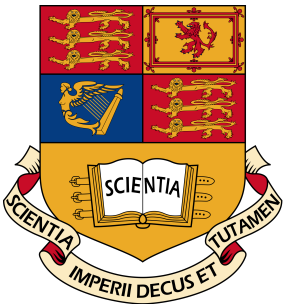

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Algorithms for Graph Partitioning

Author:
Shahrokh Shahi

Supervisor:
Dr. Mahdi Cheraghchi
Bashi Astaneh

Submitted in partial fulfillment of the requirements for the MSc degree in Computing Science of Imperial College London

September 2016

Abstract

Graph partitioning is a fundamental algorithmic problem in combinatorics motivated by applications such as clustering and community detection in social networks as well as theoretical importance in particular in spectral methods and the Unique Games Conjecture in computational complexity.

The main theme in this problem is the following: Given a graph consisting of a collection of "loosely connected dense subgraphs", design an efficient algorithm to detect, either exactly or approximately, the underlying dense subgraphs. There are many variations of the problem, ranging from information theoretic possibility of discovering the communities to efficient algorithms for doing so as well as "local walk" and distributed models for the algorithmic task.

In this research, a systematic study of the major techniques and discoveries in this area has been conducted with an emphasis on the methods based on the spectral graph theory. In the last years, spectral graph partitioning approaches have become very popular and there has been a growing interest in their applications, mainly on account of their efficiency and mathematical elegance. Therefore, the main concepts of the spectral partitioning algorithms are comprehensively discussed in this research and some novel applications of these methods have been concluded at the end.

Dedicated to
my beloved mother
for her amazing strength in life and motherhood

Acknowledgements

I would like to express my special thanks to my supervisor, Dr. Cheraghchi, for his unyielding support and invaluable help and encouragement. Working under his supervision has been more than a great pleasure.

I would like to thank my parents for their love, blessing, and constant and unfailing support in all aspect of my life. Without their support and faith in me, I would not be the man I am today. No words are enough to express my gratitude toward them.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Motivation and Objectives

In general, graphs are frequently used to model many types of relations and processes in science and engineering applications. A wide-range of practical problems can be represented by graphs. Particularly, in computer science, graphs are employed to represent computer networks, social networks, data organization, computation flow, etc.

Accordingly, there exist many applications in which cutting a graph into smaller pieces is a fundamental requirement. Partitioning large graphs has an important role in complexity reduction and parallelization in various applications such as scientific simulation, social networks, and, road networks.

Motivated by such applications, this research aims at investigating the available graph partitioning techniques to obtain a comprehension of key concepts in this area. Such an understanding will play an important role in improving the available algorithms and designing novel strategies for new applications. Pointing out the key features of each algorithm, such research can also be helpful in development of functional software tools for real-world applications.

Fundamentally, graph partitioning problems are NP-hard problems and the solutions are typically obtained through heuristics and approximation algorithms. Accordingly, a wide variety of algorithms have been proposed in literature to provide practical solutions. The first objective of this project is providing a structured survey of these algorithms and their applications.

Supported by a strong mathematical background, spectral graph theory provides a variety of functional methods for graph partitioning. More specifically, spectral partitioning algorithms provide solution to partitioning problems by computing eigenvalues and eigenvectors of some matrices associated to the graph. On account of the great performance and growing interest in these methods, the second aim of this project is studying the spectral graph partitioning algorithms, in detail.

This research mainly focuses on discussing the available algorithms and their applications; however, proposing new implementation and applications of spectral partitioning approaches can be considered as an extension to this research project.

To conduct this research, initially, the application of graph partitioning methods will be investigated. Then, a systematic survey of these techniques will be presented. Eventually, the rest of research will be concentrated on spectral partitioning algorithms and the extensions.

1.2 Graph Partitioning Applications

Graph partitioning has many applications in computer science. To understand the importance of this methods, some of the prominent applications are briefly described in this section.

1.2.1 Parallel Processing

Generally, parallel processing is required in massive scientific computations. Parallel graph computations take advantage of graph partitioning in problems such graph eigenvalue computations, breadth-first search, and, PageRank and connected components. For the distribution of works to processors of a parallel machine, graph partitioning concepts is implemented to ensure the load balance and minimize the communication between processors [10].

In this area, two distinct type of partitioning has been introduced: static partitioning and periodic partitioning. Static partitioning is the approach in which graph partitioning has been applied once in the beginning of the computation. This approach can only be employed when the problem domain does not change as the computation proceeds. Periodic partitioning should be employed in scientific computing or visualization with evolving computational domains e.g. Adaptive Mesh Refinement. In these cases, the graph model may be augmented with additional nodes and edges to model new additional features [10].

1.2.2 Complex Networks

Most biological, social, and technological networks contain many features with patterns of connection between their elements. These complex networks can be represented by graphs and introduces numerous further applications of graph partitioning problems. A common application in this context is to identify groups of similar entities with specific features. These types of localization can be recognized based on the principal of finding groups of entities weakly connected to the rest of the network [13]. Most of the time, such connectivity also represents similarity [63]. Real-life application of complex networks often leads to optimization problems on weighted graphs [13].

- **Power Grids** Power networks can be represented by a complex network graph. Splitting a power network into self-sufficient divisions is an approach to prevent the propagation of cascading failures in this networks [57]. In this context, the vulnerabilities of the power system will be evaluated by employing graph partitioning approaches. Especially, spectral graph partitioning is used in the vulnerability detection by splitting the network into regions with excess generation and excess load [13].
- **Biological Networks** Graph representations can be employed for modeling many complex biological systems, such as protein-protein interactions and gene co-expression networks. The correspondence graph can be created by assuming the biological entities, e.g. proteins, as vertices and establishing the edges according to participation of the entities in some biological processes [13]. Partitioning and clustering of such networks may have several goals, such as data reduction obtained by the assumption of the biologically similarity in the clustered nodes [17].
- **Social Networks** Community detection is one of the most popular topics in social network science. Graph partitioning methods are often used as first approximation in this problem [84].
- **Geographically Embedded Networks** Another application of graph partitioning methods in complex networks is in analyzing the spatial network data which have been increased drastically in the recent years by growing application of location-aware devices, such as GPS. In this situation, using graph partitioning approaches to improve the efficiency of data analyzing algorithms for spatial data and geographical networks is unavoidable [13].
- **Road Networks** Graph partitioning can also be employed to improve the performance of route planning algorithms in road networks. For instance, in the algorithm known as arc-flags, a geometric partitioning approach is used as preprocessing step to reduce the search space in Dijkstra's algorithm [13].

1.2.3 Image Processing

In computer vision, image segmentation is the field in which graph partitioning and clustering methods have been employed, noticeably. The main purpose of image segmentation is to partition the pixels of an image into specific groups known as objects. This approach leads to an extreme compressed representation of the image and the computations after segmentation are consequently cheaper. Therefore, the segmentation execution can be considered as the most demanding in an image processing procedure [67]. In a graph-based representation of an image each pixel (or a group of pixels) of the image is related to a node in the graph. Then, two nodes will be connected through a weighted edge, if some similarity exist between them. The similarity is usually defined as small geodesic distance which can result in a mesh-like graph. Moreover, the edge weights represent the difference in the intensity between the connected nodes [67].

1.2.4 VLSI Physical Design

Graph partitioning is widely used in physical design of digital circuit for very large-scale integration (VLSI) systems [13]. The purpose of the partitioning in this problems is reducing the complexity of the VLSI design by divide it into smaller components. The typical optimization objective in this problem is minimizing the total weight of connection between subcircuits. In this manner, the graph nodes are cells which are small units of the circuit, such as gates, and the edges of the graph are the wires which should be kept minimum [13].

1.2.5 Clustering

Clustering algorithms are a functional tool for exploring data structures and may be employed in many disciplines. Two prominent clustering approaches can be found in the literature: kernel methods and spectral methods.

In general, clustering methods have been employed in many contexts such as data mining, image segmentation and pattern classification [73]. The aim of clustering methods is to group patterns on the basis of similarity criteria. Accordingly, clusters are sets of similar patterns and pattern recognition is the most crucial aspect in clustering [63]. Clustering performance can be improved by a good choice of representation of patterns.

From another practical point of view, the clustering techniques can be divided into two main categories: hierarchical and partitioning.

In hierarchical clustering techniques [32], a structure can be obtained that it can recursively be divided into substructures. The ultimate hierarchical structure of clusters is called dendrogram [32].

Partitioning clustering approaches are often based on the optimizing an associated objective function and try to obtain a single partition of data. The result is emerging hypersurfaces among clusters [45]. For instance, Figure 1.1 exhibits two potential nonlinear clusters.

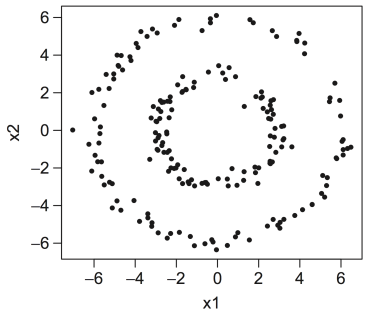

Figure 1.1: A data set composed of two rings of points [32]

It has been demonstrated that standard partitioning methods are not able to separate this data set into two rings by using two centroids. This problem could be

solved by employing many centroids which provides a complex description for a simple data set [32]. Accordingly, several modifications and new approach have been implemented which is resulted in two prominent families of clustering algorithms: kernel and spectral clustering methods [32, 73].

The kernel clustering methods is including a wide range of algorithms such as K-means, fuzzy, c-means, and, SOM. Employing the kernels in this methods allows to map implicitly data to a high dimensional space called feature space [32].

Spectral clustering method is on the basis of concepts in spectral graph theory. In this approach, an associated weighted graph will be constructed corresponding to the initial data set in which each node represents a pattern and the similarity among each pair of pattern is shown by a weighted edge between them. In this manner, the clustering problem transforms to a graph partitioning problem. Therefore, the spectral graph theory tools can be implemented to solve clustering problems [35].

K-means clustering method is one of the most functional and well-known kernel clustering methods. It is actually a method of vector equalization which is popular for cluster analysis in data mining. K-means clustering aims at partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean serving as prototype of the cluster. This problem is computationally difficult (NP-hard); however, there are efficient heuristic algorithms that are commonly employed and converge quickly to local optimum [35, 45].

1.2.6 Community Detection

A well-known family of problems employing graph partitioning methods is community detection in complex networks. Many real-world networks essentially organized due to a community structure [17]. Therefore, many research has been devoted to discover and highlight these structure of networks. As it described in section 1.2.2, many real-world complex networks can be represented by graphs. On account of complex features found in the this complex networks, the corresponding graph model can be intricate including various features. Thus, various algorithms have been proposed in the literature to detect the communities and each of them is focusing on specific properties of this networks.

A *community* can be considered as a set of entities so as each entity is closer to the other ones within the community than to the entities outside it. In other words, communities are groups of entities sharing some comon properties [66, 17].

According to the clustering concepts presented in section 1.2.5, community detection is similar to the clustering problems in data mining context [17]. In clustering problems large sets of data will be partitioned into clusters which are homogeneous groups. By introducing the concept of community and common entities, the community discovery can be viewed as a clustering problem in graphs so that an unsupervised classification of graph nodes is the objective. Specifically, community detection is the most data mining application on social networks [11].

1.3 Report Outline

Following the objective of the research, the remainder of this report is organized as follows:

- Chapter 2 provides the essential mathematical background required in this research. In particular, a review of graph theory and linear algebra will be presented in this chapter.
- Chapter 3 provides a general overview of graph partitioning methods and a comprehensive classification of available approaches will be presented in this chapter.
- Chapter 4 discusses the spectral graph partitioning methods in detail. In this chapter all the significant basis of spectral graph theory will be presented. Besides, some related topics of spectral methods and theorems will be studied, as well.
- Chapter 5 briefly introduces the spectral graph clustering algorithms and explains the relation between graph clustering and graph partitioning methods.
- Chapter 6 proposed a novel application of spectral partitioning concepts in graph-constrained group testing. More specifically, a state-of-the-art algorithm for partitioning a graph into expander subgraphs and the possibility of its application in graph-constrained group testing problems is discussed.
- Chapter 7 introduces an addition to this project which is a MATLAB toolbox developed alongside this research to improve the comprehension of spectral graph partitioning concepts.
- Chapter 8 points out the achievements of this research and also provides some suggestions for future works.

Chapter 2

Preliminaries

2.1 Overview

In this chapter the necessary terminology and mathematical background required for the rest of the report is provided. This chapter includes a review of graph theory in which the basic terminology of the graph theory has been discussed. Additionally, even though familiarity with the basic concepts of linear algebra is assumed, the key concepts required to understand spectral graph theory is presented at the second part of this chapter.

2.2 A Review of Graph Theory

In a general view, graph is a mathematical structure used to model relations between objects. The formal and technical definitions in the graph theory are as follows [93, 73]:

2.2.1 Basic Concepts

- **Graph:** In the graph theory, a graph G is considered as a pair of sets $G = (V, E)$, where V is a non empty set of nodes (or vertices) and E is the set of edges. In an undirected graph, each edge can be defined by an unordered pair v, w. In a directed graph, edges are ordered pairs.
- **Graph Order:** The number of the vertices $n = |V|$ is defined as *the order of the graph*.
- **Graph Size:** The number of edges $m = |E|$ is *the size of the graph*.
- **Weighted Graph:** In a weighted graph, a weight is assigned to each edge by defining a weight function $w : E \to \mathbb{R}$.

