spatial position (quadrant of the coordinate system) indicates the cluster where it belongs.

Notice that the M-dimensional eigenmaps based on the pseudo-inverse of the Laplacian are the same as those for the original graph Laplacian, since they share the same eigenvectors. If the spectral vectors $\mathbf{q}_n = [u_1(n), u_2(n), \ldots, u_M(n)]$ are scaled with the square roots of the eigenvalues of the Laplacian pseudo-inverse, we obtain

$$\mathbf{q}_n = \left[\frac{u_1(n)}{\sqrt{\lambda_1}}, \frac{u_2(n)}{\sqrt{\lambda_2}}, \ldots, \frac{u_M(n)}{\sqrt{\lambda_M}}\right].$$

The elements of this spectral vector are now equal to the first M elements (omitting $0 \cdot u_0(n)$) of the full-dimension spectral vector

$$\mathbf{q}_n = [u_1(n), u_2(n), \ldots, u_{N-1}(n)]\bar{\mathbf{\Lambda}}^{-1/2}, \tag{4.36}$$

where $\bar{\mathbf{\Lambda}}$ is a diagonal matrix with elements $\lambda_1, \lambda_2, \ldots, \lambda_{N-1}$.

4.5.1 Commute Time Mapping

Physical meaning of the new vector positions in the spectral space, defined by (4.36), is related to the notion of *commute time*, which is a property of a diffusion process on a graph (Horaud, 2009; Qiu and Hancock, 2007). The commute time, $CT(m, n)$ between vertices m and n is defined as the expected time for the random walk to reach vertex n starting from vertex m, and then to return. The commute time is therefore proportional to the Euclidean distance between these two vertices, with the vertex positions in the new spectral space defined by $\mathbf{q}_n$ in (4.36), that is

$$CT(m, n) = V_{\mathcal{V}} \|\mathbf{q}_m - \mathbf{q}_n\|_2^2 = V_{\mathcal{V}} \sum_{i=1}^{N-1} (q_i(m) - q_i(n))^2, \tag{4.37}$$

where $V_{\mathcal{V}}$ is the volume of the whole graph, $V_{\mathcal{V}} = \sum_{n=0}^{N-1} D_{nn}$.

To put this into perspective, in a graph representation of a *resistive electric circuit/network*, for which the edge weights are equal to the conductances (inverse resistances, see Part III), the commute time, $CT(m, n)$, is defined as the equivalent resistance between the electric circuit nodes (vertices) m and n (Chandra *et al.*, 1996).

The covariance matrix of the scaled spectral vectors in (4.36) is given by

$$\mathbf{S} = \frac{1}{N}\sum_{n=0}^{N-1} \mathbf{q}_n^T \mathbf{q}_n = \frac{1}{N}\bar{\mathbf{\Lambda}}^{-1}.$$

In other words, the principal directions in the reduced dimensionality space of M eigenvectors, $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_M$, correspond to the maximum variance of the graph embedding, since $1/\lambda_1 > 1/\lambda_2 > \cdots > 1/\lambda_M$. This, in turn, directly corresponds to *principal component analysis (PCA)*.

Remark 35: **Two-dimensional case comparison.** The two-dimensional spectral space of the standard graph Laplacian eigenvectors is defined by $\mathbf{u}_1$ and $\mathbf{u}_2$, while the spectral vector in this space is given by

$$\mathbf{q}_n = [u_1(n), u_2(n)]. \tag{4.38}$$

In the case of commute time mapping, the two-dimensional spectral domain of the vertices becomes

$$\mathbf{q}_n = \left[\frac{u_1(n)}{\sqrt{\lambda_1}}, \frac{u_2(n)}{\sqrt{\lambda_2}}\right], \tag{4.39}$$

that is, the commute time mapping is related to the graph Laplacian mapping through axis scaling by $1/\sqrt{\lambda_k}$.

We can conclude that when $\lambda_1 \approx \lambda_2$, the two mappings in (4.38) and (4.39) are almost the same, when normalized.

However, when $\lambda_1 \ll \lambda_2$, the relative eigenvalue gap between the one dimensional and two-dimensional spectral space is large, since $\delta_r = 1 - \lambda_1/\lambda_2$ is close to 1. This means that the segmentation into two disjoint subgraphs will be "close" to the original graph, while at the same time this also indicates that the eigenvector $\mathbf{u}_2$ does not contribute to a new "closer" segmentation (in the sense of Section 4.3.2), since its gap $\delta_r = 1 - \lambda_2/\lambda_3$ is not small. Therefore, the influence of $\mathbf{u}_2$ should be reduced, as compared to the standard spectral vector of graph Laplacian where both $\mathbf{u}_1$ and $\mathbf{u}_2$ employ unit weights to give $\mathbf{q}_n = [u_1(n), u_2(n)]$. Such downscaling of the influence of the almost irrelevant eigenvector, $\mathbf{u}_2$, when $\lambda_1 \ll \lambda_2$, is equivalent to the commute time mapping, since $\mathbf{q}_n = [\frac{u_1(n)}{\sqrt{\lambda_1}}, \frac{u_2(n)}{\sqrt{\lambda_2}}] = \frac{1}{\sqrt{\lambda_1}}[u_1(n), u_2(n)\sqrt{\frac{\lambda_1}{\lambda_2}}] \sim [u_1(n), 0]$.

For example, for the graph from Example 29, shown in Figure 4.17(a), the commute time mapping will produce the same vertex presentation as in Figure 4.17(b), which is obtained with the eigenvectors of the graph Laplacian, when the vertical axis, $\mathbf{u}_2$, is scaled by

$$\sqrt{\frac{\lambda_1}{\lambda_2}} = \sqrt{\frac{0.0286}{0.0358}} = 0.8932.$$

This eigenmap will also be very close to the eigenmap in Figure 4.17(b), produced based on the graph Laplacian eigenvectors and the spectral vector $\mathbf{q}_n = [u_1(n), u_2(n)]$.

4.5.2 Diffusion (Random Walk) Mapping

Finally, we shall now relate the commute time mapping to the diffusion mapping.

Definition: Diffusion on a graph deals with the problem of propagation along the edges of a graph, whereby at the initial step, $t = 0$, the random walk starts at a vertex n. At the next step $t = 1$, the walker moves from its current vertex n to one of its neighbors l, chosen at random from the neighbors of n. The probability of going from vertex n to vertex l is equal to the ratio of the weight W_{nl} and the sum of all possible edge weights from the vertex n, that is

$$P_{nl} = \frac{W_{nl}}{\sum_l W_{nl}} = \frac{1}{D_{nn}} W_{nl}. \tag{4.40}$$

When considering all vertices together, such probabilities can be written in a matrix form, within the weight of a random walk matrix, defined as in (2.10), by

$$\mathbf{P} = \mathbf{D}^{-1}\mathbf{W}. \tag{4.41}$$

Diffusion distance. *The Diffusion distance* between the vertices m and n, denoted by $D_f(m, n)$, is equal to the distance between the vector (N-dimensional ordered set) of probabilities for a random walk to move from a vertex m to all other vertices (as in (4.40)), given by

$$\mathbf{p}_m = [P_{m0}, P_{m1}, \ldots, P_{m(N-1)}]$$

and the corresponding vector of probabilities for a random walk to move from a vertex n to all other vertices, given by

$$\mathbf{p}_n = [P_{n0}, P_{n1}, \ldots, P_{n(N-1)}],$$

that is

$$\begin{aligned} D_f^2(m,n) &= \|(\mathbf{p}_m - \mathbf{p}_n)\mathbf{D}^{-1/2}\|_2^2 V_{\mathcal{V}} \\ &= \sum_{i=0}^{N-1} (P_{mi} - P_{ni})^2 \frac{1}{D_{ii}} V_{\mathcal{V}} \end{aligned}$$

where $V_{\mathcal{V}} = \sum_{n=0}^{N-1} D_{nn}$ is constant for a given graph, which is equal to the sum of degrees (volume) of all graph vertices in $\mathcal{V}$.

Example 33: For the graph from Figure 2.2, with its weight matrix, $\mathbf{W}$, and the degree matrix, $\mathbf{D}$, given respectively in (2.4) and (2.6), the random walk weight matrix in (4.41) is of the form

$$\mathbf{P} = \begin{array}{c} \\ \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \\ \mathbf{p}_4 \\ \mathbf{p}_5 \\ \mathbf{p}_6 \\ \mathbf{p}_7 \end{array} \begin{array}{c} \begin{array}{cccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{bmatrix} 0 & 0.19 & 0.61 & 0.20 & 0 & 0 & 0 & 0 \\ 0.28 & 0 & 0.43 & 0 & 0.28 & 0 & 0 & 0 \\ 0.47 & 0.22 & 0 & 0.16 & 0.15 & 0 & 0 & 0 \\ 0.29 & 0 & 0.32 & 0 & 0 & 0 & 0.39 & 0 \\ 0 & 0.21 & 0.21 & 0 & 0 & 0.46 & 0 & 0.12 \\ 0 & 0 & 0 & 0 & 0.77 & 0 & 0 & 0.23 \\ 0 & 0 & 0 & 0.50 & 0 & 0 & 0 & 0.50 \\ 0 & 0 & 0 & 0 & 0.23 & 0.25 & 0.52 & 0 \end{bmatrix} \end{array} \tag{4.42}$$

with $V_{\mathcal{V}} = 7.46$.

Therefore, the diffusion distance between, for example, the vertices $m = 1$ and $n = 3$, for the $t = 1$ step, is

$$D_f(1,3) = \|(\mathbf{p}_1 - \mathbf{p}_3)\mathbf{D}^{-1/2}\|_2 \sqrt{V_{\mathcal{V}}} = 1.54,$$

while the diffusion distance between the vertices $m = 6$ and $n = 3$ is $D_f(6,3) = 2.85$. From this simple example, we can see that the diffusion distance is larger for vertices $m = 6$ and $n = 3$ than for the neighboring vertices $m = 1$ and $n = 3$. This result is in a perfect accordance with the clustering scheme (expected similarity) in Figure 4.7(b), where the vertices $m = 1$ and $n = 3$ are grouped into the same cluster, while the vertices $m = 6$ and $n = 3$ belong to different clusters.

The probability vectors, $\mathbf{p}_n$, are called the *diffusion clouds* (in this case for step $t = 1$), since they resemble a cloud around a vertex n. The diffusion distance can then be considered as a distance between the diffusion clouds (sets of data) around a vertex m and a vertex n. If the vertices are well connected (approaching a complete graph structure) then this distance is small, while for vertices with long paths between them, this distance is large.

The diffusion analysis can be easily generalized to any value of the diffusion step, t, whereby after t steps, the matrix of probabilities in (4.41) becomes

$$\mathbf{P}^t = (\mathbf{D}^{-1}\mathbf{W})^t.$$

The elements of this matrix, denoted by $P_{mn}^{(t)}$, are equal to the probabilities that a random walker moves from a vertex m to a vertex n, in t steps. The t-step diffusion distance between the vertices m and n, is accordingly defined as

$$D_f^{(t)}(m,n) = \|(\mathbf{p}_m^{(t)} - \mathbf{p}_n^{(t)})\mathbf{D}^{-1/2}\|_2\sqrt{V_{\mathcal{V}}},$$

where

$$\mathbf{p}_m^{(t)} = [P_{m0}^{(t)}, P_{m1}^{(t)}, \ldots, P_{m(N-1)}^{(t)}]$$

and

$$\mathbf{p}_n^{(t)} = [P_{n0}^{(t)}, P_{n1}^{(t)}, \ldots, P_{n(N-1)}^{(t)}].$$

It can be shown that *the diffusion distance is equal to the Euclidean distance between the considered vertices when they are presented in a new space of their generalized Laplacian eigenvectors, which are then scaled by their corresponding eigenvalues;* **this new space is referred to as the diffusion map** (*cf.* eigenmaps).