- **Planar Graph:** A graph is planar if it can be drawn in a plane without any of the edge crossing.
- **Graph Density:** The density of a graph has been defined as the ratio of the number of edges per the maximum possible,

$$\text{Graph density} = \frac{m}{\binom{n}{2}} \tag{2.1}$$

- **Neighborhood:** Vertex v is a neighbor of u, if v and u are endpoints of an edge in the graph. The set of neighbors for a given vertex v is denoted by $\Gamma(v)$.
- **Node Degree:** In an undirected unweighted graph, the number of the adjacent edges of a node is considered as its degree and denoted by d. However, in an undirected weighted graph, the degree of a node can also be defined as the sum of the weights of its adjacent edges.
  - A vertex of degree 0 is called *isolated* vertex
  - *Minimum Degree* of a graph G, denoted by $\Delta(G)$, is the minimum degree among the vertices of G.
  - *Maximum Degree* of a graph G, denoted by $\delta(G)$, is the maximum degree among the vertices of G.
- **Regular Graph:** A graph is regular if all the vertices have the same degree. Accordingly, if the degree of all vertices is r, the graph is considered as a r-regular graph.

2.2.2 Subgraphs

- **Subgraph:** A graph H is called a subgraph of a graph G, written $H \subset G$, if $V(H) \subset V(G)$ and $E(H) \subset E(G)$.
- **Spanning Subgraph:** It is a subgraph with the same vertex set as the original graph, i.e. $V(H) = V(G)$.

2.2.3 Walk, Path, Trail, Circuit, and Circle

- **Walk:** A $u-v$ walk is defined as a sequence of vertices in a graph G, beginning with u and ending with v. If $u = v$, then the walk is *closed*, otherwise, it is known as an *open* walk.
  - The number of edges in a walk is called the *length* of the walk.
  - A walk of length 0 is known as a *trivial* walk.
- **Path:** A path is defined as a walk in which no vertex is repeated more than once. In other word, a path in a graph is a sequence of vertices in which there exists an edge connecting every two consecutive vertices.

- **Trail:** A walk in which each edge is visited no more than once.
- **Circuit:** A closed trail of length ≥ 3 is known as a circuit.
- **Cycle:** A circuit in which no vertex is repeated, except for the first and last.

2.2.4 Connected Graphs

- **Connected Graph:** A graph is recognized as a connected graph, if there is a path between every pair of vertices.
- **Connected Component:** A connected component is a subgraph with a path connecting every pair of vertices.
- **Distance:** The distance between two vertices, u and v denoted by $d(u, v)$, is defined as the smallest length of any $u - v$ path in a graph G.
- **Diameter:** The greatest distance between any two vertices of a connected graph G is defined as the diameter of the graph and denoted by $diam(G)$.

2.2.5 Specific Classes of Graphs

- **Complete Graph:** A graph is complete, if there is an edge connecting any two nodes of a graph. A complete graph of order n is denoted by K_n and it is of size $\binom{n}{2}$. Therefore, according to Equation 2.1, the density of a complete graph is one.
- **Bipartite Graph:** A graph G partitioned into two subset of vertices U and W, is called partite sets, if every edge of G joins a vertex of U to a vertex of W. Figure 2.1 depicts a bipartite graph.

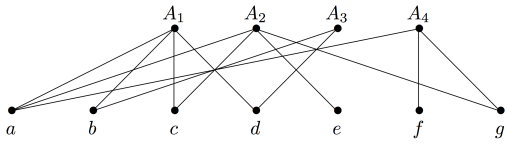

Figure 2.1: A bipartite graph [51]

- **Complete Bipartite Graph:** A bipartite graph with two subset of vertices U and W, is called complete bipartite, if there is an edge between every vertex of U and W.
- **k-partite Graph:** A graph G is a k-partite graph, when $V(G)$ is partitioned into k subsets, such that (u, v) is an edge of G if u and v belong to different partite sets.

2.2.6 Graph Matrices

- **Adjacency Matrix:** The adjacency matrix $\boldsymbol{A}$ of a given graph G of order n is an $n \times n$ matrix $\boldsymbol{A} = [a_{ij}]$, where $a_{ij} = 1$, if there is an edge connecting node v_i and v_j and 0, otherwise.
- **Weight Matrix:** The edge weights are represented by a weight matrix $W = [w_{ij}]$, where w_{ij} is the edge weight between vertices v_i and v_j.

2.3 A Review of Linear Algebra

In this section, some concepts of linear algebra required in spectral graph theory and particularly spectral partitioning methods is discussed [80].

2.3.1 Basic Definitions

Consider $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ as an $n \times n$ matrix and $\boldsymbol{x} \in \mathbb{R}^n$ as a column vector. In this case,

- The matrix $\boldsymbol{A}$ is identified as *linear transformation* from n-dim vectors to n-dim vectors.
- A set of vectors $\boldsymbol{v_1}, \boldsymbol{v_2}, ..., \boldsymbol{v_k}$ are *linearly independent*, if $c_1\boldsymbol{v_1}+...c_k\boldsymbol{v_k} = 0$ implies $c_1 = c_2 = ... = c_k = 0$; otherwise, they are *linearly dependent*.
- The *null space* or the *kernel* of a matrix $\boldsymbol{A}$ is defined as nullspace$(\boldsymbol{A}) = \{\boldsymbol{x} \in \mathbb{R}^n | \boldsymbol{A}\boldsymbol{x} = 0\}$ and null($\boldsymbol{A}$) denotes its dimension.
- The *range* of a matrix $\boldsymbol{A}$ is defined as range$(\boldsymbol{A}) = \{\boldsymbol{A}\boldsymbol{x} | \boldsymbol{x} \in \mathbb{R}^n\}$ and rank($\boldsymbol{A}$) denotes its dimension.
- Note rank($\boldsymbol{A}$)+null($\boldsymbol{A}$)=n.
- Note rank($\boldsymbol{A}$) = maximum number of linearly independent column of $\boldsymbol{A}$.

2.3.2 Eigenvalues and Eigenvectors

Let $A \in \mathbb{C}^{n \times n}$ be a square matrix, $\lambda \in \mathbb{C}$ is a scaler and $\boldsymbol{v} \in \mathbb{C}^n - \{\boldsymbol{0}\}$ is a non-zero vector. In this case, λ is called an eigenvalue of $\boldsymbol{A}$ if

$$\boldsymbol{A}\boldsymbol{v} = \lambda\boldsymbol{v} \tag{2.2}$$

and $\boldsymbol{v}$ is known as the corresponding eigenvector. Equation (2.2) is equivalence to $(\boldsymbol{A} - \lambda\boldsymbol{I})\boldsymbol{v} = 0$, where $\boldsymbol{I}$ is the identity matrix. Thus, for a non-zero vector $\boldsymbol{v}$ can be inferred

$$\det(\boldsymbol{A} - \lambda\boldsymbol{I}) = 0 \tag{2.3}$$

By the definition of determinant, $\det(\boldsymbol{A} - \lambda \boldsymbol{I})$ is a polynomial of degree n in λ that is called *characteristic polynomial* of $\boldsymbol{A}$ and, over the complex numbers, has exactly n solutions. Each of the roots is an eigenvalue and any vector in nullspace($\boldsymbol{A} - \lambda \boldsymbol{I}$) is an eigenvector. The characteristic polynomial can be solved by Gaussian elimination or interpolation and any root of it is an eigenvalue.

Geometrically, an eigenvector is a direction that is fixed by the linear transformation. It should be noted that all matrices have not real eigenvalues, but it can be demonstrated that real symmetric matrices have real eigenvalues. Therefore, the eigenvalues of the adjacency matrix of an undirected graph are all real.

2.3.3 Orthogonality

The next important and useful concept in spectral theorem is the *orthogonality*. Given two n-dimensional vectors $\boldsymbol{u} = (u_1, u_2, ..., u_n)^T$ and $\boldsymbol{v} = (v_1, v_2, ..., v_n)^T$, the *inner product* of $\boldsymbol{u}$ and $\boldsymbol{v}$ is defined as

$$\langle \boldsymbol{u}, \boldsymbol{v} \rangle = \sum_{i=1}^{n} u_i v_i \tag{2.4}$$

Besides, the length of a vector $\boldsymbol{u}$ which is also known as the *norm* of $\boldsymbol{u}$ is defined as

$$\| \boldsymbol{u} \| = \sqrt{\langle \boldsymbol{u}, \boldsymbol{u} \rangle} \tag{2.5}$$

By these definitions, two vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ are orthogonal (or perpendicular) if and only if $\langle \boldsymbol{u}, \boldsymbol{v} \rangle = 0$. Accordingly,

- A set S of vectors are called orthogonal, if for any distinct $\boldsymbol{u}, \boldsymbol{v} \in S$, $\boldsymbol{u}$ and $\boldsymbol{v}$ are orthogonal.
- A set S of vectors are called *orthonormal*, if S is orthogonal and for any $\boldsymbol{u} \in S, \| \boldsymbol{u} \| = 1$
- A set S of vectors are called a *basis* for a vector space V, if S is linearly independent and every vector of V is a linear combination of the vectors in S.

2.3.4 Spectral Theorem for Real Symmetric Matrices

In this section, some of the functional concepts in spectral theory is discussed. The proof of these theorems are available in literature and has not been discussed here.

Theorem 1
If $\boldsymbol{A}$ is a real symmetric $n \times n$ matrix, then, there is an orthonormal basis of $\mathbb{R}^n$ consisting of eigenvectors of $\boldsymbol{A}$, and the corresponding eigenvalues are real numbers.

Eigenspace

Let λ be an eigenvalue of a matrix $\boldsymbol{A}$. Then the eigenspace of λ is defined as nullspace($\boldsymbol{A} - \lambda \boldsymbol{I}$). Thus, for a real symmetric matrix $\boldsymbol{A}$, it can be demonstrated that null($\boldsymbol{A} - \lambda \boldsymbol{I}$) = multiplicity of λ.

Eigen-decomposition

Let $\{\boldsymbol{v_1}, \boldsymbol{v_2}, ..., \boldsymbol{v_n}\}$ be a basis of orthonormal eigenvectors. Then any vector $\boldsymbol{x}$ can be written in the form of $c_1\boldsymbol{v_1} + c_2\boldsymbol{v_2} + ... + c_n\boldsymbol{v_n}$

2.3.5 Graph Spectrum

The adjacency matrix of a graph has been defined in Section 2.2.6. According to the spectral theorem, the adjacency matrix has an orthonormal basis of eigenvectors with real eigenvalues.

In the following, the spectrum of some specific graphs discussed in Section 2.2.5 is presented.

2.3.6 Complete Graph

If G is a complete graph K_n, then $\boldsymbol{A}(G) = \boldsymbol{J} - \boldsymbol{I}$, where $\boldsymbol{J}$ denotes the all-one matrix. Obviously, any vector is an eigenvector of $\boldsymbol{I}$ with eigenvalue 1. Therefore, the eigenvalues of $\boldsymbol{A}$ will be one less than that of $\boldsymbol{J}$. Since $\boldsymbol{J}$ is of rank 1, there are $n - 1$ eigenvalues of 0. Moreover, the all-one vector is an eigenvalue of $\boldsymbol{J}$ with eigenvalue n, Therefore, $\boldsymbol{A}$ has one eigenvalue of $n - 1$, and $n - 1$ eigenvalue of -1.

2.3.7 Bipartite Graph

A bipartite graph can also be characterized by the spectrum. In is claimed that

1. If G is a bipartite graph and λ is an eigenvalue of $\boldsymbol{A}(G)$ with multiplicity k, then $-\lambda$ is an eigenvalue of $\boldsymbol{A}(G)$ with multiplicity k. This relation reveals that the spectrum of a bipartite graph is symmetric around the origin. The converse is also true which is presented in the following statements.

2. If the nonzero eigenvalues of the adjacency matrix of a graph, $\boldsymbol{A}(G)$, occurs in pairs, then G is bipartite.

Both of these claims are proved in [80]. A nice result of them is that it can be generalized to show the following:

λ_n is close to $-\lambda_1$ if and only if G is close to a bipartite graph (i.e. a large max-cut). This result leads to a nontrivial approximation to the max-cut problem explained in Section 4.10.2.

2.3.8 Positive Semidefinite Matrix

A real symmetric matrix $\boldsymbol{M}$ is a *positive semidefinite* matrix, if $\boldsymbol{x}^T\boldsymbol{M}\boldsymbol{x} \geq 0$ for every vector $\boldsymbol{x}$. It is demonstrated that a real symmetric matrix $\boldsymbol{M}$ is positive semidefinite if and only if all eigenvalues are non-negative if and only if $\boldsymbol{M} = \boldsymbol{B}\boldsymbol{B}^T$ for some $\boldsymbol{B}$.

It is worth mentioning that the Laplacian matrix can be written as $\boldsymbol{B}\boldsymbol{B}^T$. Thus, it is a positive semidefinite matrix and all the eigenvalues of this matrix are non-negative. Therefore, 0 is the smallest eigenvalue of Laplacian matrix.

2.3.9 Connectedness

There are two important corollaries concerning connectedness in algebraic graph theory [36, 80]:

1. A graph is connected if and only if 0 is an eigenvalue of $\boldsymbol{L}(G)$ with multiplicity 1. A practical thing about this characterization is that it can be generalized in the following way:

   λ_1 and λ_2 are very close if and only if the graph is close to be disconnected. As a very useful result, this provides a nontrivial approximation to the sparsest cut problem (See Section 4.9).