The eigenanalysis relation for the random walk weight matrix for the state $t = 1$ now becomes

$$(\mathbf{D}^{-1}\mathbf{W})\,\mathbf{u}_k = \lambda_k^{(P)}\mathbf{u}_k.$$

Since the weight matrix can be written as $\mathbf{W} = \mathbf{D} - \mathbf{L}$, this yields $\mathbf{D}^{-1}(\mathbf{D} - \mathbf{L})\mathbf{u}_k = \lambda_k^{(P)}\mathbf{u}_k$, or

$$(\mathbf{I} - \mathbf{D}^{-1}\mathbf{L})\mathbf{u}_k = \lambda_k^{(P)}\mathbf{u}_k,$$

to finally produce the generalized graph Laplacian equation,

$$\mathbf{L}\mathbf{u}_k = \lambda_k \mathbf{D}\mathbf{u}_k,$$

with $\lambda_k = (1 - \lambda_k^{(P)})$. *This relation indicates that a one-step diffusion mapping is directly obtained from the corresponding generalized graph Laplacian mapping.*

After t steps, the random walk matrix (of probabilities) becomes

$$\mathbf{P}^t = (\mathbf{D}^{-1}\mathbf{W})^t,$$

for which the eigenvalues are $\lambda_k^{(P)t} = (1 - \lambda_k)^t$, while the (right) eigenvectors remain the same as for the graph Laplacian, see (3.7).

The spectral space for vertices, for a t-step diffusion process (*diffusion mapping*), is then defined based on the spectral vector

$$\mathbf{q}_n = [u_1(n), u_2(n), \ldots, u_{N-1}(n)](\mathbf{I} - \bar{\mathbf{\Lambda}})^t,$$

and is equal to the generalized Laplacian spectral space mapping, whereby the axis vectors $\mathbf{q}_n = [u_1(n), u_2(n), \ldots, u_{N-1}(n)]$ are multiplied by the corresponding eigenvalues, $(1 - \lambda_k)^t$.

It can be shown that the diffusion distance between vertices in the new diffusion map space is equal to their Euclidean distance (Coifman and Lafon, 2006), that is

$$D_f^{(t)}(m, n) = \sqrt{V_{\mathcal{V}}}\|\mathbf{q}_m - \mathbf{q}_n\|_2. \tag{4.43}$$

Example 34: For the graph from Figure 2.2, whose weight matrix, $\mathbf{W}$, and the degree matrix, $\mathbf{D}$, are defined in (2.4) and (2.6), the diffusion distance between the vertices $m = 1$ and $n = 3$ can be calculated using (4.43) as

$$D_f^{(1)}(1, 3) = \sqrt{V_{\mathcal{V}}}\|(\mathbf{q}_1 - \mathbf{q}_3)\|_2 = 1.54,$$

where the spectral vectors, $\mathbf{q}_1 = [u_1(1)(1 - \lambda_1)^1, \ldots, u_N(1)(1 - \lambda_N)^1]$ and $\mathbf{q}_3 = [u_1(3)(1 - \lambda_1)^1, \ldots, u_N(3)(1 - \lambda_N)^1]$ are obtained using the generalized graph Laplacian eigenvectors, $\mathbf{u}_k$, and the corresponding eigenvalues, λ_k, from $\mathbf{L}\mathbf{u}_k = \lambda_k \mathbf{D}\mathbf{u}_k$. This is the same diffusion distance value, $D_f(1, 3)$, as in Example 33.

Dimensionality reduced diffusion maps. Dimensionality of the vertex representation space can be reduced in diffusion maps by keeping

only the eigenvectors that correspond to the M most significant eigenvalues, $(1-\lambda_k)^t$, $k = 1, 2, \ldots, M$, in the same way as for the Laplacian eigenmaps, For example, the two-dimensional spectral domain of the vertices in the diffusion mapping is defined as

$$\mathbf{q}_n = [u_1(n)(1-\lambda_1)^t, u_2(n)(1-\lambda_2)^t].$$

While the *analysis and intuition for the diffusion mapping is similar to that for the commute time mapping*, presented in Remark 35, diffusion maps have an additional degree of freedom, the step t.

Example 35: For the graph in Figure 4.10, which corresponds to a set of real-world images, the commute time two-dimensional spectral vectors in (4.39), normalized by the first eigenvector value through a multiplication of its coordinates by $\sqrt{\lambda_1}$, assume the form

$$\mathbf{q}_n = \left[u_1(n), \frac{\sqrt{\lambda_1}}{\sqrt{\lambda_2}} u_2(n)\right] = [u_1(n), 0.62u_2(n)].$$

The corresponding vertex colors designate diffusion-based clustering, as shown in Figure 4.25(a). Figure 4.25(b) shows the vertices of this graph, colored with the two-dimensional diffusion map spectral vectors, which are normalized by $(1-\lambda_1)$, to yield

$$\mathbf{q}_n = \left[u_1(n), \frac{1-\lambda_2}{1-\lambda_1} u_2(n)\right] = [u_1(n), 0.09u_2(n)].$$

Finally, the sum over all steps, $t = 0, 1, 2, \ldots$, of the diffusion space yields

$$\mathbf{q}_n = [u_1(n), u_2(n), \ldots, u_{N-1}(n)]\bar{\mathbf{\Lambda}}^{-1},$$

since the sum of a geometric progression is equal to

$$\sum_{t=0}^{\infty} (\mathbf{I} - \bar{\mathbf{\Lambda}})^t = \bar{\mathbf{\Lambda}}^{-1}.$$

This mapping also corresponds to the cumulative diffusion distance, given by

$$D_c(n, l) = \sum_{t=0}^{\infty} D_f^{(t)}(n, l).$$

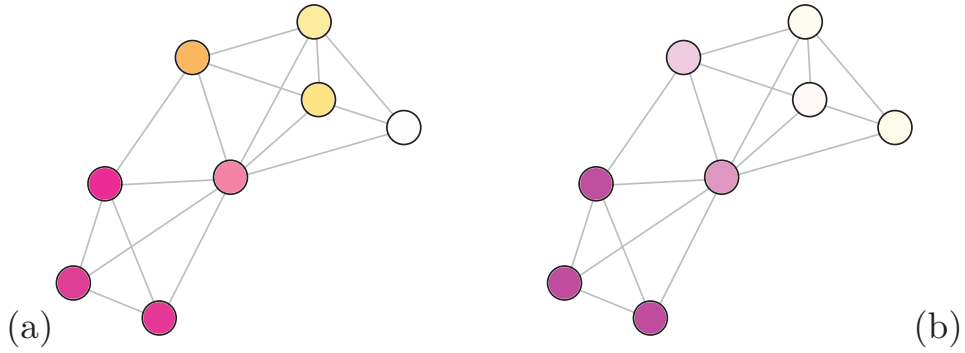


Figure 4.25: Graph structure for the images from Figure 4.10, with vertex color embedding which corresponds to the two-dimensional normalized spectral vectors in (a) the commute time representation, $\mathbf{q}_n = [u_1(n), 0.62u_2(n)]$, and (b) the spectral eigenvectors of the diffusion process, $\mathbf{q}_n = [u_1(n), 0.09u_2(n)]$, with $t = 1$. For the commute time presentation in (a), the graph Laplacian eigenvectors, $\mathbf{u}_1$ and $\mathbf{u}_2$, are used, while for the diffusion process presentation in (b) the generalized Laplacian eigenvectors, $\mathbf{u}_1$ and $\mathbf{u}_2$, are used.

The diffusion eigenmaps can be therefore obtained by appropriate axis scaling of the standard eigenmaps, produced by the generalized eigenvectors of the graph Laplacian.

Remark 36: The commute time and the diffusion process mappings are related in the same way as the mappings based on the graph Laplacian eigenvectors and the generalized eigenvectors of the graph Laplacian.

4.6 Summary of Embedding Mappings

A summary of the considered embedding mappings is given in Table 4.1. Notice that various normalization schemes may be used to obtain the axis vectors, $\mathbf{y}_n$, from the spectral vectors, $\mathbf{q}_n$ (see Algorithm 3).

These examples of dimensionality reduction reveal close connections with spectral clustering algorithms developed in standard machine learning and computer vision; in this sense, the notions of *dimensionality reduction and clustering can be considered as two sides of the same coin* (Belkin and Niyogi, 2003). In addition to the reduction of dimensionality for visualization purposes, the resulting spectral vertex space of lower dimensionality may be used to mitigate the complexity and accuracy issues experienced with classification algorithms, or in other words to bypass the course of dimensionality.

A recent approach to graph dimensionality reduction, called the Uniform Manifold Approximation and Projection (UMAP), can be found in McInnes *et al.* (2018). This dimension reduction technique may be used for visualization similarly to t-distributed Stochastic Neighbor Embedding (t-SNE), which employs a probabilistic approach whereby, with high probability, similar objects are modeled by nearby points and dissimilar objects by distant points, as in van der Maaten and Hinton (2008).

5

Graph Sampling Strategies

In the case of extremely large graphs, subsampling and down-scaling of graphs is a prerequisite for their analysis (Leskovec and Faloutsos, 2006). For a given large (in general directed) graph, $\mathcal{G}$, with N vertices, its resampling aims to produce a much simpler graph which retains most of the properties of the original graph, but is both less complex and more physically and computationally meaningful. The similarity between the original large graph $\mathcal{G}$, and the down-scaled graph, $\mathcal{S}$, with M vertices, where $M \ll N$, is defined with respect to the set of parameters of interest, like for example, the connectivity or distribution on a graph. Such criteria may also be related to the spectral behavior of graphs.

5.1 Graph Down-Sampling Strategies

Several methods exist for graph down-scaling, of which some are listed below.

- The simplest method for graph down-sampling is the *random vertex or random node (RN) selection method*, whereby a random subset of vertices is used for the analysis and representation of large graphs and data observed on such large graphs. Even though

the vertices are here selected with equal probabilities, this method produces good results in practical applications.

- Different from the RN method, where the vertices are selected with a uniform probability, *the random degree vertex/node (RDN) selection method* is based on the probability of vertex selection that is proportional to the vertex degree. In other words, vertices with more connections, thus having larger $D_n = \sum_m W_{nm}$, are selected with higher probability. This makes the RDN approach biased with respect to highly connected vertices.

- *The PageRank method* is similar to the RDN, and is based on the vertex rank. The PageRank is defined by the importance of the vertices connected to the considered vertex n. Then, the probability that a vertex n will be used in a down-scaled graph is proportional to the PageRank of this vertex. This method is also known as the *random PageRank vertex (RPN) selection*, and is biased with respect to the highly connected vertices (with a high PageRank).

- A method based on a random selection of edges that will remain in the simplified graph is called *the random edge (RE) method*. This method may lead to graphs that are not well connected, and which exhibit large diameters.

- The RE method may be combined with random vertex selection to yield *a combined RNE method*, whereby the initial random vertex selection is followed by a random selection of one of the edges that is connected to the selected vertex.

- In addition to these methods, more sophisticated methods based on random vertex selection and *random walk (RW) analysis* may be defined. For example, we can randomly select a small subset of vertices and form several random walks starting from each selected vertex. The *Random Walk (RW)*, *Random Jump (RJ)* and *Forest Fire graph* down-scaling strategies are all defined in this way.

5.2 Graph Sparsification

We now provide an in-depth discussion of graph sparsification, one of the main graph sampling strategies that approximates a given graph by a sparse graph (a graph for which the number of the edges is significantly smaller than quadratic in the number of vertices). Appropriately sparsified graphs allow for a simpler analysis of large graphs, while producing similar results as if the original graphs were analyzed.

Definition: A subgraph or *sparsifier*, $\mathcal{G}'$, of a graph, $\mathcal{G}$, is a graph which maintains the same set of vertices, $\mathcal{V}$, but with a fewer edges. The design of a sparsification strategy should ensure that a desired property/operation of the original graph is approximately preserved.

5.2.1 Cut-Preserving Sparsification

This approach to the sparsification of graphs aims at preserving (approximately) graph cuts. Consider an unweighted graph $\mathcal{G}$ with N vertices. A new, cut-preserving sparsified graph $\mathcal{G}'$ is then obtained by randomly pruning the edges of the original graph $\mathcal{G}$ with the aim of preserving the cut values. The set of vertices is the same for both the original and the resulting graphs. Assume next that the vertices, $\mathcal{V}$, are grouped into disjoint subsets, $\mathcal{E}$ and $\mathcal{H}$, with $\mathcal{E} \cup \mathcal{H} = \mathcal{V}$. The aim is to ensure that every cut of the sparsified graph, $\mathcal{G}'$, with the same set of vertices, $\mathcal{V}$, and the new edges with weighs W'_{mn}, denoted by

$$Cut_{\mathcal{G}'}(\mathcal{E}, \mathcal{H}) = \sum_{\substack{m \in \mathcal{E} \\ n \in \mathcal{H}}} W'_{mn},$$

is close to the corresponding cut of the original graph, that is

$$(1 - \epsilon) Cut_{\mathcal{G}}(\mathcal{E}, \mathcal{H}) \leq Cut_{\mathcal{G}'}(\mathcal{E}, \mathcal{H}) \leq (1 + \epsilon) Cut_{\mathcal{G}}(\mathcal{E}, \mathcal{H}), \tag{5.1}$$

where ϵ is sufficiently small.

To this end, random edge selection is achieved in the following way:

- every edge is kept in the new graph, $\mathcal{G}'$, with an assumed probability p;

- the weight of the edge which is kept in the new graph, $\mathcal{G}'$, is changed from 1 to $1/p$.

In this way, the number of edges, N_e, in the original graph, $\mathcal{G}$, is reduced to the expected number of edges equal to pN_e.

The inequality in (5.1) is satisfied with a certain probability, for a given ϵ. Consider an undirected and unweighted graph, with M edges in one cut. Every edge in this cut is either removed (with probability $(1-p)$) or kept with probability p. If the edge is kept, its weight assumes the value $1/p$. For the M edges in a considered cut, the probability that k of M edges will be kept is equal to

$$P_k = \binom{M}{k} p^k (1-p)^{M-k}.$$

The resulting value of the new cut is a random variable with Bernoulli distribution, given by

$$P\left(Cut_{\mathcal{G}'}(\mathcal{E},\mathcal{H}) = k\frac{1}{p}\right) = \frac{k}{p}\binom{M}{k} p^k (1-p)^{M-k}.$$

The mean value of this cut is

$$E\{Cut_{\mathcal{G}'}(\mathcal{E},\mathcal{H})\} = \frac{1}{p}(pM) = M,$$

while the variance of this Bernoulli distributed random variable is

$$\mathrm{Var}\{Cut_{\mathcal{G}'}(\mathcal{E},\mathcal{H})\} = \frac{1}{p^2}p(1-p)M = \frac{1-p}{p}M.$$

Having in mind that, for a large M, the Bernoulli distribution approaches the Gaussian distribution we can conclude that the relation is satisfied with a probability of 0.95 for $\epsilon = 2\sqrt{(1-p)/(pM)}$, according to the two-sigma rule for the Gaussian distribution and after the normalization with $Cut_{\mathcal{G}}(\mathcal{E},\mathcal{H}) = M$.

For cuts with a very small number of edges, if all edges are sampled with the same probability, p, there is a significant probability that the minimum cut would be destroyed by removing all edges in this cut. A way to overcome this problem is to slightly adapt the selection procedure, so that the minimum cut is always kept, and the other edges

are sparsified in the usual way. This simple scheme therefore adopts the probability of removing the edges related to the number of the edges in a cut. For example, in cuts with a small number of edges, the probability of removing the edges should be very small.

5.2.2 Spectral Graph Sparsification

Recent research efforts on *spectral graph sparsification* focus on definition of subgraphs or sparsifiers that can robustly preserve the spectrum (eigenvalues and eigenvectors) of the original graph Laplacian (Imre *et al.*, 2020).

The criterion for spectral similarity of two graphs is based on the quadratic Laplacian form

$$\mathbf{x}^T\mathbf{L}\mathbf{x} = \frac{1}{2}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1} W_{mn}(x(m) - x(n))^2, \tag{5.2}$$

where $\mathbf{x}$ is an arbitrary vector with N elements.

Definition: The graphs, $\mathcal{G}$ and $\mathcal{G}'$, with respective graph Laplacians, $\mathbf{L}$ and $\mathbf{L}'$, are σ-spectrally similar if their quadratic forms satisfy

$$\frac{1}{\sigma}\mathbf{x}^T\mathbf{L}'\mathbf{x} \leq \mathbf{x}^T\mathbf{L}\mathbf{x} \leq \sigma\mathbf{x}^T\mathbf{L}'\mathbf{x}. \tag{5.3}$$

The quality of the sparsification can be evaluated through the condition number, $\lambda_{\max}/\lambda_{\min}$, of the generalized eigenvalue relation

$$\mathbf{L}\mathbf{u} = \lambda\mathbf{L}'\mathbf{u} \tag{5.4}$$

with the constant, σ, satisfying the relation, $\sigma^2 \geq \lambda_{\max}/\lambda_{\min}$, where $\lambda_{\max}$ and $\lambda_{\min}$ are respectively the maximum and minimum generalized eigenvalue of (5.4). A smaller σ ($\sigma \approx 1$ or $\lambda_{\max} \approx \lambda_{\min}$) indicates higher spectral similarity.

The state-of-art techniques in this area employ an analogy with effective resistances in circuit theory (Spielman and Srivastava, 2011). The underpinning idea is as follows; a graph $\mathcal{G}$ with N vertices can be considered as a resistive network with resistances $R_{mn} = 1/W_{mn}$ between the vertices m and n, which are connected by an edge (more detail on the equivalence between a general graph and the resistive

network is given in Part III). For any two vertices, m and n, that are connected by an edge, the effective resistance can be calculated in several ways: (1) Using transformations of the corresponding electrical circuit (including the so-called star-mesh transformations); (2) Injecting unit current into the vertex, m, and taking the same current out from the vertex, n. The effective resistance is then equal to the difference of potentials in the vertices m and n; and (3) Through the eigenvalue (spectral) decomposition of the corresponding graph. The effective resistance, $R_{\mathit{eff}}(m,n)$, is then obtained from (4.37) as

$$CT(m,n) = V_{\mathcal{V}}\|\mathbf{q}_m - \mathbf{q}_n\|_2^2 = V_{\mathcal{V}} \sum_{i=1}^{N-1} (q_i(m) - q_i(n))^2 = V_{\mathcal{V}} R_{eff}(m,n), \tag{5.5}$$

where $R_{\mathit{eff}}(m,n)$ denotes the effective resistance between vertices m and n (this relation will be proven in Part III), and is given by

$$R_{\mathit{eff}}(m,n) = \sum_{i=1}^{N-1} (q_i(m) - q_i(n))^2 = \|\mathbf{q}_m - \mathbf{q}_n\|_2^2. \tag{5.6}$$

Spectral graph sparsification can now be thought of as a process of sampling edges from the graph, with probabilities of keeping edges proportional to their effective resistances. This approach rests upon the observation that if the effective resistance is small with respect to the resistance of the edge directly connecting the vertices, m and n, then these two vertices are well connected via other edges and the considered direct edge can be removed without significant influence on the whole graph. In turn, upon this edge is removed, from (5.2) we see that the total dissipated energy in the circuit corresponding to graph $\mathcal{G}$ will not change significantly, and will remain close to the energy in the electric network corresponding to the new pruned graph.

Note that if the effective resistance is close to $R_{\mathit{eff}}(m,n) \approx 1/W_{mn}$, then the other network connections are weak and the considered edge should be kept.

Such a simplified pruning algorithm can be implemented as follows.

- For the considered graph, find the graph Laplacian, $\mathbf{L}$.
- Calculate the eigenvectors, $\mathbf{u}_k$, and the eigenvalues, λ_k, of the graph Laplacian, $k = 0, 1, 2, \ldots, N-1$.

- Form the commute time spectral vectors with elements, $q_k(n) = u_k(n)/\sqrt{\lambda_k}$, for $k = 1, 2, \ldots, N-1$.
- Find the effective resistances, $R_{\mathit{eff}}(m,n) = \sum_{k=1}^{N-1}(q_k(m) - q_k(n))^2$.
- For every pair of vertices, m and n, connected by an edge, use $R_{\mathit{eff}}(m,n)$ as a measure for the probability that the considered edge should be kept in the graph.

  The effective conductance (inverse to the effective resistance) between the vertices, m and n, is equal to

$$\frac{1}{R_{\mathit{eff}}(m,n)} = W_{mn} + C_{mn},$$

  where $C_{mn} = \frac{1}{R_{\mathit{eff}}(m,n)} - W_{mn}$ is the effective conductance between m and n due to all other connections, except for the direct one defined by W_{mn}. Its relative value, normalized by W_{mn}, is given by

$$\frac{C_{mn}}{W_{mn}} = \frac{1}{W_{mn}R_{\mathit{eff}}(m,n)} - 1.$$

  We can now state that the influence of indirect connections between the vertices, m and n, is significant with respect to the existing direct connection, if

$$\frac{1}{W_{mn}R_{\mathit{eff}}(m,n)} - 1 \gg 1.$$

  The probability of keeping the edge (m,n) becomes

$$P_{mn} = \frac{W_{mn}}{\frac{1}{R_{\mathit{eff}}(m,n)}} = W_{mn}R_{\mathit{eff}}(m,n).$$

  If there are no indirect connections between m and n, then $1/R_{\mathit{eff}} = W_{mn}$ and the edge (m,n) must be kept with probability $P_{mn} = 1$. In general, $1/R_{\mathit{eff}} \geq W_{mn}$ holds. By increasing the number of indirect connections, $1/R_{\mathit{eff}}$ becomes increasingly larger than W_{mn} (*cf.* R_{eff} increasingly smaller than $1/W_{mn}$), thus indicating that the probability of keeping this edge should be decreasing.

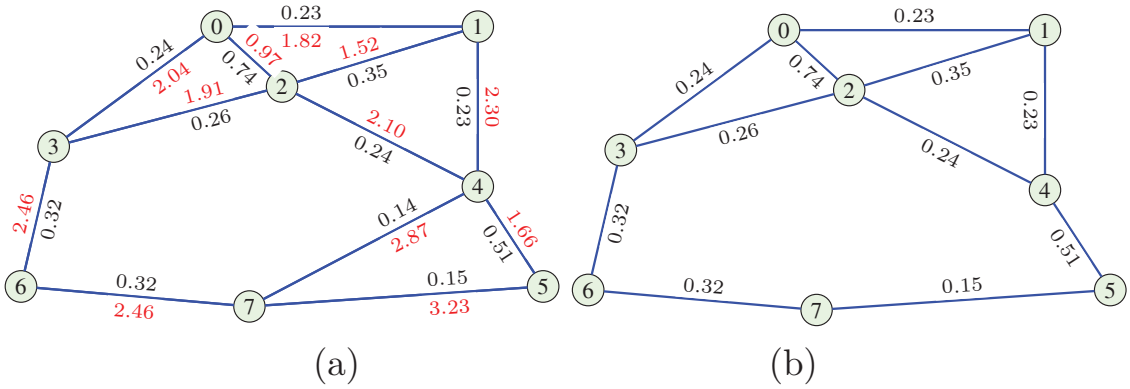


Figure 5.1: Principle of spectral graph sparsification. (a) The graph from Figure 2.2 with the edge weights, W_{mn}, and the effective resistances, $R_{eff}(m,n)$, for each pair of connected vertices. (b) The pruned graph from (a) whereby the edge $(4,7)$, characterized by the minimum value of $W_{mn}R_{eff}(m,n)$, is removed.

Example 36: Consider the graph from Figure 2.2. The effective resistances are calculated using $R_{eff}(m,n) = \sum_{k=1}^{N-1}(q_k(m) - q_k(n))^2$, with the spectral vectors calculated using the graph Laplacian eigenvectors as $q_k(n) = u_k(n)/\sqrt{\lambda_k}$, for $k = 1, 2, \ldots, N-1$. The values of effective resistances are given in red in Figure 5.1(a). When these resistances are multiplied by the corresponding edge weights, the lowest product is obtained for $W_{47}R_{eff}(4,7) = 2.87 \cdot 0.14 = 0.40$. Therefore, this is the candidate for an edge with the lowest probability of being kept, and the best candidate for pruning. The worst candidate for pruning would be $W_{45}R_{eff}(4,5) = 0.85$. After pruning the edge $(4,7)$, the pruned graph, $\mathcal{G}_P$, is shown in Figure 5.1(b). Another common criterion for pruning suggests that the edge with the smallest effective resistance should be pruned; this criterion would suggest to prune the edge $(0,2)$.