2. The Laplacian matrix $\boldsymbol{L}(G)$ has 0 as its eigenvalue with multiplicity k if and only if the graph G has k connected components (see Section 4.12).

Chapter 3

Graph Partitioning Approaches

3.1 Overview

In this chapter the definition of graph partitioning problem is featured. Then, following the objective of this research, a structured overview of available graph partitioning algorithms is presented and the key ideas of each technique has been briefly described. To achieve this purpose, both global and heuristic methods are discussed, and the multilevel graph partitioning method is thoroughly studied at the end of this chapter.

3.2 Graph Partitioning Problem

3.2.1 Definition

Graph partitioning problem (GPP) is defined on data represented in the form of a graph $G = (V, E)$, where V is the set of vertices and E is the set of edges, such that it is possible to partition G into smaller components with specific properties.

In a general definition of the graph partitioning problems, given a graph $G = (V, E, W_V, W_E)$, where W_V and W_E are vertices weights and edge weights, respectively, the objective is choosing a partition $V = V_1 \cup V_2 \cup ... \cup V_P$ such that

- The sum of the vertex weights in each N_i is distributed evenly (load balance)
- The sum of all edge weights of edges connecting all different partitions is minimized (e.g. decreasing parallel overhead in parallel computations)

In a real-world example, if V is considered as tasks, then W_V is task costs and the work load $W_E(j, k)$ is sent by task j to task k. In this way, graph partitioning problem actually means dividing works evenly and minimizing the required communications

[63]. One of the fundamental case of graph partitioning problems is graph bisection which is dividing a given graph into two parts [80]. This family of algorithms can be employed for complete graph partitioning by implementing recursion approaches.

3.2.2 Problem Complexity

Generally, graph partitioning problems are categorized as NP-hard problems [13]. In computational complexity theory, NP-hardness is a class of problems which are at least as hard as the hardest problems in NP. NP stands for Non-Deterministic Polynomial time which is regarded as the set of all decision problems for which there exist some verifiable proofs in polynomial time by a non-deterministic Turing machine to prove that a given solution is actually a solution [89].

On account of the hardness nature of graph partitioning problems, most of the GP algorithms are based on the heuristic approaches. In the following, a comprehensive classification of the available GP algorithms has been discussed.

3.3 Global Algorithms

One of the main and fundamental methods in graph partitioning is the global algorithms which work on the entire graph and compute a solution directly. These algorithms are usually computationally expensive; therefore, they usually employed for smaller graphs and may only be restricted to bipartitioning problems, although they can be generalized to k-partitioning problems by implementing recursion approaches. They also can be employed as subroutines in some other complex algorithms such as multilevel approaches [13].

3.3.1 Exact Algorithms

There exists a large amount of exact methods solving graph partitioning problems optimally. Most of these methods dedicated to the bipartitioning problems and some others solve the general partitioning problems. Most of these algorithms are based on branch-and-bound approach [13, 53].

Taking this approach, bounds can be obtained by employing different methods, such as semi-definite programming, linear programming, etc. Depending on the method used, two different situation may be happened. In one possible scenario, the bounds are very good resulted in small branch-and-bound trees but hard to compute. In the second possible scenario, bounds are weaker which lead to larger trees and faster computation.

It should be noted that this approach can only be employed in very small problems by accepting very large running times. Furthermore, the experimental evaluation of this method only has been conducted for less than 4 block numbers. Besides, the

methods suggested for bipartitioning problems can solve the problems in a moderate running time only if the bisection width of graph is small [18].

3.3.2 Spectral Partitioning

One of the most first and most important methods to partition a graph is spectral bisection. This methods have been employed by Donath and Hoffman [24, 23] and Fiedler [31]. In this method, the information required in partitioning procedure is typically obtained by computing the eigenvalues and eigenvectors of the Laplacian matrix $\boldsymbol{L}$ of the graph. The spectral partitioning approaches will comprehensively be discussed in the next chapter.

3.3.3 Graph Growing

Graph growing is a simple approach for obtaining a bisection of a graph. Most of the algorithms implementing this approach are based on breadth-first search (BFS). In this methods, the procedure is started from a random node v and the nodes traversed by a BFS will be assigned to the first block, V_1. The search will be stopped after half of the original node weights are assigned to this block and the remaining nodes, $V - V1$, will be assigned to the second block, V_2.

A local search algorithm can be employed to improve the resulted partition. Furthermore, multiple restarts of the algorithm can be helpful in obtaining a good solution. Another effective improvement is trying to find a good starting node by choosing a node which has maximal distance from a random seed node [4]. Some of this algorithms always add the node to the block that results in the smallest increase in the cut [47, 13].

3.3.4 Flows

In this approach, the max-flow min-cut theorem [33] has been employed for bisecting a graph by computing a maximum flow and then consequently a minimum cut between node sets. The drawback of this approach is ignorance of the balance during the partitioning procedure. Therefore, it is not clear how to apply this approach to the balanced graph partitioning problems. But, it can still be employed to partition random regular graphs with small bisection width [13, 12].

The maximum flow approaches are also can be used as a subroutine for improving the partitioning procedure or coarsening in the multilevel framework. Furthermore, flow computations can be employed when the quality of partition is measured by expansion or conductance [54, 7].

3.3.5 Geometric Partitioning

When the coordinates of graph nodes are available, geometric partitioning approaches can be employed. These methods are particularly functional in finite element models and other geometrically-defined graphs. In such graphs, blocks with small cut are considered as geometrically compact regions. This approach has been implemented in different schemes:

- **Recursive Coordinate Bisection (RCB):** In this method, in each step of recursions, the graph nodes are projected onto the coordinate axis with the longest expansion of the domain and the median of their projections is then utilized to bisect nodes [78]. The bisection plane is orthogonal to the coordinate axis which can provide large separators for partitioning meshes with skewed dimensions [78, 13].

- **Inertial Partitioning:** This method can be considered as an improvement to RCB by choosing the bisecting plane orthogonal to a plane L which minimizes the moments of inertia of nodes [28]. In this way, the chosen projection plane L minimizes the sum of the squared distances of all nodes which consequently leads to an improvement in the worst case performance [13].

- **Random Sphere Algorithm:** This algorithm generalizes the RBC algorithm by stereographically projecting the d-dimensional nodes to a random $(d+1)$-dimensional sphere bisected by a plane through its center point [37]. For instance, Figure 3.2 demonstrates the projected points of a given finite element mesh illustrated in Figure 3.1.

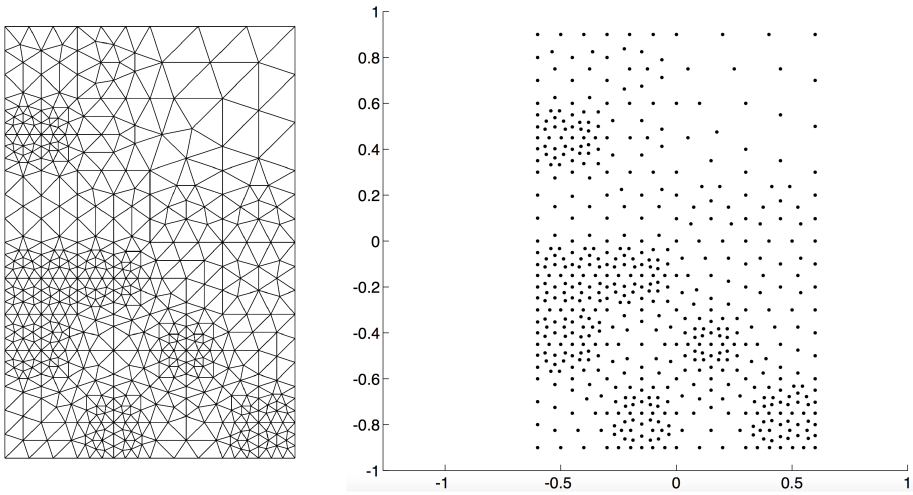

Figure 3.1: The input finite element mesh (left), the mesh points (right) [37]

This method performs well for well-behaved graphs, such as planar graphs and k-nearest neighbor graphs [13].

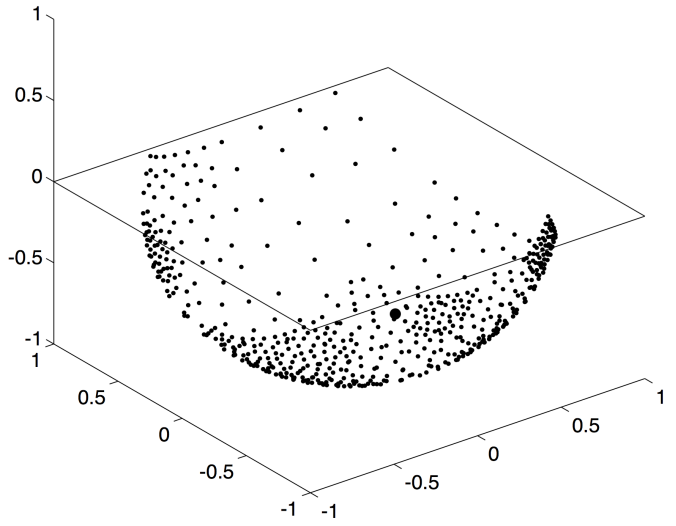

Figure 3.2: Projected mesh points onto a sphere [37]

- **Space-Filling Curves:** This type of geometry-based partitioning algorithms reduces the d-dimensional problem to the one-dimensional case. Fundamentally, space-filling curves define a bijective mapping from V to $\{1, ..., |V|\}$. By implementing this mapping, nodes locality is preserved and the partition will computationally be cheaper than RCB method [44].
- **Multilevel Graph Drawing Algorithm:** Embedding arbitrary graphs into the coordinate space is an approach to demonstrate the graph information in the geometric form. This approach has been implemented by multilevel graph drawing algorithm [52].

3.3.6 Stream Graph Partitioning

One of the most recent research area in big data processing is the streaming data models [13]. In such models, the data arrives in streams and the processing needs to be done by using less space than the overall input size. Streaming graph partitioning (SGP) algorithms are extremely suitable for these problems. SGP algorithms are even faster than multilevel algorithms but give lower solution quality. They can also be implemented in dynamic networks where fast repartitioning methods are greatly needed, if a good initial solution is supplied by a strong static data algorithm [81].

3.4 Improved Heuristic Methods

The majority of the graph partitioning algorithms have been more efficient by iteratively improving starting solutions [13]. Some of these improved approaches are discussed in this section.

3.4.1 Node-swapping Local Search

Local search approach is a simple way for optimization in which a solution will iteratively be changed by choosing a new node from a neighborhood. Different techniques can be distinguished by their definition of the neighborhood and their selection strategies [13].

Kernighan and Lin [49] were probably the first to present a local search method for graph partitioning problems. In their method, the selection strategy aims at finding the swap of node assignments which leads to the largest decrease in the total cut size. Decreasing in total cut size can also be negative. An iteration comes to the end when all nodes have been moved in this way. Then, the solution will be reset to the best solution obtained in this iteration. Finally, the termination will happen when the last iteration has no improvement.

The most important downside of the KL algorithm is its expensive asymptotic running time. The first implementation of the KL method assumed to take time $O\left(n^2 \log n\right)$ which can be improved to $O\left(m \max\left(\log n, \Delta\right)\right)$, where Δ is the maximum degree of the graph [26]. Fiduccia and Mattheyses [29] improved this method by designing and implementing proper data structures which leads to an $O(m)$ asymptotic running time. This modified method is known as KL/FM local search algorithm.

The KL/FM method was further improved by only allowing nodes to move and stopping a round when the edge cut does not decrease after moving x nodes [48]. In this approach the quality can be improved by random tie breaking and running additional rounds even when no improvements have been found [13].

There exists a highly localized version of KL/FM algorithm proposed by Osipov and Sanders [65] in which the search spreads from a single boundary node. Moreover, a stochastic model of the search is implemented to predict that whether further improvement can be obtained or the the search should be terminated. In this way, the method will have a better chance to climb out of local minimums and find better cuts for the GP solvers KaSPar [65] and KaHIP [70].

Another possible approach is moving just a single node at a time which is allowing flexible tradeoffs between reducing the cut or improving balance [43]

Furthermore, defining a more general neighborhood relation for bipartitioning problems, Diekmann et al. [19, 62] introduce a new algorithm in which whole set of the nodes are exchanged between the blocks instead of moving single nodes. This approach can improve the cut and the quality of the solution is often better than most of the other methods, while its running time is almost equal to the KL/FM algorithm [62]. Algorithm 1 demonstrates the main procedure of this approach.

Algorithm 1: The main algorithm of Improved Bisection [19]

```
   Data: The given graph G = (V, E)
   Result: graph partitions
 1 initialization;
 2 limit = cutsize/2;
 3 while limit ≠ 0 do
 4 |   search for k-helpful set S with k ≥ limit ;
 5 |   if no k-helpful set S with k ≥ limit found then
 6 |   |   if any k-helpful set S with k > 0 found then
 7 |   |   |   S = set with highest helpfulness found;
 8 |   |   |   limit = H(S);
 9 |   |   else
10 |   |   |   S = 0 ;
11 |   |   |   limit = 0;
12 |   |   end
13 |   end
14 |   if S ≠ 0 then
15 |   |   move S to the other side ;
16 |   |   search for a balancing set S̄ of S ;
17 |   |   if successful then
18 |   |   |   move S̄ to the other side ;
19 |   |   |   limit = limit × 2 ;
20 |   |   else
21 |   |   |   move S back to its original position ;
22 |   |   |   limit = ⌊limit/2⌋ ;
23 |   |   end
24 |   end
25 end
```

where the k-helpful set is defined as the set of nodes that improves the cut size by k and the balancing set is the set increases the cut size by not more than $k-1$ edges.