In that case, the spectral distance between the original and pruned graphs becomes (Jovanović and Stanić, 2012)

$$SD(\mathcal{G}, \mathcal{G}_P) = \sum_{k=0}^{N-1} |\lambda_k - \lambda_k^P| = 0.28.$$

Note that if the "worst" edge $(4,5)$, with the maximum value of $W_{mn}R_{eff}(m,n)$ is pruned, then the spectral distance becomes $SD(\mathcal{G}, \mathcal{G}_P) = 1.02$.

A main obstacle for using spectral sparsification is that for large graphs it is computationally very demanding, as the estimation of edge

effective resistances requires computing the eigenvectors and eigenvalues of the graph Laplacian. This topic is currently under intensive investigation.

5.2.3 Uniform Graph Sparsifier

This sparsification strategy randomly selects M edges, with replacement, with probabilities proportional to their weights (Sadhanala *et al.*, 2016). The sparsified graph is then formed using the same vertices, but with the selected edges having equal weights, that is

$$W'_{mn} = \frac{1}{2M} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} W_{mn} = \frac{W}{2M}.$$

The so produced random graph maintains its expected energy equal to the energy in the original graph, and for any $x(n)$, that is

$$\mathrm{E}\{\mathbf{x}^T \mathbf{L}' \mathbf{x}\} = \mathbf{x}^T \mathbf{L} \mathbf{x}. \tag{5.7}$$

To prove this, note that the number of times, $N_{W_{mn}}$, that an edge between vertices m and n, with the corresponding weight W_{mn}, is selected, is equal to $E\{N_{W_{mn}}\} = W_{mn} 2M/W$. Then, the expected value of the weight W'_{mn} is $E\{W'_{mn}\} = E\{N_{W_{mn}} \frac{W}{2M}\}$, which gives $E\{W'_{mn}\} = W_{mn}$, and $E\{\mathbf{L}'\} = \mathbf{L}$.

5.3 Graph Coarsening

We have so far addressed graph sparsification based on a reduction in the number of edges, while the number of vertices remained unaltered. Note that the number of vertices defines the size and dimensionality of the graph, with the analysis quickly becoming computationally prohibitive for large graphs. Graph coarsening belongs to graph down-sampling strategies and refers to the reduction in the number of vertices of the original graph. Graph coarsening is typically used in graph partitioning and for the visualization of large graphs in a computationally efficient manner (Tremblay and Loukas, 2020). In general, it can be performed by grouping the vertices into $N_c < N$ groups, subsequently forming new vertices, and finally connecting these new vertices (former groups

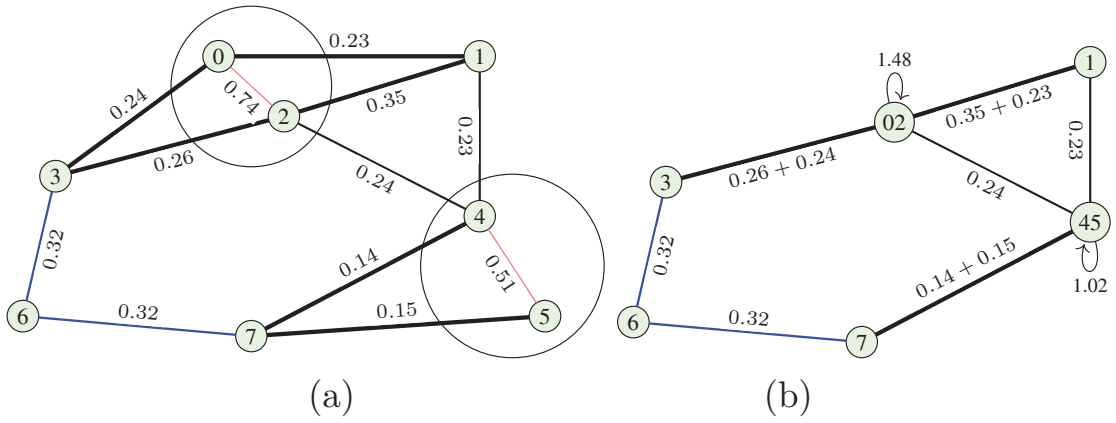


Figure 5.2: Graph coarsening. (a) The original graph from Figure 2.2 with the edges $(0, 2)$ and $(4, 5)$ used for vertex merging and forming "super-vertices" designated by circles. (b) The coarsened graph with a reduced number of vertices, obtained by forming two "super-vertices" 02 and 45. The resulting edge weights are obtained by summing up all corresponding edge weights belonging to the "super-vertices".

of vertices) with the "equivalent weights", which represent a sum of all weights between the groups. Groups of vertices are formed using the matching in graphs (explained below).

Example 37: Consider the graph $\mathcal{G}$ from Figure 2.2. To form a coarsened version, $\mathcal{G}_c$, of this graph, which has a reduced number of vertices, we shall first form two "super-vertices". For example, the "super-vertices" 02 and 45 can be formed respectively from the vertices 0 and 2 and vertices 4 and 5, as in Figure 5.2(a). The "super-edges" connecting these super-vertices are obtained as cumulative values for the vertex edges forming the new "super-vertices". The weight matrix of this coarsened graph is of dimension $N_c = 6$, and is given by

$$\mathbf{W}_c = \begin{array}{c} \\ 02 \\ 1 \\ 3 \\ 45 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{cccccc} 02 & 1 & 3 & 45 & 6 & 7 \end{array} \\ \begin{bmatrix} 1.48 & 0.58 & 0.50 & 0.24 & 0 & 0 \\ 0.58 & 0 & 0 & 0.23 & 0 & 0 \\ 0.50 & 0 & 0 & 0 & 0.32 & 0 \\ 0.24 & 0.23 & 0 & 1.02 & 0 & 0.29 \\ 0 & 0 & 0.32 & 0 & 0 & 0.32 \\ 0 & 0 & 0 & 0.29 & 0.32 & 0 \end{bmatrix} \end{array}, \tag{5.8}$$

with the "super-vertices" exhibiting self-loops with the weights equal to double the value of the removed edge (edge within the "super-vertex"). In some applications, the self-loops are filtered-out (removed).

The new, reduced-dimension weight matrix, $\mathbf{W}_c$, of the coarsened graph could be alternatively obtained using the "super-vertex" indicator matrix $\mathbf{P}$, whose elements are 1 if the vertex in the original graph $\mathcal{G}$ belongs to the considered "super-vertex" and zero elsewhere, that is

$$\mathbf{P} = \begin{array}{c} 02 \\ 1 \\ 3 \\ 45 \\ 6 \\ 7 \end{array} \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{array}{c} \\ \\ \\ \\ \\ \\ \end{array} \cdot \tag{5.9}$$
$$\begin{array}{cccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array}$$

The relation between the weight matrix of the coarse graph, $\mathbf{W}_c$, and that of the original graph, $\mathbf{W}$, is therefore

$$\mathbf{W}_c = \mathbf{P}\mathbf{W}\mathbf{P}^T,$$

where $\mathbf{W}$ is defined in (2.4) and the resulting coarsened graphs is shown in Figure 5.2(b).

Graph lifting (uncoarsening). Graph lifting is an inverse operation to graph coarsening, and represents a process of obtaining a larger scale (fine) graph from a coarsened (smaller) graph. The weight matrix, $\mathbf{W}_L$, of the lifted graph is obtained from the weight matrix of the coarsened graph, $\mathbf{W}_c$, as

$$\mathbf{W}_L = \mathbf{P}^+\mathbf{W}_c(\mathbf{P}^+)^T,$$

where $\mathbf{P}^+$ is the pseudo-inverse of the indicator matrix, such that $\mathbf{P}\mathbf{P}^+ = \mathbf{I}$, where $\mathbf{I}$ is the identity matrix.

For the considered example, the lifted weight matrix is

$$\mathbf{W}_L = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{cccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{bmatrix} 0.37 & 0.29 & 0.37 & 0.25 & 0.06 & 0.06 & 0 & 0 \\ 0.29 & 0 & 0.29 & 0 & 0.11 & 0.11 & 0 & 0 \\ 0.37 & 0.29 & 0.37 & 0.25 & 0.06 & 0.06 & 0 & 0 \\ 0.25 & 0 & 0.25 & 0 & 0 & 0 & 0.32 & 0 \\ 0.06 & 0.11 & 0.06 & 0 & 0.25 & 0.25 & 0 & 0.14 \\ 0.06 & 0.11 & 0.06 & 0 & 0.25 & 0.25 & 0 & 0.14 \\ 0 & 0 & 0 & 0.32 & 0 & 0 & 0 & 0.32 \\ 0 & 0 & 0 & 0 & 0.14 & 0.14 & 0.32 & 0 \end{bmatrix} \end{array}. \tag{5.10}$$

The same relations as for the weights hold for the corresponding graph Laplacian of the original graph, $\mathbf{L}$, graph Laplacian of the coarsened graph, $\mathbf{L}_c$, and the graph Laplacian of the lifted graph, $\mathbf{L}_L$, that is

$$\mathbf{L}_c = \mathbf{P}\mathbf{L}\mathbf{P}^T.$$
$$\mathbf{L}_L = \mathbf{P}^+\mathbf{L}_c(\mathbf{P}^+)^T.$$

Notice that for the normalized graph Laplacian, the definition of the indicator matrix should be slightly modified (Jin *et al.*, 2020).

It is of particular interest to consider *spectral similarity* of the original large size graph and the corresponding reduced-size graph (or a lifted graph). If spectral similarity is preserved by graph coarsening, then instead of operating on the large graph, $\mathcal{G}$, the eigendecomposition is first performed at a lower computational cost on the smaller dimensional coarsened graph ($N_c \ll N$). Then, spectral analysis is performed by lifting the graph to the original large dimensionality and refining the results (Loukas and Vandergheynst, 2018; Tremblay and Loukas, 2020).

Various criteria for spectral similarity exist, including an element-wise form of (5.3). We here employ as a spectral similarity metric a simple spectral distance between the original graph, $\mathcal{G}$, and the coarsened and lifted graph of the same dimension, $\mathcal{G}_L$, defined by

$$SD(\mathcal{G}, \mathcal{G}_L) = \sum_{k=0}^{N-1} |\lambda_k - \lambda_k^{cL}|,$$

where λ_k are the eigenvalues of the original graph, and λ_k^{cL} the eigenvalues of the coarsened and lifted graph. For the considered example, the spectral distance is $SD(\mathcal{G}, \mathcal{G}_L) = 1.48$.

Generalization. The process of graph coarsening may be continued until a desired number of vertices is obtained. For example, the vertex 1 and the "super-vertex" 02 can be grouped into a new "super-vertex" 102. Then, the new edges are calculated using the indicator matrix and the matrix $\mathbf{W}_c$.

In general, the coarsening involves a sequence of graphs

$$\begin{gathered}\mathcal{G} = \mathcal{G}_0 = \{\mathcal{V}, \mathcal{B}, \mathcal{W}\} = \{\mathcal{V}_0, \mathcal{B}_0, \mathcal{W}_0\} \\ \mathcal{G}_1 = \{\mathcal{V}_1, \mathcal{B}_1, \mathcal{W}_1\} \\ \vdots \\ \mathcal{G}_c = \{\mathcal{V}_c, \mathcal{B}_c, \mathcal{W}_c\},\end{gathered}$$

whereby at every iteration, the coarsened graph, $\mathcal{G}_{l+1} = \{\mathcal{V}_{l+1}, \mathcal{B}_{l+1}, \mathcal{W}_{l+1}\}$, is obtained from the previous one through a weight matrix transformation based on the corresponding indicator matrices,

$$\mathbf{W}_l = \mathbf{P}_l \mathbf{W}_{l-1} \mathbf{P}_l^T,$$

while the lifting is performed as $\mathbf{W}_{l-1} = \mathbf{P}_l^+ \mathbf{W}_l (\mathbf{P}_l^+)^T$.