3.4.2 Extension to k-way Local Search

For a variety of applications explained in Section 1.2, such as parallel processing and VLSI design, the vertices of a graph are needed to be partitioned into $p > 2$ subsets. These problem are known as p-way partitioning problems. The most commonly used p-way partitioning method is recursive bisection in which a graph is firstly divided into two partitions by an efficient bisection algorithm. Then, the subgraphs will be divided, recursively [79]. The scheme of this approach is illustrated in Algorithm 2.

Algorithm 2: Recursive Bisection Scheme [79]

Data: The given graph $G = (V, E)$ and an integer p, $K = n/p$
Result: a p-way partition of G

```
1 Procedure Recursive Bisection Scheme(G,p)
2     apply BISECTION to find a bisection G_L and G_R of G ;
3     if |G_L| > K then
4         Recursive Bisection Scheme(G_L,p/2) ;
5         Recursive Bisection Scheme(G_R,p/2) ;
6     end
7 end
8 return the subgraphs G_1, ..., G_p ;
```

In the recursive bisection algorithms, lacking of global knowledge may lead to creating partitions that are very far away from the optimal solution [79]. Therefore, designing and implementing k-way local search algorithms is a requirement. The KL/FM algorithm can be extended to a local search algorithm for k-partitioning problem.

One of the extensions of the KL/FM algorithm to k-way local search is using $k(k-1)$ priority queues so that each type of move (source block, target block) uses one of them. In the case of single movement, the node maximizing the gain will be chosen.

In some further improvements, a k-way version of the KL/FM algorithm that runs in linear time O(m) [48]. In this algorithm, a single global priority queue is used for all types of moves in which the priority is determined by the maximum local gain which is the maximum reduction in the cut when the node has been moved to one of the neighbour blocks. In this way, the selected movement leads to the maximum improvement in the objective function and maintaining or improving the balance constraint.

The main approach in the most of the current local search algorithms is exchanging nodes between the blocks of the partition in order to decrease the cut size and maintain balance. One of the improvement to this approach proposed by Sanders and Schulz [72, 71] in which a combination of multiple local search method has been employed to relax the balance constraint for node movements but maintain it, globally. In their approach, the combination problem to find negative cycles in a graph is reduced by employing some efficient algorithms.

3.4.3 Tabu Search

The tabu search method can be implemented to obtain a k-way local search algorithm [38, 39] which has been applied in many graph partitioning problems. The main approach implemented in most of these algorithms is as follows [34]:
In comparison to the traditional KL/FM method in which a node is exactly moved once per each iteration, in this approach, some specific types of moves may be excluded for a number of iterations. This number of iterations depends on an aperiodic

function f and the current iteration i. Accordingly, in each iteration of this algorithm, a non-excluded node with the highest gain will be excluded. If the movement of node v in block A is denoted by (v, A) and it is excluded for $f(i)$ iterations after the movement to the block with the highest gain, then the node cannot be put back to block A for $f(i)$ iterations.

3.4.4 Flow Based Improvement

Flow-based graph partitioning method has been explained in Section 3.3.4. In this section a max-flow min-cut based technique will be discussed proposed by Sanders and Schulz [70, 71] to improve the edge cut of a bipartitioning problem and generalize it to k-partitioning algorithm.

In this algorithm, an $s - t$ flow problem is constructed by growing an area around the given boundary nodes/cut edges. In this area, each $s-t$ cut is related to a feasible bipartition of the given graph (a bipartition fulfills the balance constraint).Then, a max-flow min-cut algorithm can be employed to find a min-cut in this area which is a nondecreased cut between the blocks.

The algorithm can be improved by applying the method iteratively, running searching procedure in a larger areas to find more feasible cuts, or employing a heuristic approach obtain a better balanced minimum cut [13].

3.4.5 Bubble Framework

The bubble framework algorithms [20] are an extension to the graph growing methods explained in Section 3.3.3. Bubble framework is actually an iterative procedure for partitioning a graph into $k > 2$ well-shaped blocks. The good geometric block shapes can be important in many applications, such as convergence rate of some iterative linear solvers.

In this procedure, at first, k seed nodes distributed evenly over the graph will be selected. Furthermore, an iterative improvement will be applied in the second and the third step. Then, starting from the selected k seed nodes, k breadth first searches will be run to grow the blocks, as it explained Section 3.3.3, except that the the BFS are implemented such that the current node is assigned to the smallest block. Afterwards, local search algorithms are used to balance the load of the blocks and to refine the cut of the current partition. Finally, at the end of each iteration, new seed nodes will be computed for the next round. In this step, the new center of a block is the node that minimizes the sum of the distances to all other nodes in the block,

The second and the third step of the algorithm will be iterated till the seed nodes stop changing or no improvement happens for more than 10 iterations. This algorithm is computationally expensive and its complexity is O(km). Figure 3.3 illustrates the three steps of the algorithm.

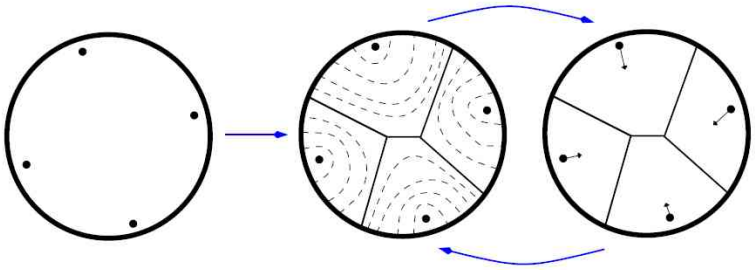

Figure 3.3: The three steps of the Bubble framework [13]

In this figure, the black nodes indicate the seed nodes. In the second step, the BFS procedures has been performed around the seed nodes and a partition has been found. In the third step, new seed nodes are found. Algorithm 3 demonstrates the flowchart of the Bubble framework [20].

Algorithm 3: The Bubble algorithm [20]

Data: $G(V, E)$
Result: $graphpartitions$

Procedure *Bubble GP*
  perform BFS from an element v of minimal degree;
  take an element with furthest distance to v as seed of part 1;
  for $i = 2$ *to* P **do**
    perform BFS from seed $1, ..., i-1$, simultaneously;
    take an element with furthest as seed of part i;
  end
  repeat
    grow parts from seeds in BFS manner;
    calculate centers of parts and assign them as new seeds;
  until *(the seeds do not change OR the aspect ratio (AR) of subdomain did not improve for 10 iterations)*;
end

The explained algorithm has been improved by using distance measures to reflect the graph structure [61, 74]. For instance, diffusion has been applied as growing mechanism around the initial seeds and the method has been then generalized to be applicable for weighted graphs [62]. Some of the diffusion schemes will be discussed in Section 3.4.6.

3.4.6 Random Walks and Diffusion

A random walk on a graph is a procedure starting on a node v and then the next node will randomly be chosen to visit from the set of neighbours based on transition possibilities. Transition probabilities are actually indicates the importance of an

edge. Random walk is an iterative procedure which can be repeated an arbitrary number of times. Transition probabilities are obtained by transition matrix P whose entries are the transition probabilities of the corresponding edges [60].

Diffusion, in general, is a natural process describing a substance's desire to distribute evenly in space. Applying diffusion on a graph is actually implementing an iterative procedure in which splittable entities are exchanged between neighbours, usually until all node have the same amount. Therefore, diffusion can be considered as a special version of random walk.

Both of the random walk and the diffusion algorithm can be employed to identify dense graph regions. Once a random walk reaches to a dense region, it will likely stay there for a longer time, before leaving it through one of the relatively few outgoing edges. $\mathbf{P}^t_{u,v}$ denotes the probability of a random walk that starts in node u to be located on v after t steps. The relative size of this parameter can be considered as a measure for assigning u and v to the same or different clusters [73].

3.5 Multilevel Graph Partitioning

Multilevel graph partitioning approach is one of the most successful heuristic methods in graph partitioning problems [13]. This approach consists of three main phases: coarsening (which is also known as contraction), initial partitioning, and uncoarsening.

The coarsening phase is actually responsible for approximating the original problem and providing a graph with fewer degrees of freedom. In multilevel graph partitioning solvers the coarsening is obtained by constructing a hierarchical coarsened graphs so that the cuts in the coarse graphs indicate the cuts on the original fine graph. The proper hierarchy can be created through a variety of methods. Most of these methods contracts nodes on the fine level. The contracting $U \subset V$ is essentially replacing the set of node U with a single node u so as

$$c(u) = \sum_{w \in U} c(w) \tag{3.1}$$

Contraction may also produce parallel edges which are replaced by a single edge. In this case, the weight of the new edge is the summation of the parallel edges. Therefore, the balanced partitions on the coarse level represent the balanced partitions on the fine level with the same cut values [13]. The coarsening phase has been demonstrated in Figure 3.4.

The coarsening procedure is usually terminated when the graph becomes sufficiently small to be initially partitioned by means of a proper and usually expensive algorithm. In this step, any of the fundamental algorithms discussed in Section 3.3 can be employed to achieve the initial partitioning. It should be noticed that implementing expensive methods and consequently obtaining high quality partitions at the coarsest level does not guarantee obtaining the high quality results in at the finest

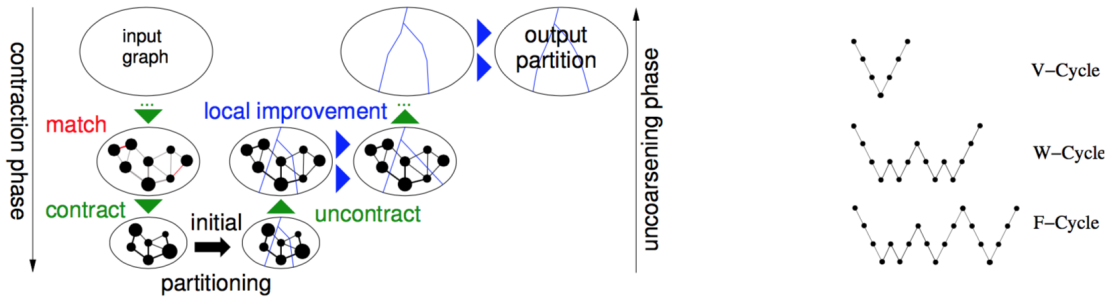

Figure 3.4: The multilevel graph partitioning approach [13]

level. Therefore, some graph partitioning solvers prefer to apply several faster diverse algorithms with random tie breaking instead of running only expensive global optimization procedures [13].

The next step, uncoarsening, includes two stages. The first one is mapping the solution achieved at the coarse level to the fine level graph. The second step is improving the result which is typically obtained by applying one of the methods explained in the Section 3.4. These two abovementioned stages will be carried on until the finest hierarchy level, which is the original inputted graph, has been processed. Each of the coarsening-uncoarsening process is also called V-cycle (Figure 3.4).

There exist some reasons which explains the great performance of the multilevel graph partitioning approach [13]:

- Due to coarsening step, a lot of procedure can be executed on nodes without huge increasing in the overall execution time.
- Any changes on a single node at the coarse level is actually corresponds to a big change in the final solution. Therefore, there is a better chance to find an improvement in the coarse level which is not easy to find in the finest level
- The local improvements in the finest level are faster than a single level approach because they have been already started from a good solution obtained in the coarse level.
- The iterative execution of multilevel partitioning algorithms, such as chains of V-cycle, is a privilege of this approach since provides an opportunity to improve the quality of coarsening by using the previous iteration's solution. Furthermore, the inter-hierarchical coarsening-uncoarsening iteration can also be improved in such way that more processes will be executed ate the coarser levels. An instance of this approach is proposed as W-cycle and F-cycle [70] which has been simply demonstrated in Figure 3.4.
- Multilevel graph partitioning approaches are intrinsically parallelization-schemes friendly since in the multilevel approaches, only the local processing at possibly different coarsening levels construct the global solution.

3.5.1 Coarsening Approaches

Contracting a Single Edge

A fundamental coarsening approach is to contract only two nodes connected by an edge. This approach leads to an almost n levels hierarchy, therefore, it is called n-level graph partitioning [65]. To obtain high quality partitions, a k-way scheme of highly localized local search methods, such as the approaches explained in Section 3.4.1, can be successfully applied.

Contracting a Matching

Contracting large matchings is the most widely used coarsening strategy. In this approach, the contracted sets are actually pairs of nodes which are connected by some non-intersected edges. This scheme leads to a geometrically decreasing size of the graph and a logarithmic number of levels, subsequently.

Coarsening for Scale-free Graphs

The abovementioned matching-based graph coarsening methods may fail to obtain good hierarchies for graphs with irregular structure. In an extreme case, applying matching algorithm on a star-shaped graph can contract only one edge per level that is not desirable in most cases. Therefore, other functional approach should be implemented.

The matching algorithms are suggested to modify in such a way that an unmathched node can be matched with one of the neighbors. In this way, instead of matchings, whole groups of nodes are contracted to create the graph hierarchies. This approach can be applied on graphs with power-law degree distribution.

Another approach implemented to create graph hierarchies for social networks is based on pairwise merging of nodes with the same neighbors and then merging multiple nodes. In this manner, multiple neighbors of a high degree node will collapse with this node.

Flow Based Coarsening

One of the coarsening approaches implemented in a two-level graph partitioning solver for road networks is based on max-flow computations. In this approach, finding the natural cuts dividing determined regions from the remainder of the graph is the first step. Then, the uncut components will be contracted reducing the graph size by up to two orders of magnitude.

Coarsening with Weighted Aggregation

Generally, in the weighted aggregation approach, all nodes at fine level belong to nodes at the coarse level with some probabilities. In the aggregation-based coarsening methods, at first, the nodes of the given fine graph which survive in the coarsened graph will be detected. Afterwards, all other nodes are assigned to these coarse nodes. This approach is established upon Algebraic Multigrid (AMG) methods which are known as a set of hierarchical linear solvers.

3.6 Summary

The formal definition of graph partitioning problem has been presented in this chapter. Moreover, the available graph partitioning algorithms have been discussed. In the presented classification of these methods, global algorithms are introduced as the fundamental methods in graph partitioning which works on the entire graph. It is also explained that, from a practical point of view, these approaches can only be employed for smaller graphs or as subroutines in other algorithms.

Another family of methods for graph partitioning discussed in this chapter is improved heuristic methods that they are widely used in various graph partitioning problems. The key idea of these algorithms is iteratively improvement of an initial solution.

At the end of this chapter, multilevel graph partitioning method is discussed as the most successful heuristic approach in graph partitioning.

Chapter 4

Spectral Graph Partitioning

4.1 Overview

In this chapter, the spectral graph partitioning approaches are discussed in detail. In general, spectral techniques are important in the design and analysis of the algorithms and have lots of applications in the following areas:

- Graph partitioning (sparsest cut, multi-partitioning, small-set expansion),
- Combinatorial optimization (maximum flow, maximum cut)
- Random walk, pagerank, local graph partitioning
- Coding, expander graphs, expander codes

Fundamentally, spectral graph theory is the study of the properties of a graph by analyzing the characteristic polynomial, eigenvalues, and, eigenvectors of the matrices associated to the graph, such as adjacency matrix or Laplacian matrix. Correspondingly, a particular class of graph partitioning algorithms is known as spectral partitioning methods which are mainly based on the eigen-decomposition of Laplacian matrices of either weighted or unweighted graphs [63]. This methods actually reduce the graph partitioning problems to finding an eigenvalue of a symmetric matrix that can be solved by effective well-known algorithms.

In practice, the spectral graph partitioning scheme implements a technique of relaxing a binary-values unknown to a real-valued one. In general, in a bisection problem, each vertex is definitely assigned to one subgraph, either G_1 or G_2. However, in the relaxed graph partitioning problem vertices are considered divisible [75], e.g. 0.25 of one vertex might be assigned to G_1 and the rest 0.75 to G_2. By means of this relaxation approach, a combinatorical optimization problem is actually turned into a numerical optimization problem which can be solved by polynomial-time algorithms.

After solving a relax problem, the real-valued solution should convert to an approximately optimal solution of the original graph partitioning problem. This aim can be obtained by rounding the vertex values by comparing the values with some threshold value and finally assigning each vertex to G_1 or G_2. One of the possible threshold values is 0.5, i.e. if a vertex value is less than 0.5, then it is assigned to G_1. Otherwise, it is assigned to G_2. It is demonstrated that this threshold can lead to an unbalanced cut. Another possible threshold which can provide balanced cut is the median value of vertices [75].

After executing the rounding procedure on the optimal solution of the continuous problem, a discrete partition is obtained which is expected to be nearly optimized. Furthermore, in the spectral partitioning, there is a bound, called Cheeger's inequality, that can provides a guarantee to find a good cut. More specifically, it bounds the quality of the optimal cut obtained by the spectral graph partitioning method. In this chapter, the Cheeger's inequality and its application has also been discussed in details.

4.2 Definitions

The graph partitioning problem has been defined in the previous chapter. In this section some definition used in spectral graph theory will be defined [75].

Considering a partition of G, the *cut* or *edge separator* is defined as the set of edges in E whose endpoints are in different subgraphs. In other words, the partition cuts these edges. In these problems, an edge weight is an indication of how bad it is to cut that edge. In this way, the sum of the weights of all the edges in a cut is defined as the weight of the cut and the graph partitioning problems aims at minimizing this quantity.

A perfectly balanced bipartition is called *bisection* and a perfectly balanced multipartition is recognized as a *multisection*. To obtain a measure of how a graph is balanced, a $mass$ can be assigned to each vertex. The mass of a subgraph is defined as the sum of the masses of its vertices. Thus, a multisection means subgraphs with equal masses. To obtain this aim, a rounding method that minimizes the difference between the masses of subgraphs can be implemented, or as an alternative, the objective functions are optimized in such a way that cut weights is traded against some mass imbalance.

Two objective functions are defined for judging unbalanced partitions. The first one is the *isoperimetric ratio* of a bipartition

$$\frac{\text{Cut}(G_1, G_2)}{\min\{\text{Mass}(G_1), \text{Mass}(G_2)\}} \tag{4.1}$$

where Cut(G_1, G_2) is the weight of the edge separator and Mass(G_i) denotes the total mass of the vertices in G_i. By this definition, the minimum isoperimetric ratio among all cuts of G is known as the *isoperimetric number* of G. If the mass of each

vertes is assigned 1, then Mass$(G_i) = |V_i|$. In this case, the isoperimetric number is also known as the *edge expansion*. Moreover, if the mass of each vertex is defined as the sum of the adjoining edge weights, then the isoperimetric number is also called the *conductance* or the *Cheeger constant* of G.

On account of the abovementioned explanations, Equation (4.1) provides a general definition. It is worthwhile to note that in some references [80], the Cut(G_1, G_2) is also denoted by $|\delta(S)|$, where $\delta(S)$ is the set of edges with one endpoint in $S \subseteq V$ and the other endpoint in $V - S$. In other words, $\delta(S)$ is the edge separator of the partition $(G_1 = S, G_2 = V - S)$. By this specific notation, the edge expansion of a set $S \subseteq V$ can be presented as

$$\frac{|\delta(S)|}{min\{|S|, |V - S|\}} \tag{4.2}$$

Similarly, the conductance of a set $S \subseteq V$ in an unweighted graph can specifically be rewritten as

$$\frac{|\delta(S)|}{\min\{vol(S), vol(V - S)\}} \tag{4.3}$$

where $vol(S)$ is defined as $\sum_{V \in S} deg(V)$.

Accordingly, a subset is considered sparse, if its conductance or expansion is small. In a regular graph, the notion of conductance and expansion will be equivalent. In general, the conductance of a graph is more applicable and is denoted as

$$\phi(G) = \min_{S \subseteq V} \phi(S) = \min \frac{|\delta(S)|}{\min\{vol(S), vol(V - S)\}} = \min_{S \subseteq V, |S| \leq \frac{n}{2}} \frac{\delta(S)}{vol(S)} \tag{4.4}$$

The second objective function is the *sparsity* of a bipartition which is also known as the *uniform sparsity* or *cut ratio* and defined as

$$\frac{\text{Cut}(G_1, G_2)}{\text{Mass}(G_1)\text{Mass}(G_2)} \tag{4.5}$$

By this definition, the cut with minimum sparsity is called the *sparsest cut* or specifically the *uniform sparsest cut*. In this case, if the sum of the adjoining edge weights is considered as the vertex mass, then the sparsest cut and the sparsity are identified as the *normalized cut* and the *normalized cut criterion*, respectively.

Moreover, the *average cut* criterion is another parameter which is

$$\text{Cut}(G_1, G_2) \left(\frac{1}{\text{Mass}(G_1)} + \frac{1}{\text{Mass}(G_2)} \right) \tag{4.6}$$

and equal to the sparsity times Mass(G); therefore, optimizing one is actually equivalent to optimizing the other. This parameter is also equal to the sum of the isoperimetric ratios of the subgraphs G_1 and G_2.

4.3 Scope

In general, spectral graph partitioning algorithms are focused on NP-hard graph partitioning problems which can be well partitioned by spectral algorithms. This problems are including finding the minimum bisection, the cut with minimum isoperimetric ratio, and the cut with minimum uniform sparsity. There exist some other graph partitioning problems that can be solved in polynomial time. For instance, the minimum cut problem aims at finding a cut of minimum weight and it might isolate just a single vertex. This problem can be solved in $O(|V||E| + |V|^2 \log |V|)$ time [75, 83]. There is also the source-target minimum problem in which some vertices are determined to be in G_1 and some others are also determined to be in the other subgraph, G_2. The goal in this problem is to find the minimum cut which satisfies this requirement. By considering the max-flow min-cut theorem [27, 33], this type of problem can be solved by polynomial-time algorithms.

An advantage of the spectral partitioning algorithms is that they can handle negative edge weights. Most of the other graph partitioning approaches are proposed for graphs with positive edge weights and consequently fail for the graphs with negative edge weights. For example, the max-flow min-cut theorem does not hold for the graph with negative edge weights.

4.4 The Laplacian Matrix

Acquaintance with the Laplacian matrix and its properties is the first step in understanding the spectral graph theory and partitioning methods.

Consider an undirected graph $G = (V, E)$ and its weighted matrix $\boldsymbol{W}$, such that $w_{ij} \geq 0$ for $i, j = 1, ..., n$. Then, consider a partition of G with two subgraphs G_1 and G_2. Let $\boldsymbol{x}$ be an n-component *indicator vector* in which $x_i = c_1$ if vertex i belongs to G_1, otherwise, $x_i = c_2$, where c_1 and c_2 are two distinct arbitrary constants. The contribution of an edge (i, j) to the cut weight is calculated by

$$w_{ij}\frac{(x_i - x_j)^2}{(c_1 - c_2)^2} = \begin{cases} w_{ij}, & \text{if } (i,j) \text{ is cut,} \\ 0, & \text{if } (i,j) \text{ is not cut.} \end{cases} \tag{4.7}$$

Therefore, it can be written [75]

$$\text{Cut}(G_1, G_2) = \sum_{(i,j)\in E} w_{ij}\frac{(x_i - x_j)^2}{(c_1 - c_2)^2} = \frac{\boldsymbol{x}^T \boldsymbol{L}\boldsymbol{x}}{(c_1 - c_2)^2} \tag{4.8}$$

where $\boldsymbol{L}$ is a symmetric $n \times n$ matrix called the *Laplacian matrix* of G which can be defined as follows

$$L_{ij} = \begin{cases} -w_{ij}, & i \neq j, \\ \sum_{k\neq i} w_{ik}, & i = j. \end{cases} \tag{4.9}$$

In fact, $\boldsymbol{L}$ is a matrix presentation of a graph. In a formal definition, the unnormalized graph Laplacian matrix $\boldsymbol{L} = [l_{ij}]_{n\times n}$ is obtained by

$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W} \tag{4.10}$$

where $\boldsymbol{D} = [d_{ij}]_{n\times n}$ is a diagonal matrix with $d_{ij} \in \mathbb{R}$ that is defined by $d_{ii} = d_i = \sum_{j=1}^{n} w_{ij}$. In other words, d_i is the degree of node v_i.

In an unweighted graph, the weight matrix will be substituted by the adjacency matrix $\boldsymbol{A}$:

$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A} \tag{4.11}$$

Laplacian matrix is the key feature in the most of the spectral algorithms. Some of the theorems and properties concerning the Laplacian matrix $\boldsymbol{L}$ are as follows [40, 63]:

1. $\boldsymbol{L}$ is symmetric positive-semi-definite matrix.
2. For each $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{x}^T\boldsymbol{L}\boldsymbol{x} = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(x_i - x_j)^2$, where x_i is the i-th component of $\boldsymbol{x}$.
3. The smallest eigenvalue of $\boldsymbol{L}$ is $\lambda_1 = 0$ with the associated indicator eigenvector $\mathbf{1} = (1, ..., 1)^T$.

These main properties can simply be proved [40]. Intuitively, if every weight w_{ij} is non-negative, Equation (4.8) demonstrates implies that $\boldsymbol{x}^T\boldsymbol{L}\boldsymbol{x} \geq 0$ for all $\boldsymbol{x}$ which it means $\boldsymbol{L}$ is positive semidefinite. Therefore, it has no negative eigenvalues.

Furthermore, if G is a connected graph ans all of its edges have positive weight, Equation (4.8) is zero if and only if all the components of $\boldsymbol{x}$ is equal to zero if only if all the components of $\boldsymbol{x}$ are identical. In this case, $\boldsymbol{L}$ has exactly one indicator eigenvector with eigenvalue zero. The indicator eigenvector actually means the cut weight is zero if and only if all vertices are in one subgraph and the other one is empty.

4.5 Graph Partitioning as Constrained Quadratic Optimization

The minimum value of cut weight is achieved, $\boldsymbol{x}^T\boldsymbol{L}\boldsymbol{x} = 0$, when $G_1 = G$ and $G_2 = (\emptyset, \emptyset)$; however, it is not desirable. Therefore, some balance constraint should be imposed.

Let choose the constants $c_1 = 1$ and $c_2 = -1$ to indicate which vertices belongs to G_1 and which ones are in G_2. In this manner, the vector $\boldsymbol{x}$ is called a *signed indicator*

vector. By imposing the following constraint, exactly half of the vertices are in G_1 and half are in G_2,

$$\mathbf{1}^T \boldsymbol{x} = 0. \tag{4.12}$$

This constraint practically dictates that the number of components of $\boldsymbol{x}$ that are 1 must be equal to the number that are -1. By taking this approach, the graph bisection problem is actually reduced to the optimization problem of choosing $\boldsymbol{x}$ to minimize $\boldsymbol{x}^T \boldsymbol{L} \boldsymbol{x}$, subjected to Equation (4.12), and for all i, $x_i = 1$ or -1.