Matching. In forming the "super-vertices" for graph coarsening, the notion of matching is commonly used.

Definition: A *matching* in a graph is a set of edges such that no vertex belongs to more than one edge.

For example, the edges $(0, 2)$ and $(4, 5)$ form a matching $\{02, 45\}$ in the graph from Example 37 given in Figure 5.2, while the edges $(0, 2)$ and $(2, 3)$ are not a matching, since they are both connected to vertex 2.

Definition: A matching is *maximal* if no more edges can be added to this matching.

For example, for the graph from Figure 5.2, the *maximal matching* would be the set of edges $\{02, 45, 36\}$, as no more edges can be added to this matching. However, this is not the largest possible number of

edges in a matching for this graph, and we can define a matching with a larger number of edges, like for example, the matching $\{02, 14, 57, 36\}$.

Definition: The *maximum matching* in a graph is a set of edges such that no vertex belongs to more than one edge, and another matching with a larger number of edges does not exist.

Example 38: Consider the graph, $\mathcal{G}$, from Figure 2.2. In forming a coarsened version, $\mathcal{G}_1$, of this graph we will create "super-vertices" using the maximal matching $\{02, 45, 36\}$, shown in Figure 5.3(a). The coarsened version of this graph, using the maximal matching, is given in Figure 5.3(b); this graph is coarsened again, by forming "super-vertices" 102 and 367, as in Figure 5.3(c); the final form is obtained with only two "super-vertices", as shown in Figure 5.3(d).

Notice that the edge weight in the final two-vertex graph is equal to the original graph cut for $\mathcal{E} = \{9, 1, 2, 3, 6, 7\}$ and $\mathcal{H} = \{4, 5\}$, that is, $Cut(\mathcal{E}, \mathcal{H}) = 0.23 + 0.24 + 0.14 + 0.15 = 0.76$.

This example can be repeated by using the maximum matching $\{02, 14, 57, 36\}$ in the first step.

Maximal matching strategies for graph coarsening include:

- *Random matching*, when a vertex n and one of its edges (m, n) are selected randomly. Next, another neighboring (or any other) vertex is randomly selected, together with one of its edges. The process is continued until no new edge can be added to this matching. The "super-vertices" are formed for each of the selected edges. After one coarsening level, the process can be repeated, until the desired number of vertices in a coarsened graph is reached, or a given number of levels is used.

- *Heavy edge matching* (HEM) algorithm is similar to the previous one, with the only difference in that once a vertex is randomly selected, then its edge with the maximum weight is used for the matching and "super-vertex" forming. In this way, the edges with the strongest weights are excluded, since they would probably not participate in the minimum cut, so that both the original and the coarsened graph share the same minimum cut.

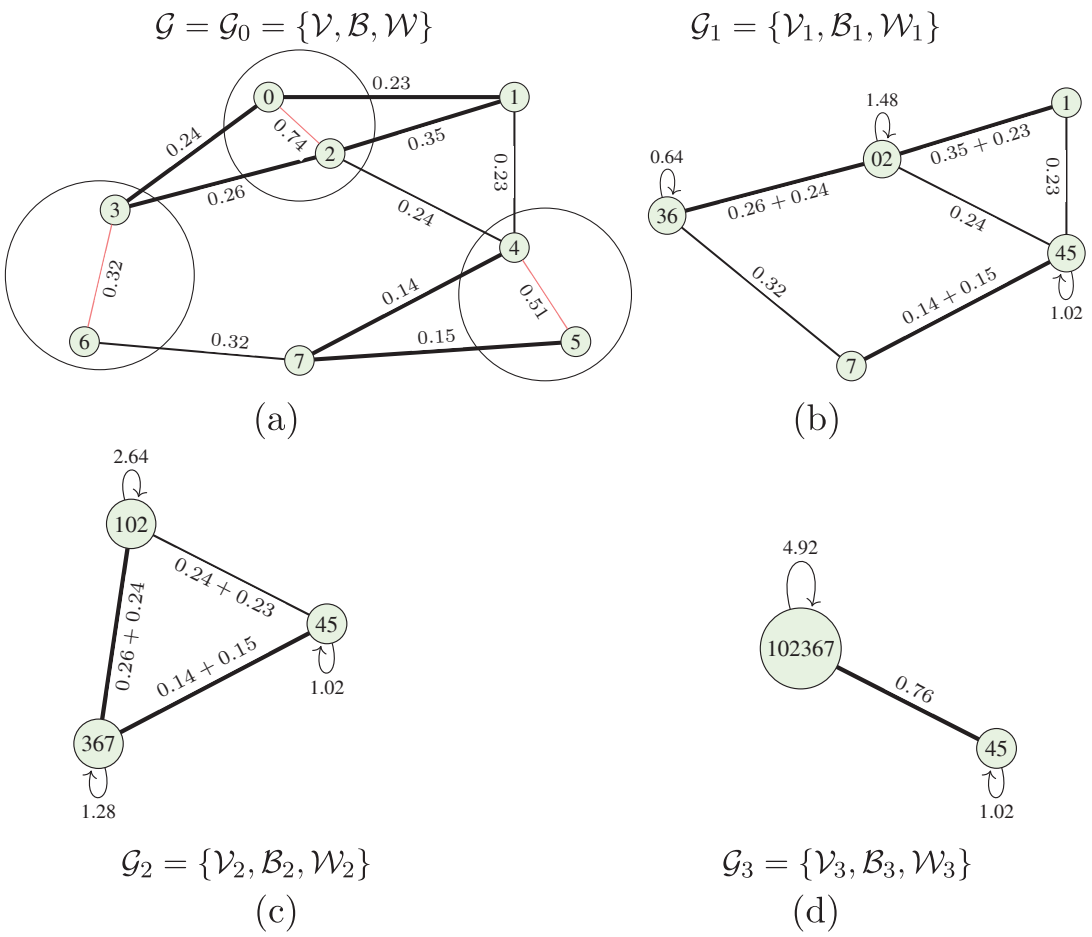


Figure 5.3: Principle of maximal matching for graph coarsening. (a) The original graph from Figure 2.2 with (b)–(d) its coarsened graphs obtained in three steps using the maximal matching, until a two-vertex graph is obtained.

- *Sorted Heavy edge matching* uses the vertices with the highest degree first, in defining the matching. Vertices with higher degrees are also preferred in the subsequent steps.

- *Edge weighted random matching* chooses an edge with a probability, P_{mn}, proportional to its weight, that is

$$P_{mn} = \frac{W_{mn}}{\frac{1}{2}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1} W_{mn}}.$$

  The edges with a higher weight are thus more likely to be selected in each step of the maximal matching procedure.

5.4 Kron Reduction of Graphs

A reduction of an electrical network via a Schur complement of the associated conductance matrix is known as the Kron reduction, due to the seminal work of Gabriel Kron. It is based on separating the vertices into two groups: active vertices and inner vertices. The inner vertices can be eliminated from the graph without changing the electric network conditions; this is achieved via equivalent transformations, such as the "star-mesh" transformations (Dorfler and Bullo, 2012). The Kron reduction of graphs is also relevant in other physical domains, including computing applications and the reduction of Markov chains. Since this approach requires quite specific physical interpretation of the active and inner vertices, it will be discussed in detail in Part III of this monograph.

6

Conclusion

Although within the graph data analytics paradigm, graphs have been present in various forms for centuries, the advantages of the graph framework for data analytics, as opposed to the optimization of the graphs themselves, but for recently has received little attention. In order to provide a comprehensive and Data Science friendly introduction to graph data analytics, an overview of graphs from this specific practitioner-friendly signal processing point of view is a prerequisite.

In this part of our tutorial, we have introduced graphs as irregular signal domains, together with their properties that are relevant for data analytics applications which rest upon the estimation of signals on graphs. This has been achieved in a systematic and example rich way and by highlighting links with classic matrix analysis and linear algebra. Spectral analysis of graphs has been elaborated upon in detail, as this is the main underpinning methodology for efficient data analysis, the ultimate goal in Data Science. Both the adjacency matrix and the Laplacian matrix have been used in this context, along with their spectral decompositions. Finally, we have highlighted important aspects of graph segmentation, Laplacian eigenmaps, graph cuts, graph sparsification and coarsening, and have emphasized their role as the

foundation for advances in Data Analytics and unsupervised learning on graphs.

Part II of this monograph will address theory and methods of processing data on graphs, while Part III is devoted to unsupervised graph topology learning, from the observed data, and Machine learning on graphs.

Appendices

A

Power Method for Eigenanalysis

Computational complexity of the eigenvalue and eigenvector calculation for a symmetric matrix is of the order of $\mathcal{O}(N^3)$, which is computationally prohibitive for very large graphs, especially when only a few the smoothest eigenvectors are needed, like in spectral graph clustering. To mitigate this computational bottleneck, an efficient iterative approach, called the Power Method, may be employed.

Consider the normalized weight matrix,

$$\mathbf{W}_N = \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2},$$

and assume that the eigenvalues of $\mathbf{W}_N$ are $|\lambda_0| > |\lambda_1| > \cdots > |\lambda_{M-1}|$, with the corresponding eigenvectors, $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_{M-1}$. Consider also an arbitrary linear combination of the eigenvectors, $\mathbf{u}_n$, through the coefficients α_n,

$$\mathbf{x} = \alpha_1\mathbf{u}_1 + \alpha_2\mathbf{u}_2 + \cdots + \alpha_{M-1}\mathbf{u}_{M-1}.$$

A further multiplication of the vector $\mathbf{x}$ by the normalized weight matrix, $\mathbf{W}_N$, results in

$$\begin{aligned}\mathbf{W}_N\mathbf{x} &= \alpha_1\mathbf{W}_N\mathbf{u}_1 + \alpha_2\mathbf{W}_N\mathbf{u}_2 + \cdots + \alpha_{M-1}\mathbf{W}_N\mathbf{u}_{M-1} \\ &= \alpha_1\lambda_1\mathbf{u}_1 + \alpha_2\lambda_2\mathbf{u}_2 + \cdots + \alpha_{M-1}\lambda_{M-1}\mathbf{u}_{M-1}.\end{aligned}$$

A repetition of this multiplication k times yields

$$\begin{aligned}\mathbf{W}_N^k\mathbf{x} &= \alpha_1\lambda_1^k\mathbf{u}_1 + \alpha_2\lambda_2^k\mathbf{u}_2 + \cdots + \alpha_{M-1}\lambda_{M-1}^k\mathbf{u}_{M-1} \\ &= \alpha_1\lambda_1^k\left(\mathbf{u}_1 + \alpha_2\frac{\lambda_2^k}{\lambda_1^k}\mathbf{u}_2 + \cdots + \alpha_{M-1}\frac{\lambda_{M-1}^k}{\lambda_1^k}\mathbf{u}_{M-1}\right) \\ &\approxeq \alpha_1\lambda_1^k\mathbf{u}_1.\end{aligned}$$

In other words, we have just calculated the first eigenvector of $\mathbf{W}_N$, given by

$$\mathbf{u}_1 = \mathbf{W}_N^k\mathbf{x}/\|\mathbf{W}_N^k\mathbf{x}\|_2$$

which are achieved through only matrix products of $\mathbf{W}_N$ and $\mathbf{x}$ (Tammen *et al.*, 2018; Trevisan, 2013). The convergence of this procedure depends on the eigenvalue ratio λ_2/λ_1, and requires that α_1 is not close to 0. Note that $\mathbf{W}_N$ is a highly sparse matrix, which significantly reduces the calculation complexity.

After the eigenvector $\mathbf{u}_1$ is obtained, the corresponding eigenvalue can be calculated as its smoothing index, $\lambda_1 = \mathbf{u}_1^T\mathbf{W}_N\mathbf{u}_1$.