4.6 Bounds on the Weight of a Bisection

The balance constraint (4.12) implies that $\boldsymbol{x}$ should be orthogonal to $\mathbf{1}$, which happens to be an eigenvector of $\boldsymbol{L}$ with eigenvalue zero that is the smallest eigenvalue of $\boldsymbol{L}$ when there exists no negative weight edge. Thus, it can be inferred that there is a relationship between the smallest balanced cut and the second smallest eigenvalue of the Laplacian matrix. This relationship is actually the essence of the spectral graph partitioning algorithm.

Consider $\lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$ as the eigenvalues of $\boldsymbol{L}$ and $\boldsymbol{v_1}, \boldsymbol{v_2}, ..., \boldsymbol{v_n}$ as the corresponding unit eigenvectors. Thus, $\lambda_1 = 0$ and $\boldsymbol{v_1} = (1/\sqrt{n})\mathbf{1}$. Subsequently, let $\boldsymbol{V} = [\boldsymbol{v_1}, ..., \boldsymbol{v_n}] \in \mathbb{R}^{n \times n}$, and Λ be the diagonal $n \times n$ matrix with the eigenvalues in the diagonal positions. Therefore, due to Equation (2.2), it can be written

$$\boldsymbol{L}\boldsymbol{V} = \boldsymbol{V}\boldsymbol{\Lambda} \tag{4.13}$$

As it explained before, the eigenvectors of $\boldsymbol{L}$ are all orthogonal to each other; therefore, the columns of $\boldsymbol{V}$ form and orthonormal basis for $\mathbb{R}^n$. Accordingly, $\boldsymbol{V}$ is called an orthonormal matrix, meaning that $\boldsymbol{V}^T\boldsymbol{V} = \boldsymbol{I}$.

4.6.1 Rayleigh Quotient

The main object is actually reduced to finding an $\boldsymbol{x}$ which minimizes $\boldsymbol{x}^T \boldsymbol{L} \boldsymbol{x}$. The form of the objective function is reminding the *Rayleigh quotient* which, for a given matrix $\boldsymbol{A}$, has been defined as

$$\frac{\boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x}}{\boldsymbol{x}^T \boldsymbol{x}} = \frac{\sum_{i,j} a_{ij} x_i x_j}{\sum_i x_i^2} \tag{4.14}$$

and is the main tool in relating eigenvalues and eigenvectors to optimization problems.

Based on the definition (4.14), there are also two proven results [80]:

- If $\boldsymbol{A}$ is a real symmetric matrix with eigenvalues $\lambda_1 \geq \lambda_2 \geq ... \lambda_n$, and orthonormal eigenvectors $\boldsymbol{v_1}, \boldsymbol{v_2}, ..., \boldsymbol{v_n}$ then

$$\lambda_1 = \max_x \frac{\boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x}}{\boldsymbol{x}^T \boldsymbol{x}} \tag{4.15}$$

This claim is extended to other eigenvalues.

- Let T_k be the set of vectors that are orthogonal to $\boldsymbol{v_1}, \boldsymbol{v_2}, ..., \boldsymbol{v_{k-1}}$, then

$$\lambda_k = \max_{\boldsymbol{x} \in T_k} \frac{\boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x}}{\boldsymbol{x}^T \boldsymbol{x}} \tag{4.16}$$

The latter provides a characterization of λ_k, but it still requires the knowledge of the previous eigenvectors. In order to obtain a characterization without knowing the eigenvectors the following theorem has been presented [80].

Courant-Fischer Theorem

$$\lambda_k = \max_{\substack{\boldsymbol{x} \subseteq \mathbb{R}^n \\ dim(S)=k}} \min_{\boldsymbol{x} \in S} \frac{\boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x}}{\boldsymbol{x}^T \boldsymbol{x}} = \min_{\substack{\boldsymbol{x} \subseteq \mathbb{R}^n \\ dim(S)=n-k+1}} \max_{\boldsymbol{x} \in S} \frac{\boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x}}{\boldsymbol{x}^T \boldsymbol{x}} \tag{4.17}$$

This theorem is more useful in giving bound on eigenvalues and will be employed to prove the related theorems.

By substituting the Laplacian matrix $\boldsymbol{L}$ in the Equation (4.14), $\frac{\boldsymbol{x}^T \boldsymbol{L} \boldsymbol{x}}{\boldsymbol{x}^T \boldsymbol{x}}$, it can be demonstrated that the minimizing the cut weight is actually equivalent to minimizing the Rayleigh quotient, because the components of $\boldsymbol{x}$ is constrained to be 1 or -1 and so the denominator $\boldsymbol{x}^T \boldsymbol{x}$ is always n.

By considering the well known *Rayleigh-Ritz theorem* [80], it can be inferred that the Rayleigh quotient for every non-zero vector $\boldsymbol{x}$ is in the range $[\lambda_1, \lambda_n]$. Furthermore, by imposing Equation (4.12), the Rayleigh quotient is in the range $[\lambda_2, \lambda_n]$. To prove this, one can write $\boldsymbol{x}$ as a linear combination of the unit eigenvectors,

$$\boldsymbol{x} = \boldsymbol{V} \boldsymbol{a}, \ \boldsymbol{a} \in \mathbb{R}^n. \tag{4.18}$$

Then, it can be written

$$n = \boldsymbol{x}^T \boldsymbol{x} = \boldsymbol{a}^T \underbrace{\boldsymbol{V}^T \boldsymbol{V}}_{\boldsymbol{I}} \boldsymbol{a} = \boldsymbol{a}^T \boldsymbol{a}, \tag{4.19}$$

and

$$\boldsymbol{x}^T \boldsymbol{L} \boldsymbol{x} = \boldsymbol{a}^T \boldsymbol{V}^T \boldsymbol{L} \boldsymbol{V} \boldsymbol{a} \overset{(4.13)}{=} \boldsymbol{a}^T \boldsymbol{V}^T \boldsymbol{V} \boldsymbol{\Lambda} \boldsymbol{a} = \boldsymbol{a}^T \boldsymbol{\Lambda} \boldsymbol{a} = \sum_{i=1}^{n} \lambda_i a_i^2 \tag{4.20}$$

The first term of the summation in Equation (4.20) is zero and can be dropped because $\boldsymbol{a} = \boldsymbol{V}^T \boldsymbol{V} \boldsymbol{a} = \boldsymbol{V}^T \boldsymbol{x} \Rightarrow a_1 = \boldsymbol{v_1}^T \boldsymbol{x} = \boldsymbol{1}^T \boldsymbol{x} = 0$. Therefore, the objective function is essentially bounded by the following inequalities

$$\lambda_2 \sum_{i=2}^{n} a_i^2 \leq \boldsymbol{x}^T \boldsymbol{L} \boldsymbol{x} \leq \lambda_n \sum_{i=2}^{n} a_i^2. \tag{4.21}$$

Finally, due to the fact that $\sum_{i=2}^{n} a_i^2 = \boldsymbol{a}^T \boldsymbol{a} = \boldsymbol{x}^T \boldsymbol{x}$, the bounds of the Rayleigh quotient are presented by

$$\lambda_2 \leq \frac{\boldsymbol{x}^T \boldsymbol{L} \boldsymbol{x}}{\boldsymbol{x}^T \boldsymbol{x}} \leq \ \lambda_n. \tag{4.22}$$

Therefore, according to Equation (4.8) and the fact that $c_1 = 1$ and $c_2 = -1$, the bounds of the weight of a bisection can be obtained

$$\frac{\lambda_2 n}{4} \leq \text{Cut}(G_1, G_2) \leq \frac{\lambda_n n}{4}. \tag{4.23}$$

The lower bound is so important because it can inform that there is no small bisection without solving the NP-hard problem of finding the cut.

It can also be claimed that the vector $\sqrt{n}\boldsymbol{v_2}$, with all components supposedly ± 1, gives an optimal bisection. The coefficient $\sqrt{2}$ is appears since $\boldsymbol{v_2}$ is a unit vector with length of 1, but by the length of an indicator vector is $\sqrt{n}$. By substituting $\boldsymbol{x} = \sqrt{n}\boldsymbol{v_2}$ into Equation (4.8), it is achieved that $\text{Cut}(G_1, G_2) = \lambda_2 n/4$, which is the lower bound value. So, it is actually an optimal bisection of the graph.

Even though it is difficult and pragmatically impossible to expect that every components of $\sqrt{n}\boldsymbol{v_2}$ to be 1 or -1, a good cut can be obtained by rounding the vector $\boldsymbol{v_2}$ in the spectral graph partitioning approach. The vector $\boldsymbol{v_2}$ is also called the *Fiedler vector*, since it has been suggested to employ for graph partitioning by Miroslav Fiedler [30, 31].

Fiedler demonstrated that in a special case where all the edge weights are 1 and the graph is not complete , the *vertex connectivity* which is the minimum number if vertices that must be removed to partition the graph into two connected components is at least λ_2. He also showed that regardless the balance condition, the number of edges in the minimum cut is never less than the vertex connectivity [30].

4.7 Spectral Graph Partitioning

4.7.1 The Relaxed Optimization Problem

In the spectral graph partitioning method, the optimization problem described in Section 4.5 can be relaxed by ignoring the constraint $x_i = \pm 1$ and replacing it by a feasible constraint that $\| \boldsymbol{x} \| = \sqrt{n}$. Therefore, the relaxed optimization problem can be presented as [75]

$$\begin{aligned} \text{minimize} \quad & \boldsymbol{x}^T \boldsymbol{L} \boldsymbol{x}, \\ \text{subject to} \quad & \boldsymbol{x}^T \boldsymbol{x} = n \text{ and } \mathbf{1}^T \boldsymbol{x} = 0 \end{aligned} \tag{4.24}$$

It should be noted that the relaxed problem will remain no memory of whether we want a balanced cut or a biased cut, since c_1 and c_2 have fixed values and do not appear in Equation (4.24). It can be a drawback for this approach. Therefore, the approach of balancing can only be decided in the rounding phase.

4.7.2 The Spectral Partitioning Algorithm without Vertex Masses

Let G be a connected, weighted graph. The first step in spectral partitioning method is computing the Laplacian matrix $\boldsymbol{L}$. Afterwards, an eigenvalue $\boldsymbol{v_2}$ corresponding to the second-smallest eigenvalue λ_2 should be calculated. In general, the eigenvector $\boldsymbol{v_2}$ does not consist of two-valued components. On the contrary, the components might be distributed over a range of real values. Hence, these values should be rounded by implementing a wide range of proposed rounding heuristics. Almost all of these approaches are *threshold cuts* in which a threshold value is chosen to distinguish the vertices belong to G_1 from the ones are assigned to G_2. Some of the well-known rounding approaches are investigated in the following.

The simplest rounding approach is the method called *sign cut* or *zero threshold cut* in which a vertex i is assigned to G_1, if the i-th component of $\boldsymbol{v_2}$ is positive, otherwise, it is assigned to G_2. Fiedler [30] demonstrated that if all of the components of $\boldsymbol{v_2}$ are non-zero, then both of the G_1 and G_2 are connected.

The sign cut approach might produce an unbalanced cut. The constrained 4.12 only guarantees that the mean component of $\boldsymbol{x}$ is zero, but it cannot ensure that there are equal numbers of positive and negative components.

An improved approach to obtain a balance cut is employing the *median* component as the threshold of bisecting. This partition is known as the *median cut*. In this method, if several vertices exist with the value equal to the median component, they will be divided equally between the subgraphs.

In practice, there exist some applications in which a near-perfect balanced cut is crucial. For instance, a balanced cut can play an important role in an efficient distribution of parallel computations between a set processors. On the other hand, there are other applications which can tolerate imperfect balance. In these applications, it might be the case in which a much smaller cut is obtained, if a little imperfection in balance is permitted. Most of the divide-and-conquer algorithms can take advantage of this kind of flexibility.

The most well-known approach to select a threshold is the *sweep cut* approach, also known as *criterion cut*. In this approach, at first, the components of $\boldsymbol{v_2}$ should be sorted. Afterwards, the cut weight for every practicable threshold is explicitly computed and the threshold with the smallest cut will be chosen. By swapping through the list of sorted components, the cut changes increasingly. The search can be done in $O(|V| + |E|)$ times.

In addition to the cut weight, the other criteria can be employed in the sweep cut method, such as isoperimetric ratio, the sparsity, or the normalized cut criterion explained in Section 4.2.

Another approach is the related rounding method which leads to partitions called the *jump cut* or *gap cut*. In this approach, after sorting the components of $\boldsymbol{v_2}$, the largest gap between two successive values will be used as the threshold. Typically, the largest gap is restricted to be in the middle third of the sorted list for balance. The

outcome cut is usually less reliable than the sweep cut, but it is easer to implement and program.

Clustering can also be considered as a form of graph partitioning in which the number of subgraph and the size of them are not known in advance and it is commonly determined by the data. Clustering approaches work well, if a graph can be partitioned into subgraphs which have an intense internal connectivity in comparison to the connectivity between subgraphs. In most of the case, the clustering procedure is applied on a weighted graph that represents the strength of connectivity within a set of objects. Thus, in the most common spectral clustering method, the Fiedler vector is computed, Then, its components will be partitioned with a clustering algorithm, such as Lloyd's algorithm for k-means clustering [59, 75].