After calculating $\mathbf{u}_1$ and λ_1, we can remove their contribution from the normalized weight matrix, $\mathbf{W}_N$, through deflation, as $\mathbf{W}_N \leftarrow \mathbf{W}_N - \lambda_1\mathbf{u}_1\mathbf{u}_1^T$, and then continue to calculate the next largest eigenvalue and its eigenvector, λ_2 and $\mathbf{u}_2$. This procedure can be repeated iteratively until the desired number of eigenvectors is found.

The relation of the normalized weight matrix, $\mathbf{W}_N$, with the normalized graph Laplacian, $\mathbf{L}_N$, is given by

$$\mathbf{L}_N = \mathbf{I} - \mathbf{W}_N,$$

while the relation between the eigenvalues and eigenvectors of $\mathbf{L}$ and $\mathbf{W}_N$ follows from $\mathbf{W}_N = \mathbf{U}^T\mathbf{\Lambda}\mathbf{U}$, to yield

$$\mathbf{L}_N = \mathbf{I} - \mathbf{U}^T\mathbf{\Lambda}\mathbf{U} = \mathbf{U}^T(\mathbf{I} - \mathbf{\Lambda})\mathbf{U}.$$

The eigenvalues of $\mathbf{L}_N$ and $\mathbf{W}_N$ are therefore related as $\lambda_n^{(L)} = 1 - \lambda_n$, and share the same corresponding eigenvectors, $\mathbf{u}_n$, of the normalized graph Laplacian and the normalized weight matrix. This means that $\lambda_1 = 1$ corresponds to $\lambda_0^{(L)} = 0$ and that the second largest eigenvalue of $\mathbf{W}_N$ produces the Fiedler vector of the normalized Laplacian.

Note that the second largest eigenvalue of $\mathbf{W}_N$ *is not necessarily* λ_2 *since the eigenvalues of* $\mathbf{W}_N$ *can be negative.*

Example 39: The weight matrix $\mathbf{W}$ from (2.4) is normalized by the degree matrix from (2.6) to arrive at $\mathbf{W}_N = \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$. The power algorithm is then used to calculate the four largest eigenvalues and the corresponding eigenvectors of $\mathbf{W}_N$ in 200 iterations, to give $\lambda_n \in \{1.0000, -0.7241, -0.6795, 0.6679\}$. These are very close to the four exact largest eigenvalues of $\mathbf{W}_N$, $\lambda_n \in \{1.0000, -0.7241, -0.6796, 0.6677\}$. Note that the Fiedler vector of the normalized graph Laplacian is associated with $\lambda_4 = 0.6679$ as it corresponds to the second largest eigenvalue of $\mathbf{W}_N$, when the eigenvalue signs are accounted for. Even when calculated using the approximative power method, the Fiedler vector is close to its exact value, as shown in Figure 4.8(d), with the maximum relative error of its elements being 0.016.

Notice that it is possible to calculate the Fiedler vector of a graph Laplacian even without using the weight matrix. Consider a graph whose eigenvalues of the Laplacian are $\lambda_0 = 0 > \lambda_1 > \lambda_2 > \cdots > \lambda_{N-1}$, where λ_1 corresponds to the largest value of the sequence $\lambda_0 = 0, 1/\lambda_1, 1/\lambda_2, \ldots, 1/\lambda_{N-1}$. These are also the eigenvalues of the pseudo-inverse of the graph Laplacian, $\mathbf{L}^+ = \text{pinv}(\mathbf{L})$. Now, since the pseudo-inverse of the graph Laplacian, $\mathbf{L}^+$, and the graph Laplacian, $\mathbf{L}$, have the same eigenvectors, we may apply the power method to the pseudo-inverse of the graph Laplacian, $\mathbf{L}^+$, and the eigenvector corresponding to the largest eigenvalue is the Fiedler vector.

Algorithm 2. Power Method for eigenanalysis.

Input:

- Normalized weight matrix $\mathbf{W}_N$
- Number of iterations, It
- Number of the desired largest eigenvectors, M

1: for $m = 1$ to M do
2: $\mathbf{u}_m \in \{-1, 1\}^M$, drawn randomly (uniformly)
3: for $i = 1$ to It do
4: $\mathbf{u}_m \leftarrow \mathbf{W}_N \mathbf{u}_m / ||\mathbf{W}_N \mathbf{u}_m||_2$
5: $\lambda_m \leftarrow \mathbf{u}_m^H \mathbf{W}_N \mathbf{u}_m$
6: end do
7: $\mathbf{W}_N \leftarrow \mathbf{W}_N - \lambda_m \mathbf{u}_m \mathbf{u}_m^H$
8: end do

Output:

- Largest M eigenvalues $|\lambda_0| > |\lambda_1| > \cdots > |\lambda_{M-1}|$ and the corresponding eigenvectors $\mathbf{u}_1, \ldots, \mathbf{u}_{M-1}$
- Fiedler vector of the normalized graph Laplacian is the eigenvector $\mathbf{u}_{n_1}$ of the second largest eigenvalue, λ_{n_1}, $\lambda_0 = 1 > \lambda_{n_1} > \cdots > \lambda_{n_{M-1}}$.

B

Algorithm for Graph Laplacian Eigenmaps

The algorithm for the Laplacian eigenmap and spectral clustering based on the eigenvectors of the graph Laplacian, the generalized eigenvectors of the graph Laplacian, and the eigenvectors of the normalized Laplacian, is given in the pseudo-code form in Algorithm 3.

Comments on the Algorithm: For the *normalized Laplacian*, Line 2 should be replaced by $\mathbf{L} \leftarrow \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$, while for the *generalized eigenvectors* Line 3 should be replaced by $[\mathbf{U}, \mathbf{\Lambda}] \leftarrow \text{eig}(\mathbf{L}, \mathbf{D})$, see also Table 4.1. The indicator values of vertex positions in the output graph are: $P = 0$, for the original vertex space, and $P = 1$, for the spectral vertex space. The indicator of mapping is: $Map = 1$, for the *commute time* mapping (matrix $\bar{\mathbf{\Lambda}}$ is obtained from $\mathbf{\Lambda}$, by omitting the trivial element $\lambda_0 = 0$), and $Map = 2$, for the *diffusion mapping* (in this case the generalized eigenvectors must be used in Line 3, $[\mathbf{U}, \mathbf{\Lambda}] \leftarrow \text{eig}(\mathbf{L}, \mathbf{D})$ and the diffusion step t should be given as an additional input parameter), otherwise $Map = 0$. The indicator of the eigenvectors normalization is: $S = 0$, for the case without normalization, $S = 1$, for two-norm normalization, $S = 2$, for the case of binary normalization, $S = 3$, for binary normalization with the mean as a reference, and $S = 4$, for marginal normalization. The indicator of

Algorithm 3. Graph Laplacian Based Eigenmaps.

Input:

- Vertex $\mathcal{V} = \{0, 1, \ldots, N-1\}$ positions, rows of $\mathbf{X}$
- Weight matrix $\mathbf{W}$, with elements W_{mn}
- Laplacian eigenmap dimensionality, M
- Position, mapping, normalization, and coloring indicators P, Map, S, C

1: $\mathbf{D} \leftarrow \text{diag}(D_{nn} = \sum_{m=0}^{N-1} W_{mn}, n = 0, 1, \ldots, N-1)$
2: $\mathbf{L} \leftarrow \mathbf{D} - \mathbf{W}$
3: $[\mathbf{U}, \mathbf{\Lambda}] \leftarrow \text{eig}(\mathbf{L})$
4: $u_k(n) \leftarrow U(n, k)$, for $k = 1, \ldots, M$, $n = 0, 1, \ldots, N-1$.
5: $\mathbf{M} \leftarrow \max_n(U(n, 1{:}L))$, $\mathbf{m} \leftarrow \min_n(U(n, 1{:}L))$
6: $\mathbf{q}_n \leftarrow [u_1(n), u_2(n), \ldots, u_L(n)]$, for all n
7: **If** Map=1, $\mathbf{q}_n \leftarrow \mathbf{q}_n \bar{\mathbf{\Lambda}}^{-1/2}$, **end**
8: **If** Map=2, $\mathbf{q}_n \leftarrow \mathbf{q}_n (\mathbf{I} - \bar{\mathbf{\Lambda}})^t$, **end**
9:
$$\mathbf{y}_n \leftarrow \begin{cases} \mathbf{q}_n, & \text{for } S = 0, \\ \mathbf{q}_n / \|\mathbf{q}_n\|_2, & \text{for } S = 1, \\ \text{sign}(\mathbf{q}_n), & \text{for } S = 2, \\ \text{sign}(\mathbf{q}_n - (\mathbf{M} + \mathbf{m})/2), & \text{for } S = 3, \\ (\mathbf{q}_n - \mathbf{m})./(\mathbf{M} - \mathbf{m}), & \text{for } S = 4 \end{cases}$$
10: $\mathbf{Y} \leftarrow \mathbf{y}_n$, as the rows of $\mathbf{Y}$
11:
$$\mathbf{Z} \leftarrow \begin{cases} \mathbf{X}, & \text{for } P = 0, \\ \mathbf{Y}, & \text{for } P = 1 \end{cases}$$
12:
$$\textbf{ColorMap} \leftarrow \begin{cases} \textbf{Constant}, & \text{for } C = 0, \\ (\mathbf{Y} + 1)/2, & \text{for } C = 1 \end{cases}$$
13: GraphPlot($\mathbf{W}, \mathbf{Z}, \textbf{ColorMap}$)
14: Cluster the vertices according to $\mathbf{Y}$ and refine using the k-means algorithm (Remark 30) or the ratio cut recalculation algorithm (Remark 33).

Output:

- New graph
- Subsets of vertex clusters

vertex coloring is: $C = 0$, for the same color for all vertices is used, and $C = 1$, when the spectral vector defines the vertex colors.

C

Other Graph Laplacian Forms

We here review some other forms of the graph Laplacian, including the Laplacian for directed graphs, graph Laplacian for the graphs with negative weights, and graph p-Laplacian.

C.1 Graph Laplacian for Directed Graphs

Directed graphs are typically analyzed based on the adjacency matrix and its spectrum. It is important to notice that the difference between a constant vector $\mathbf{x}$ and a vector $\mathbf{Ax}$ does not result in a zero-valued vector; this is because, in general, the solution to the eigenvalue relation, $\mathbf{Au} = \lambda\mathbf{u}$, is not a vector with constant elements. The value $\mathbf{Ax}$ and the difference $\mathbf{x} - \mathbf{Ax}/\lambda_{\max}$ will be used in Part II to define the shift on a graph. In order to introduce an operator on a directed graph which will restore the property of zero-valued total-variation for constant vectors, the Laplacian for directed graphs was introduced in Singh *et al.* (2016) and further analyzed in Sardellitti *et al.* (2017).

Since each edge in a directed graph connects one outgoing and one incoming vertex, in order to avoid ambiguity, the edges will be assigned

to the incoming vertex. The in-degree is then calculated as

$$D^{in}(m,m) = \sum_{n=0}^{N-1} W_{mn},$$

with the Laplacian of a directed graph defined by

$$\mathbf{L} = \mathbf{D}^{in} - \mathbf{W}.$$

In this way, the sum of each row in the Laplacian of a directed graph is zero-valued, meaning that any constant vector is also an eigenvector, with the corresponding $\lambda = 0$. Spectral analysis is then performed using the eigendecomposition of $\mathbf{L}$, and since in this case $\mathbf{L}$ is not symmetric, the eigenvalues may be complex-valued.

The total variation of a vector, $\mathbf{x}$, for a shift operator, $\mathbf{S}$, is defined by

$$\|\mathbf{x} - \mathbf{S}\mathbf{x}\|_1,$$

using the L_1 norm or

$$\|\mathbf{x} - \mathbf{S}\mathbf{x}\|_2,$$

using the L_2 norm. If the operator $\mathbf{S}$ is defined as $\mathbf{S} = \mathbf{I} - \mathbf{L}$, where $\mathbf{I}$ is the identity matrix, then

$$\|\mathbf{x} - \mathbf{S}\mathbf{x}\|_2^2 = \|\mathbf{L}\mathbf{x}\|_2^2 = (\mathbf{L}\mathbf{x})^T\mathbf{L}\mathbf{x} = \mathbf{x}^T\mathbf{L}^T\mathbf{L}\mathbf{x}.$$

If the vector $\mathbf{x}$ is an eigenvector, $\mathbf{x} = \mathbf{u}_k$, then

$$\|\mathbf{u}_k - \mathbf{S}\mathbf{u}_k\|_2^2 = \mathbf{u}_k^T\mathbf{L}^T\mathbf{L}\mathbf{u}_k = |\lambda_k|^2,$$

which indicates that the smoothness of an eigenvector is proportional to $|\lambda_k|^2$. Therefore, in analogy to frequency in classical signal analysis it can be used as an indicator of the variation of an eigenvector.