4.7.3 Vertex Masses

In some application, such as partitioning computations between parallel processors, different processors might represent different computational capacity. This difference can be reflected by assigning different masses to the vertex of the corresponding graph. In this context, a positive mass m_i is assigned to each vertex i. The mass of a graph G is denoted by MASS(G) and defined as the sum of the masses of its vertices. Accordingly, the objective of a partitioning procedure is obtaining two subgraphs G_1 and G_2 with nearly equal mass.

Let the mass matrix $\boldsymbol{M}$ be a diagonal matrix such that $M_{ii} = m_i$. In this way, the balance constraint can be re-written as

$$\mathbf{1}^T \boldsymbol{M}\boldsymbol{x} = 0 \tag{4.25}$$

In the relaxed optimization problem (4.24), the balance constraint (4.12) is implicitly imposed by ignoring the smallest eigenvalue of $\boldsymbol{L}$. By considering the vertices' masses, the quadratic constraint is proposed as [88] $\boldsymbol{x}^T\boldsymbol{M}\boldsymbol{x} = \text{Mass}(G)$. Consequently the relaxed problem can be re-written as

$$\begin{aligned} &\text{minimize} && \boldsymbol{x}^T\boldsymbol{L}\boldsymbol{x}, \\ &\text{subject to} && \boldsymbol{x}^T\boldsymbol{M}\boldsymbol{x} = \text{Mass}(G) \text{ and } \mathbf{1}^T\boldsymbol{M}\boldsymbol{x} = 0 \end{aligned} \tag{4.26}$$

This problem can be solved by computing the eigenvector $\boldsymbol{v_2}$ corresponded to the second-smallest eigenvalue of the symmetric generalized eigensystem [24]

$$\boldsymbol{L}\boldsymbol{v} = \lambda\boldsymbol{M}\boldsymbol{v} \tag{4.27}$$

The generalized Rayleigh quotient is subsequently defined as

$$\frac{\boldsymbol{x}^T\boldsymbol{L}\boldsymbol{x}}{\boldsymbol{x}^T\boldsymbol{M}\boldsymbol{x}} \tag{4.28}$$

In a similar way to Section (4.6.1), It can be demonstrated that for every vector $\boldsymbol{x} \neq \mathbf{0}$ satisfying the balance constraint (4.25), the Rayleigh quotient will be in the range $[\lambda_2, \lambda_n]$. Moreover, the weight of bisection is bounded as

$$\frac{\lambda_2 \text{Mass}(G)}{4} \leq \text{Cut}(G_1, G_2) \leq \frac{\lambda_n \text{Mass}(G)}{4} \tag{4.29}$$

which is implies that the eigenvector $\boldsymbol{x} = \sqrt{\text{Mass}(G)}\boldsymbol{v_2}$ is a solution of the relaxed optimization problem (4.26). In this case, if all the components of $\boldsymbol{x}$ is just ± 1, it describes a minimum bisection because the substitution results in $\text{Cut}(G_1, G_2) = \boldsymbol{x}^T \boldsymbol{L}\boldsymbol{x}/4 = \lambda_2(Mass)(G)/4$, that is the lower bound of (4.29). Similarly, $\boldsymbol{x} = \sqrt{\text{Mass}(G)}\boldsymbol{v_n}$ is a maximum of the relaxed optimization problem, and if all of its components is $\boldsymbol{x}$, it will be a maximum bisection.

4.7.4 The Spectral Partitioning Algorithm with Vertex Masses

Consider a given connected, weighted graph G with vertices' masses. The first step to partition this graph is computing the Laplacian matrix $\boldsymbol{L}$ and the mass matrix $\boldsymbol{M}$. Afterwards, the the eigenvector $\boldsymbol{v_2}$ corresponded to the second-smallest eigenvalue of the generalized eigensystem $\boldsymbol{L}\boldsymbol{v} = \lambda \boldsymbol{M}\boldsymbol{v}$. Finally, a rounding procedure is applied on the eigenvector as described in Section (4.7.2).

4.8 Unbalanced Cuts

The relaxed optimization problem (4.26) is actually a special case of a general discrete optimization problem in which different masses can be assigned to the subgraphs. To study the general discrete optimization problem, let the components of the indicator vector $\boldsymbol{x}$ are restricted to take to values, c_1 or c_2 which are not necessarily ± 1. In this way, according to Equation (4.8), the objective function will be $\boldsymbol{x}^T \boldsymbol{L}\boldsymbol{x} = (c_1 - c_2)^2 \text{Cut}(G_1, G_2)$ in which $(c_1 - c_2)^2$. Therefore, the balance constraint will be $c_1 \text{Mass}(G_1) + c_2 \text{Mass}(G_2) = 0$, According to the balance constraint and the quadratic constraint, it is obtained that [75]

$$c_1 = \sqrt{\frac{\text{Mass}(G_2)}{\text{Mass}(G_1)}}, \quad c_2 = -\sqrt{\frac{\text{Mass}(G_1)}{\text{Mass}(G_2)}}, \tag{4.30}$$

and the relaxed optimization problem can be rewritten as

$$\begin{aligned} \text{minimize} \quad & \boldsymbol{x}^T \boldsymbol{L}\boldsymbol{x}, \\ \text{subject to} \quad & \forall i, x_i = \sqrt{\frac{\text{Mass}(G_2)}{\text{Mass}(G_1)}} \text{ or } x_i = -\sqrt{\frac{\text{Mass}(G_1)}{\text{Mass}(G_2)}} \\ \text{and} \quad & \mathbf{1}^T \boldsymbol{M}\boldsymbol{x} = 0. \end{aligned} \tag{4.31}$$

Furthermore, the objective function can actually be presented as [75]

$$\begin{aligned} \boldsymbol{x}^T \boldsymbol{L}\boldsymbol{x} &= (c_1 - c_2)^2 \text{Cut}(G_1, G_2) \\ &= \text{Mass}(G)^2 \left(\frac{1}{\text{Mass}(G_1)} + \frac{1}{\text{Mass}(G_2)} \right) \text{Cut}(G_1, G_2). \end{aligned} \tag{4.32}$$

Ignoring the constant factor Mass(G), Equation (4.32) demonstrates that the objective function is essentially the average cut criterion described in Section 4.2. It is obvious that if the mass of a subgraph approaches zero, the objective function approaches infinity and penalize the unbalancing partition.

4.8.1 Bounds on the Weight of an Unbalanced Cut

Similar to Section 4.6, by considering Equation (4.31), the bounds on an unbalanced cut can be represented in the following inequalities [75, 80]

$$\frac{\lambda_2}{\text{Mass}(G)} \leq \frac{\text{Cut}(G_1, G_2)}{\text{Mass}(G_1)\text{Mass}(G_2)} \leq \frac{\lambda_n}{\text{Mass}(G)} \tag{4.33}$$

which actually bounds the sparsity of both balanced and unbalanced cuts. Both lower bound and upper bounds are computed during the execution of the spectral partitioning algorithm. If these two values are close to each other, it means the rounded solution is close to the optimal solution.

Another important inequality which bounds the cut weight is Cheeger's inequality discussed in the next section.

4.9 Cheeger's Inequality

The Cheeger's inequality is the most important and algorithmic functional theorem in the spectral graph theory. Practically, this inequality relates the second smallest eigenvalue of the Laplacian matrix to how well a graph is connected, and approximates the sparsest cut of the graph [15]. Before describing this theorem, the normalized matrices of a graph should be defined.

4.9.1 Normalized Matrices

Let $\boldsymbol{A}$ be the adjacency matrix of an undirected graph and λ_1 be its largest eigenvalue. Then, it can be demonstrated that

- $\lambda_1 \leq d_{max}$, where d_{max} denotes the maximum degree in G.
- $\lambda_1 \geq d_{avg}$, where d_{avg} denotes the average degree of G.

There is also a well-known theorem, which shows that if G is a connected undirected graph, then

1. The first eigenvalue is of multiplicity one
2. $|\lambda_i| \leq \lambda_1$ for all i.
3. All entries of the first eigenvector are non-zero and have same sign.

To remove the dependency on the maximum degree of the graph in definition of the Cheeger's inequality, the normalized Laplacian matrix is used. Given an adjacency matrix $\boldsymbol{A}$, the normalized adjacency matrix is defined as

$$\boldsymbol{\mathcal{A}} = \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}} \tag{4.34}$$

where $\boldsymbol{D}$ is the diagonal matrix whose i-th entry is the degree of vertex i. Moreover the normalized Laplacian matrix is defined as

$$\boldsymbol{\mathcal{L}} = \boldsymbol{I} - \boldsymbol{\mathcal{A}} = \boldsymbol{D}^{-\frac{1}{2}}(\boldsymbol{D} - \boldsymbol{A})\boldsymbol{D}^{-\frac{1}{2}} = \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{L}\boldsymbol{D}^{-\frac{1}{2}} \tag{4.35}$$

Now, consider $\alpha_1 \geq \alpha_2 \geq ... \geq \alpha_n$ as the eigenvalues of $\boldsymbol{\mathcal{A}}$, and $\beta_1 \leq \beta_2 \leq ... \leq \beta_n$ as the eigenvalues of $\boldsymbol{\mathcal{L}}$. Then, by employing the above-mentioned facts, it can be illustrated that

$$\begin{aligned} &1 = \alpha_1 \geq \alpha_n \geq -1, \\ &0 = \beta_1 \leq \beta_n \leq 2 \end{aligned} \tag{4.36}$$

Accordingly, let $0 \leq \lambda_1 \leq \lambda_2 \leq ...\lambda_n \leq 2$ be the eigenvalues of the normalized Laplacian matrix. As mentioned earlier, $\lambda_2 = 0$ if and only if the graph is disconnected. The Cheeger's inequality states that λ_2 is small if and only if the graph is close to disconnected, i.e. there is a sparse cut in the graph. In other words, a large λ_2 means the graph is well-connected.

4.9.2 The Theorem (Cheeger's Inequality)

In general, Cheeger's inequality is defined as

$$\frac{\lambda_2}{2} \leq \phi(G) \leq \sqrt{2\lambda_2} \tag{4.37}$$

Proving this theorem is important in many other theorems. First, the left hand side of this inequality, $\phi(G) \geq \frac{\lambda_2}{2}$ which is known as the easy direction, will be proved. Based on Equation (4.16), it can be written

$$\lambda_2 = \min_{\boldsymbol{x} \perp \boldsymbol{v_1}} \frac{\boldsymbol{x}^T\boldsymbol{\mathcal{L}}\boldsymbol{x}}{\boldsymbol{x}^T\boldsymbol{x}} = \min_{\boldsymbol{x} \perp \boldsymbol{v_1}} \frac{\boldsymbol{x}^T\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{L}\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{x}}{\boldsymbol{x}^T\boldsymbol{x}} \tag{4.38}$$

By setting $\boldsymbol{y} = \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{x}$ and substituting $\boldsymbol{x} = \boldsymbol{D}^{\frac{1}{2}}\boldsymbol{y}$, it is obtained

$$\lambda_2 = \min_{\boldsymbol{D}^{\frac{1}{2}}\boldsymbol{y}\perp\boldsymbol{v_1}} \frac{\boldsymbol{y}^T\boldsymbol{\mathcal{L}}\boldsymbol{y}}{\left(\boldsymbol{D}^{\frac{1}{2}}\boldsymbol{y}\right)^T\left(\boldsymbol{D}^{\frac{1}{2}}\boldsymbol{y}\right)} = \min_{\boldsymbol{D}^{\frac{1}{2}}\boldsymbol{y}\perp\boldsymbol{v_1}} \frac{\boldsymbol{y}^T\boldsymbol{L}\boldsymbol{y}}{\boldsymbol{y}^T\boldsymbol{D}\boldsymbol{y}} \tag{4.39}$$

Note that $\boldsymbol{v_1} = \boldsymbol{D}^{\frac{1}{2}}\mathbf{1}$, therefore $\boldsymbol{\mathcal{L}}\left(\boldsymbol{D}^{\frac{1}{2}}\mathbf{1}\right) = \left(\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{L}\boldsymbol{D}^{-\frac{1}{2}}\right)\left(\boldsymbol{D}^{\frac{1}{2}}\mathbf{1}\right) = \boldsymbol{D}^{-\frac{1}{2}}L\mathbf{1} = 0$. Thus, $\boldsymbol{D}^{\frac{1}{2}}\mathbf{1} \perp \boldsymbol{v_1}$ is equivalent to $\left\langle \boldsymbol{D}^{-\frac{1}{2}}, \boldsymbol{v_1}\right\rangle = \left\langle \boldsymbol{D}^{-\frac{1}{2}}, \boldsymbol{D}^{\frac{1}{2}}\mathbf{1}\right\rangle = \sum_{i=1}^{n} d_i y_i = 0$, and therefore,

$$\lambda_2 = \min_{\boldsymbol{y}=\sum d_i y_i = 0} \frac{\boldsymbol{y}^T\boldsymbol{L}\boldsymbol{y}}{\boldsymbol{y}^T\boldsymbol{D}\boldsymbol{y}} = \min_{\boldsymbol{y}=\sum d_i y_i = 0} \frac{\sum_{e=ij}(y_i - y_j)^2}{\sum_{i=1}^{n} d_i y_i^2} \tag{4.40}$$

To specify and upper bound on λ_2, it should only be found a vector $\boldsymbol{y}$ giving a set S of small conductance. Preferably, it is chosen $y_i = 1$, if $i \in S$, and $y_i = 0$ if $i \notin S$. Therefore,

$$\frac{\sum_{e=ij}(y_i - y_j)^2}{\sum_{i=1}^{n} d_i y_i^2} = \frac{|\delta(S)|}{vol(S)} \tag{4.41}$$

But the problem is that $\sum_{i=1}^{n} d_i y_i \neq 0$. To solve this problem, $\boldsymbol{y}$ can be modified to

$$y_i = \begin{cases} \frac{1}{vol(S)}, & \text{if } i \in S \\ \frac{-1}{vol(V-S)}, & \text{if } i \notin S \end{cases} \tag{4.42}$$

By substituting this definition in Equation (4.40), the upper bound will be

$$\lambda_2 \leq |\delta(S)| \frac{2m}{vol(S)vol(V-S)} \tag{4.43}$$

and, finally, due to $vol(V-S) \geq m$,

$$\lambda_2 \leq 2\phi(G) \tag{4.44}$$

Equation (4.44) demonstrates that λ_2 is small, if there is a sparse cut in the graph.