Example 40: For the directed graph from Figure 2.1, the adjacency matrix is given by (2.2) and the in-degree matrix by

$$\mathbf{D}^{in} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix}, \tag{C.1}$$

with the corresponding graph Laplacian

$$\mathbf{L} = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 4 & -1 & -1 & 0 & 0 & -1 \\ -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & -1 & 0 & 3 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 2 & -1 \\ 0 & 0 & -1 & 0 & 0 & 0 & -1 & 2 \end{bmatrix}. \tag{C.2}$$

Remark 37: Graph Laplacian of an undirected graph is a special case of the graph Laplacian of a directed graph, with each undirected edge being a combination of an incoming and an outgoing edge of the same weight.

C.2 Signed Graphs and Signed Graph Laplacian

Graphs for which edge weights may assume both positive and negative values are called *signed graphs*, and were introduced in Harary (1953), where the authors motivated graphs with weights $\{1, 0, -1\}$ through the modeling of social relations such as like, indifference, and dislike.

The vertex degree in a signed graph is defined as a sum of the absolute values of its weights (Hou, 2005), that is

$$D_a(m,m) = \sum_{n=0}^{N-1} |W_{mn}| = \sum_{n=0}^{N-1} W_{mn}\text{sign}(W_{mn}).$$

The corresponding signed graph Laplacian then becomes

$$\mathbf{L}_a = \mathbf{D}_a - \mathbf{W},$$

with the quadratic form of the Laplacian of a signed graph given by

$$\mathbf{x}^T\mathbf{L}_a\mathbf{x} = \frac{1}{2}\sum_{m=0}^{N-1}\sum_{m=0}^{N-1} |W_{mn}|(x(m) - \text{sign}(W_{mn})x(n)).$$

Notice that the signed graph Laplacian is positive-semidefinite.

Cut of a signed graph. The cut of a signed graph represents a sum of all absolute weights that correspond to the edges which connect the vertices between the subsets, $\mathcal{E}$ and $\mathcal{H}$, that is

$$Cut(\mathcal{E},\mathcal{H}) = \sum_{\substack{m\in\mathcal{E}\\ n\in\mathcal{H}}} |W_{mn}|.$$

All tools for the analysis of standard graphs can also be applied to signed graphs.

Notice that since the signed graph Laplacian may be positive definite, it then follows that a constant vector (with a zero eigenvalue) may not represent an eigenvector of the signed Laplacian. The concept of balanced graphs is introduced to deal with this issue, whereby a graph is said to be balanced if there exists a partition of its vertices into two disjoint subsets, $\mathcal{E}$ and $\mathcal{H}$, such that all positive edges reside within either $\mathcal{E}$ or $\mathcal{H}$, while all negative edges connect the vertices between $\mathcal{E}$ or $\mathcal{H}$. Then, the signed Laplacian, $\mathbf{L}_a$, of a connected signed graph is positive definite iff the graph is not balanced (Harary, 1953).

C.3 Graph p-Laplacian

A generalization of the graph Laplacian, called the p-Laplacian, and denoted by $\mathbf{L}_p$, is obtained from the generalization of the quadratic Laplacian form as (Bühler and Hein, 2009)

$$\mathbf{x}^T\mathbf{L}_p\mathbf{x} = \frac{1}{2}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1} W_{mn}|x(n)-x(m)|^p. \tag{C.3}$$

Obviously, for $p=2$, the quadratic form of standard graph Laplacian, $\mathbf{L}$, is obtained. The elements of $\mathbf{L}_p\mathbf{x}$, denoted by $\mathcal{L}_x^p(n)$, that satisfy (C.3) are defined as

$$\mathcal{L}_x^p(n) = \sum_{m=0}^{N-1} W_{mn}|x(n)-x(m)|^{p-1}\text{sign}(x(n)-x(m))$$

and it can be straightforwardly verified that the inner product of $x(n)$ and $\mathcal{L}_x^p(n)$ produces (C.3).

According to (4.12), the ratio graph cut, $CutN(\mathcal{E}, \mathcal{H})$, can be obtained by solving the minimization problem

$$CutN(\mathcal{E}, \mathcal{H}) = \min_{\mathcal{E} \subset \mathcal{V}} \left\{ \frac{\mathbf{x}^T \mathbf{L} \mathbf{x}}{\mathbf{x}^T \mathbf{x}} \right\} \tag{C.4}$$

with $\mathbf{x} = \mathbf{u}_1$. Therefore, the eigenvector $\mathbf{u}_1$ can be considered as a (non-constant) solution to the minimization problem in (C.4).

Similarly, the minimization problem for the p-Laplacian becomes

$$\min_{\mathcal{E} \subset \mathcal{V}} \left\{ \frac{\mathbf{x}^T \mathbf{L}_p \mathbf{x}}{\min_c \|\mathbf{x} - c\|_p^p} \right\}. \tag{C.5}$$

with the solution in the form of the first eigenvector, $\mathbf{u}_1^{(p)}$, comprising the elements, $v_1^{(p)}(n)$, of the p-Laplacian (Bühler and Hein, 2009)

$$\mathcal{L}^p_{v_1^{(p)}}(n) = \lambda_1^{(p)} |v_1^{(p)}(n)|^{p-1} \text{sign}(v_1^{(p)}(n)).$$

Notice that for $\mathbf{x} = \mathbf{v}_1^{(p)}$ the minimum value of (C.5) is equal to the eigenvalue, $\lambda_1^{(p)}$.

The Cheeger ratio cut, $\phi(\mathcal{V})$, with the p-Laplacian exhibits the following general bounds

$$\left(\frac{2}{\max_i d_i} \right)^{p-1} \left(\frac{\phi(\mathcal{V})}{p} \right)^p \leq \lambda_1^{(p)} \leq 2^{p-1} \phi(\mathcal{V}) \tag{C.6}$$

$$\frac{\phi(\mathcal{V})}{\max_i d_i} \leq \frac{\phi^*(\mathcal{V})}{\max_i d_i} \leq p \left(\frac{\phi(\mathcal{V})}{\max_i d_i} \right)^{1/p}, \tag{C.7}$$

where d_i is the degree of vertex i and $\phi^*(\mathcal{V})$ is the minimum Cheeger's ratio cut obtained by an optimal thresholding of the eigenvector $\mathbf{u}_1$ with a threshold t, that is, a vertex n belongs to the subset of vertices $\mathcal{E}$ if $u_1(n) > t$.

The above inequality implies that the bounds are tight for $p \to 1$, which indicates that the 1-Laplacian based cut is equivalent to the Cheeger ratio cut; this may be used to improve the cut performance in practical applications. Still, the main problem remains in the computational issues related to calculation of the p-Laplacian eigenvectors, especially for $p \to 1$ (Bühler and Hein, 2009; Chang, 2016; Chang *et al.*, 2016).

Acknowledgments

We wish to express our sincere gratitude to Yao Lei Xu, Kriton Konstantinidis, Shota Saito, and Giacomo Kahn whose thorough proofreading and deep insight have been of great help at various stages of manuscript preparation.

References

Afrati, F. and A. G. Constantinides (1978). "The use of graph theory in binary block code construction". In: *Proceedings of the International Conference on Digital Signal Processing*. 228–233.

Alon, N. (1986). "Eigenvalues and expanders". *Combinatorica*. 6(2): 83–96.

Bapat, R. (1996). "The Laplacian matrix of a graph". *Mathematics Student-India*. 65(1): 214–223.

Barik, S., R. B. Bapat, and S. Pati (2015). "On the Laplacian spectra of product graphs". *Applicable Analysis and Discrete Mathematics*. 9(1): 39–58.

Belkin, M. and P. Niyogi (2003). "Laplacian eigenmaps for dimensionality reduction and data representation". *Neural Computation*. 15(6): 1373–1396.

Brouwer, A. E. and W. H. Haemers (2012). *Spectra of Graphs*. New York: Springer-Verlag.

Bühler, T. and M. Hein (2009). "Spectral clustering based on the graph p-Laplacian". In: *Proceedings of the 26th ACM Annual International Conference on Machine Learning*. 81–88.

Bunse-Gerstner, A. and W. B. Gragg (1988). "Singular value decompositions of complex symmetric matrices". *Journal of Computational and Applied Mathematics*. 21(1): 41–54.

Chandra, A. K., P. Raghavan, W. L. Ruzzo, R. Smolensky, and P. Tiwari (1996). "The electrical resistance of a graph captures its commute and cover times". *Computational Complexity*. 6(4): 312–340.

Chang, K. C. (2016). "Spectrum of the 1-Laplacian and Cheeger's constant on graphs". *J. Graph Theory*. 81(2): 167–207.

Chang, K., S. Shao, and D. Zhang (2016). "The 1-Laplacian Cheeger cut: Theory and algorithms". *arXiv preprint arXiv:1603.01687*.

Chen, S., A. Sandryhaila, J. M. Moura, and J. Kovačević (2014). "Signal denoising on graphs via graph filtering". In: *Proc. 2014 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. 872–876.

Christofides, N. (1975). *Graph Theory: An Algorithmic Approach*. Academic Press.

Chung, F. (1997). *Spectral Graph Theory*. Providence, RI: AMS.

Chung, F. (2005). "Laplacians and the Cheeger inequality for directed graphs". *Annals of Combinatorics*. 9(1): 1–19.

Chung, F. (2007). "Four proofs for the Cheeger inequality and graph partition algorithms". In: *Proceedings of ICCM*. Vol. 2. 378.

Chung, F. R. and R. P. Langlands (1996). "A combinatorial Laplacian with vertex weights". *Journal of Combinatorial Theory, Series A*. 75(2): 316–327.

Coifman, R. R. and S. Lafon (2006). "Diffusion maps". *Applied and Computational Harmonic Analysis*. 21(1): 5–30.

Cvetković, D. M. and M. Doob (1985). "Developments in the theory of graph spectra". *Linear and Multilinear Algebra*. 18(2): 153–181.

Cvetković, D. M. and I. Gutman (2011). *Selected Topics on Applications of Graph Spectra*. Matematički Institut SANU (Serbian Academy of Scences and Arts).

Cvetković, D. M., M. Doob, and H. Sachs (1980). *Spectra of Graphs: Theory and Application*. Vol. 87. Academic Press.

Dhillon, I. S., Y. Guan, and B. Kulis (2004). "Kernel k-means: Spectral clustering and normalized cuts". In: *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 551–556.

Dong, X., P. Frossard, P. Vandergheynst, and N. Nefedov (2012). "Clustering with multi-layer graphs: A spectral perspective". *IEEE Transactions on Signal Processing*. 60(11): 5820–5831.

Dorfler, F. and F. Bullo (2012). "Kron reduction of graphs with applications to electrical networks". *IEEE Transactions on Circuits and Systems I: Regular Papers*. 60(1): 150–163.

Duncan, A. (2004). "Powers of the adjacency matrix and the walk matrix". *The Collection*: 1–11.

Ekambaram, V. N. (2014). *Graph-Structured Data Viewed Through a Fourier Lens*. Berkeley: University of California.

Fiedler, M. (1973). "Algebraic connectivity of graphs". *Czechoslovak Mathematical Journal*. 23(2): 298–305.

Fujiwara, K. (1995). "Eigenvalues of Laplacians on a closed Riemannian manifold and its nets". *Proceedings of the American Mathematical Society*. 123(8): 2585–2594.

Gavili, A. and X.-P. Zhang (2017). "On the shift operator, graph frequency, and optimal filtering in graph signal processing". *IEEE Transactions on Signal Processing*. 65(23): 6303–6318.

Grady, L. J. and J. R. Polimeni (2010). *Discrete Calculus: Applied Analysis on Graphs for Computational Science*. Springer Science & Business Media.

Grebenkov, D. S. and B.-T. Nguyen (2013). "Geometrical structure of Laplacian eigenfunctions". *SIAM Review*. 55(4): 601–667.

Hagen, L. and A. B. Kahng (1992). "New spectral methods for ratio cut partitioning and clustering". *IEEE Trans. Computer-Aided Design of Int. Circuits and Systems*. 11(9): 1074–1085.

Hamon, R., P. Borgnat, P. Flandrin, and C. Robardet (2016a). "Extraction of temporal network structures from graph-based signals". *IEEE Transactions on Signal and Information Processing over Networks*. 2(2): 215–226.

Hamon, R., P. Borgnat, P. Flandrin, and C. Robardet (2016b). "Relabelling vertices according to the network structure by minimizing the cyclic bandwidth sum". *Journal of Complex Networks*. 4(4): 534–560.

Harary, F. (1953). "On the notion of balance of a signed graph." *The Michigan Mathematical Journal*. 2(2): 143–146.

Horaud, R. (2009). "A short tutorial on graph Laplacians, Laplacian embedding, and spectral clustering". [Online], Available: URL: http://csustan.csustan.edu/~tom/Lecture-Notes/Clustering/GraphLaplacian-tutorial.pdf.

Hou, Y. P. (2005). "Bounds for the least Laplacian eigenvalue of a signed graph". *Acta Mathematica Sinica*. 21(4): 955–960.

Imre, M., J. Tao, Y. Wang, Z. Zhao, Z. Feng, and C. Wang (2020). "Spectrum-preserving sparsification for visualization of big graphs". *Computers & Graphics*. 87: 89–102.

Jain, A. K. (2010). "Data clustering: 50 years beyond K-means". *Pattern Recognition Letters*. 31(8): 651–666.

Jin, Y., A. Loukas, and J. JaJa (2020). "Graph coarsening with preserved spectral properties". In: *Proc. International Conference on Artificial Intelligence and Statistics*. 4452–4462.

Jones, O. (2013). *Spectra of Simple Graphs*. Whitman College. [Online]. Available: URL: https://www.whitman.edu/Documents/Academics/Mathematics/Jones.pdf.

Jordan, M. I. (1998). *Learning in Graphical Models*. Vol. 89. Springer Science & Business Media.

Jordan, M. I. (2004). "Graphical models". *Statistical Science*. 19(1): 140–155.

Jovanović, I. and Z. Stanić (2012). "Spectral distances of graphs". *Linear Algebra and Its Applications*. 436(5): 1425–1435.

Khuller, S. (1998). "Approximation algorithms for finding highly connected subgraphs". *Tech. Rep.*

Kleinberg, J. and E. Tardos (2006). *Algorithm Design*. Pearson Education India.

Krim, H. and A. B. Hamza (2015). *Geometric Methods in Signal and Image Analysis*. Cambridge University Press.

Kron, G. (1963). *Diakoptics: The Piecewise Solution of Large-Scale Systems*. Vol. 2. MacDonald.

Leskovec, J. and C. Faloutsos (2006). "Sampling from large graphs". In: *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 631–636.

Loukas, A. and P. Vandergheynst (2018). "Spectrally approximating large graphs with smaller graphs". *arXiv preprint arXiv:1802.07510*.

Lu, H., Z. Fu, and X. Shu (2014). "Non-negative and sparse spectral clustering". *Pattern Recognition*. 47(1): 418–426.

Maheswari, S. U. and B. Maheswari (2016). "Some properties of Cartesian product graphs of Cayley graphs with arithmetic graphs". *International Journal of Computer Applications*. 138(3): 26–29.

Malik, J., S. Belongie, T. Leung, and J. Shi (2001). "Contour and texture analysis for image segmentation". *International Journal of Computer Vision*. 43(1): 7–27.

Marques, A., A. Ribeiro, and S. Segarra (2017). "Graph signal processing: Fundamentals and applications to diffusion processes". In: *IEEE Int. Conf. on Accoustic, Speech and Signal Processing (ICASSP), Tutorial*.

Masoumi, M. and A. B. Hamza (2017). "Spectral shape classification: A deep learning approach". *Journal of Visual Communication and Image Representation*. 43: 198–211.

Masoumi, M., C. Li, and A. B. Hamza (2016). "A spectral graph wavelet approach for nonrigid 3D shape retrieval". *Pattern Recognition Letters*. 83: 339–348.

McInnes, L., J. Healy, N. Saul, and L. Großberger (2018). "UMAP: Uniform manifold approximation and projection". *Journal of Open Source Software*. 3(29): 861. DOI: 10.21105/joss.00861.

Mejia, D., O. Ruiz-Salguero, and C. A. Cadavid (2017). "Spectral-based mesh segmentation". *International Journal on Interactive Design and Manufacturing (IJIDeM)*. 11(3): 503–514.

Mijalkov, M., E. Kakaei, J. B. Pereira, E. Westman, and G. Volpe (2017). "BRAPH: A graph theory software for the analysis of brain connectivity". *PLOS ONE*. 12(8): e0178798.

Mohar, B. (1989). "Isoperimetric numbers of graphs". *Journal of Combinatorial Theory, Series B*. 47(3): 274–291.

Mordeson, J. N. and P. S. Nair (2012). *Fuzzy Graphs and Fuzzy Hypergraphs*. Vol. 46. Physica.

Morris, O., M. J. de Lee, and A. Constantinides (1986). "Graph theory for image analysis: An approach based on the shortest spanning tree". *IEE Proceedings F (Communications, Radar and Signal Processing)*. 133(2): 146–152.

Moura, J. M. (2018). "Graph signal processing". In: *Cooperative and Graph Signal Processing*. Ed. by P. Djuric and C. Richard. Elsevier. 239–259.

Ng, A. Y., M. I. Jordan, and Y. Weiss (2002). "On spectral clustering: Analysis and an algorithm". In: *Proc. Advances in Neural Information Processing Systems*. 849–856.

O'Rourke, S., V. Vu, and K. Wang (2016). "Eigenvectors of random matrices: A survey". *Journal of Combinatorial Theory, Series A*. 144: 361–442.

Perona, P. and W. Freeman (1998). "A factorization approach to grouping". In: *Proc. European Conference on Computer Vision*. Springer. 655–670.

Qiu, H. and E. R. Hancock (2007). "Clustering and embedding using commute times". *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 29(11): 1873–1890.

Ray, S. S. (2012). *Graph Theory with Algorithms and Its Applications: In Applied Science and Technology*. Springer Science & Business Media.

Rubinov, M. and O. Sporns (2010). "Complex network measures of brain connectivity: Uses and interpretations". *NeuroImage*. 52(3): 1059–1069. Computational Models of the Brain.

Sadhanala, V., Y.-X. Wang, and R. Tibshirani (2016). "Graph sparsification approaches for Laplacian smoothing". In: *Artificial Intelligence and Statistics*. 1250–1259.

Saito, S., D. P. Mandic, and H. Suzuki (2018). "Hypergraph p-Laplacian: A differential geometry view". In: *Proc. of the Thirty-Second AAAI Conference on Artificial Intelligence*. 3984–3991.

Sandryhaila, A. and J. M. Moura (2013). "Discrete signal processing on graphs". *IEEE Transactions on Signal Processing*. 61(7): 1644–1656.

Sandryhaila, A. and J. M. Moura (2014a). "Big data analysis with signal processing on graphs: Representation and processing of massive data sets with irregular structure". *IEEE Signal Processing Magazine*. 31(5): 80–90.

Sandryhaila, A. and J. M. Moura (2014b). "Discrete signal processing on graphs: Frequency analysis". *IEEE Transactions on Signal Processing*. 62(12): 3042–3054.

Sardellitti, S., S. Barbarossa, and P. Di Lorenzo (2017). "On the graph Fourier transform for directed graphs". *IEEE Journal of Selected Topics in Signal Processing*. 11(6): 796–811.

Schaeffer, S. E. (2007). "Graph clustering". *Computer Science Review*. 1(1): 27–64.

Scott, G. L. and H. C. Longuet-Higgins (1990). "Feature grouping by relocalisation of eigenvectors of the proximity matrix." In: *Proc. of the British Machine Vision Conference (BMVC)*. 1–6.

Shi, J. and J. Malik (2000). "Normalized cuts and image segmentation". *Departmental Papers (CIS)*: 107.

Shuman, D. I., S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst (2013). "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains". *IEEE Signal Processing Magazine*. 30(3): 83–98.

Singh, R., A. Chakraborty, and B. Manoj (2016). "Graph Fourier transform based on directed Laplacian". In: *Proc. of the IEEE 2016 International Conference on Signal Processing and Communications (SPCOM)*. 1–5.

Spielman, D. A. and N. Srivastava (2011). "Graph sparsification by effective resistances". *SIAM Journal on Computing*. 40(6): 1913–1926.

Spielman, D. A. and S.-H. Teng (2007). "Spectral partitioning works: Planar graphs and finite element meshes". *Linear Algebra and Its Applications*. 421(2–3): 284–305.

Stanković, L., D. Mandic, M. Dakovic, I. Kisil, E. Sejdic, and A. G. Constantinides (2019). "Understanding the basis of graph signal processing via an intuitive example-driven approach". *IEEE Signal Processing Magazine*. 36(6): 135–145.

Stanković, L., M. Daković, and E. Sejdić (2017a). "Vertex-frequency analysis: A way to localize graph spectral components [Lecture Notes]". *IEEE Signal Processing Magazine*. 34(4): 176–182.

Stanković, L., M. Daković, and E. Sejdić (2019). "Vertex-frequency energy distributions". In: *Vertex-Frequency Analysis of Graph Signals*. Ed. by L. Stanković and E. Sejdić. Springer. 377–415.

Stanković, L., E. Sejdić, and M. Daković (2017b). "Vertex-frequency energy distributions". *IEEE Signal Processing Letters*. 25(3): 358–362.

Stanković, L., E. Sejdić, and M. Daković (2018). "Reduced interference vertex-frequency distributions". *IEEE Signal Processing Letters*. 25(9): 1393–1397.

Stoer, M. and F. Wagner (1997). "A simple min-cut algorithm". *Journal of the ACM (JACM)*. 44(4): 585–591.

Tammen, M., I. Kodrasi, and S. Doclo (2018). "Complexity reduction of eigenvalue decomposition-based diffuse power spectral density estimators using the power method". In: *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 451–455.

Tremblay, N. and A. Loukas (2020). "Approximating spectral clustering via sampling: A review". In: *Sampling Techniques for Supervised or Unsupervised Tasks*. Springer. 129–183.

Trevisan, L. (2013). "Lecture notes on expansion, sparsest cut, and spectral graph theory". [Online], Available: URL: https://people.eecs.berkeley.edu/~luca/books/expanders.pdf.

Van Dam, E. R. and W. H. Haemers (2003). "Which graphs are determined by their spectrum?" *Linear Algebra and Its Applications*. 373: 241–272.

van der Maaten, L. and G. Hinton (2008). "Visualizing data using t-SNE". *Journal of Machine Learning Research*. 9(Nov): 2579–2605.

Vetterli, M., J. Kovačević, and V. Goyal (2014). *Foundations of Signal Processing*. Cambridge University Press.

Von Luxburg, U. (2007). "A tutorial on spectral clustering". *Statistics and Computing*. 17(4): 395–416.

Wainwright, M. J. and M. I. Jordan (2008). "Graphical models, exponential families, and variational inference". *Foundations and Trends® in Machine Learning*. 1(1–2): 1–305.

Wang, Z., E. P. Simoncelli, and A. C. Bovik (2003). "Multiscale structural similarity for image quality assessment". In: *Proc. of the Thirty-Seventh Asilomar Conference on Signals, Systems & Computers*. Vol. 2. 1398–1402.

Weiss, Y. (1999). "Segmentation using eigenvectors: A unifying view". In: *Proceedings of the Seventh IEEE International Conference on Computer Vision*. 975–982.