The right hand side of Equation (4.37) is known as difficult direction and declares if λ_2 is small, then there is a sparse cut in the graph. The small λ_2 means that there exists $\boldsymbol{y}$ such that $\sum_{i=1}^{n} d_i y_i = 0$ and $\lambda_2 = \frac{\sum_{e=ij}(y_i - y_j)^2}{\sum_{i=1}^{n} d_i y_i^2}$. In this case, a rounding algorithm is needed to round the fractional values of second eigenvectors into integral values. The following theorem is the basis of the proof.

Theorem: Given any $\boldsymbol{y}$ with $R(\boldsymbol{y}) = \frac{\sum_{e=ij}(y_i - y_j)^2}{\sum_{i=1}^{n} d_i y_i^2}$, there exists a subset $S \subseteq \text{supp}(\boldsymbol{y})$ with $\frac{|\delta(S)|}{\text{vol}(S)} \leq \sqrt{2R(\boldsymbol{y})}$, where supp($\boldsymbol{y}$) is defined as $\{i | y_i \neq 0\}$.

Algorithmically, at most, n values are needed to try for $t = y_i^2$ for $1 \leq i \leq n$. It is demonstrated that the above theorem always works, if vol(supp($\boldsymbol{y}$)) $\leq m$. Therefore, if $\boldsymbol{y}$ is considered the second eigenvector of the Laplacian matrix, then the rest of procedure can be summarized as following:

1. Finding c such that $vol(\{i|y_i < c\}) \leq m$ and $vol(\{i|y_i > c\}) \leq m$. Using the vector $\boldsymbol{z} = \boldsymbol{y} - c\mathbf{1}$. It has been known that $R(\boldsymbol{z}) \leq R(\boldsymbol{y})$.

2. Constructing $\boldsymbol{z}^+$ and $\boldsymbol{z}^-$ by employing following equation.

$$z_i^+ = \begin{cases} z_i, & \text{if } z(i) \geq 0 \\ 0, & \text{otherwise} \end{cases}$$
$$z_i^- = \begin{cases} z_i, & \text{if } z(i) \leq 0 \\ 0, & \text{otherwise} \end{cases} \tag{4.45}$$

   It is also known that $min\{R(\boldsymbol{z}^+), R(\boldsymbol{z}^-)\} \leq R(\boldsymbol{z})$, and $vol(supp(\boldsymbol{z}^+)) \leq m$ and $vol(supp(\boldsymbol{z}^-)) \leq m$.

3. Applying a rounding algorithm on $\boldsymbol{z}^+$ or $\boldsymbol{z}^-$. Then a set S can be found such that $\phi(S) \leq \sqrt{2min\{R(\boldsymbol{z}^+), R(\boldsymbol{z}^-)\}}$ and $vol(S) \leq m$, as required.

4.9.3 Notes on the Cheeger's Inequality

The Cheeger's inequality provides an approximation to the conductance of a graph. It gives a good approximation, when $\phi(G)$ is large. Otherwise, it may not give a proper approximation.

The sparsest cut problem is one of the most important problems in approximation algorithms. This is an $O(\log n)$ approximation algorithm by implementing linear programming and can be an $O(\sqrt{\log n})$ by using semidefinite programming. It is basically assumed that there is no constant factor approximation for this problem [80, 55].

This inequality is so important for following reasons:

- First, it provides an approximation independent of the graph size.
- Second, it gives a useful relation between conductance/expansion and the spectral gap, which is closely related to the convergence rate of the random walk. This relation is greatly functional in bounding the mixing time of random walk which is an important parameters in probability and algorithm design.
- Third, it gives a useful tool in constructing expander graphs, very well-connected sparse graphs, which are extremely useful in many areas in computing science.

It is also worthwhile to note that for a planar graph (Section 2.2) G with n vertices of maximum degree d, and λ_2 as the second eigenvalue of the corresponding Laplacian matrix, it can be demonstrated that [80]

$$\lambda_2 \leq \frac{8d}{n} \tag{4.46}$$

4.10 Maximum Cut Problem

4.10.1 Last Eigenvalue

After discussing about the first and the second eigenvalue of the Laplacian matrix, the other end of the spectrum will be discussed in this section. In short, the last eigenvalue and eigenvector can be employed to achieve a non-trivial approximation algorithm for the maximum cut problem. Furthermore, the ratio of the first eigenvalue to the last one can be used to give a bound on the size of maximum independent set.

4.10.2 Maximum Cut

The maximum cut problem is one of the graph cut problems. The analysis of this problem is similar to the proof of the Cheeger's inequality, both are due to the Trevisan [87]. Given an undirected graph $G = (V, E)$, the maximum cut problem is to find $S \subseteq V$ such that $|\delta(S)|$ is maximized.

When $G = (X, Y, E)$ is bipartite, then all the edges can be cut as $|\delta(X)| = |\delta(Y)| = |E|$. Let $\boldsymbol{\mathcal{A}}$ be the normalized adjacency matrix (Equation (4.34)) of G and $\alpha_1 \geq \alpha_2 \geq ... \geq \alpha_n$ as its eigenvalues. It can be shown that $\alpha_1 = -\alpha_n$ if and only if G is bipartite. It is also can be shown that α_1 is close to α_n if and only if G is close to bipartite. Travisan [87] demonstrated that α_1 is close to α_n if and only if G has a subgraph close to bipartite.

For the maximum cut, the objective function is $\frac{|\delta(S)|}{|E|}$. Therefore, the objective value is in $[0, 1]$. It is defined

$$\beta(G) = \min_{y \in \{-1,0,+1\}^V} \frac{\sum_e |y_i + y_j|}{\sum_i d_i |y_i|} \tag{4.47}$$

Alternatively, It can be written

$$\beta(G) = \min_{\substack{S \subseteq V \\ (L,R) \ partitions \ of \ S}} \frac{2\text{edges within}(L) + 2\text{edges.within}(R) + |\delta(S)|}{vol(S)} \tag{4.48}$$

It should be noticed that $\beta(G)$ is small if and only if G has a subgraph S close to bipartite. As it explained in Section 4.9.1, the eigenvalues of $\boldsymbol{\mathcal{A}}$ are in $[-1, +1]$, and the largest eigenvalue is $+1$. In this section, the matrix $\boldsymbol{I} + \boldsymbol{\mathcal{A}}$ has been introduced, thus, the eigenvalue of this matrix will obviously be in the range of $[0, 2]$. Accordingly, if $\lambda_1 \geq \lambda_2 \geq, ..., \lambda_n$ is the eigenvalues of $\boldsymbol{I} + \boldsymbol{\mathcal{A}}$, the condition $\alpha_1 \approx -\alpha_n$ can be recognized as $\lambda_n \approx 0$.

4.10.3 Theorem (Trevisan)

$$\frac{\lambda_n}{2} \leq \beta(G) \leq \sqrt{2\lambda_n} \tag{4.49}$$

This statement and its proof is very similar to that of Cheeger's inequality and has been explained in [87]. Due to the easy direction, $\frac{\lambda_n}{2} \leq \beta(G)$, if G has a subgraph close to bipartite, then λ_n is small. Moreover, if G has a maximum cut of ratio greater than $1-\epsilon$, then $\beta(G) \leq \epsilon$ and $\lambda_n \leq 2\epsilon$.

4.10.4 Approximation Algorithm

In this section, a non-trivial approximation algorithm for max cut problem has been discussed. It is easy to find a cut with at least $|E|/2$ edges, either greedy or randomly. After obtaining $y \in \{-1, 0, +1\}^V$ with $\frac{\sum_{e=ij}|y_i+y_j|}{\sum_i d_i|y_i|} \leq \sqrt{2\lambda_n}$, a subset $S \subseteq V$ and a partition (L, R) of S can be attained such that

$$\frac{2|\delta(L,L)| + 2|\delta(R,R)| + |\delta(S)|}{vol(S)} \leq \sqrt{2\lambda_n} \tag{4.50}$$

where $\delta(X, Y)$ is the set of edges with one endpoint in X and one endpoint in Y (X and Y could overlap). At first, L and R are fixed on two sides by getting all the edges in $\delta(L, R)$. Then the max cut problem will recursively solved on $G - S$, and a placement of L and R will be chosen such that at least half of the edges in $\delta(S)$ is gotten. Afterwards, according to Equation (4.50), it can be written:

$$\frac{2|\delta(L,L)| + 2|\delta(R,R)| + |\delta(S)|}{2|\delta(L,L)| + 2|\delta(R,R)| + |\delta(L,R)| + |\delta(S)|} \leq \sqrt{2\lambda_n} \tag{4.51}$$

Therefore,

$$\frac{|\delta(L,R)|}{|\delta(L,L)| + |\delta(R,R)| + |\delta(L,R)| + \frac{1}{2}|\delta(S)|} \geq 1 - \sqrt{2\lambda_n} \tag{4.52}$$

it can finally be obtained

$$\frac{|\delta(L,R)| + \frac{1}{2}|\delta(S)|}{|\delta(L,L)| + |\delta(R,R)| + |\delta(L,R)| + |\delta(S)|} \geq 1 - \sqrt{2\lambda_n} \tag{4.53}$$

Therefore, if $\lambda_n \geq \theta$ (the value of θ will be determined), then $1 - \sqrt{2\lambda_n}$ is too small and the output is a cut which has at least half of the edges. To determine the θ, when $\lambda_n \geq \theta$, it implies that $\beta(G) \geq \frac{\theta}{2}$ and in particular

$$OPT(G) \leq 1 - \frac{\theta}{2} \tag{4.54}$$

Therefore, cutting half the edges is a $\frac{0.5}{1-\frac{\theta}{2}} = \frac{1}{2-\theta}$ approximation algorithm.

4.11 More Eigenvalues

So far, the first, the second and the last eigenvalues of the Laplacian matrix has been discussed. In this section the other eigenvalues will be studied. Again, consider $0 \leq \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$ as the eigenvalues of the Laplacian matrix of G. It has been known that $\lambda_k = 0$ if and only if G has at least k connected components [55, 36]. The generalization of this statement are as follows:

1. λ_k is close to zero indicates that there is a sparse cut of size $\frac{n}{k}$. This statement is recognized as the small-set expansion problem.

2. λ_k is close to zero if and only if there are k disjoint sparse cuts in G. This statement is also called multi-partitioning problem.

4.11.1 Small-set Expansion

Fundamentally, the small-set expansion problem is to distinguish between the following two cases:

1. All small sets of size δn have high conductance ($\phi(G) \geq 1 - \epsilon$);

2. Some small set of size δn has low conductance ($\phi(G) \leq \epsilon$).

It is worth to notice that this problem has a close connection to the unique games conjecture [69] which if true would imply optimal approximation results for many problems including the maximum cut problem discussed in section 4.10.2.

Accordingly, as an important open question, the approximability of this problem is noticed in various studies [68]. Generally, there is a conjecture that for any ϵ there exists δ such that it is NP-hard to distinguish between the two abovementioned cases. This conjecture, if true, would entail the unique games conjecture [69].

AS mentioned in Section 4.9, Cheeger's inequality states that λ_2 is close to λ_1 if and only if there is a sparse cut, but it does not provide any other information about the size of the sparse cut. For instance, there is a small gap between λ_1 and λ_2 in the hypercube, but all small sets have high conductance [55].

By using this theorem, it has been shown that if λ_l is close to λ_1, then there is a sparse cut of the size close to $\frac{n}{l}$. This result will be led to a subexponential time algorithm for approximation the small-set expansion problem [8].

Assumptions

- The graph is d-regular

- The matrix $\frac{1}{2}\boldsymbol{I} + \frac{1}{2}\boldsymbol{\mathcal{A}}$ is considered, where $\boldsymbol{\mathcal{A}}$ is the normalized adjacency matrix with the eigenvalues $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_n$.
- For $S \subseteq V$, the measure of S is defined as $\mu(S) = \frac{|S|}{n}$, where $|V| = n$.

Theorem (Arora, Barak, Steurer) [8]: Suppose $\phi(S) \geq \epsilon$ for every S with $\mu(S) \leq \delta$. Then, for every even $k \geq 2$,

$$\sum_{i=1}^{n} \lambda_i^k \leq \max\left\{ n. \left(1 - \frac{\epsilon^2}{16}\right)^k, \frac{4}{\delta} \right\} \tag{4.55}$$

Applications: This theorem shows that if the rank of the top eigenspace is large, then there is a small set with small conductance. Otherwise, if there is a small sparse cut S, its projection to the top eigenspace is large. Therefore, there is a vector in the top eigenspace which is similar to the characteristic vector of S. Now, the top eigenspace is of small rank, and so an approximate exhaustive search can be applied on the top eigenspace. Then, a set with small symmetric difference to S can be recovered. Finally, a set with the most δn vertices and small conductance can be obtained [8].

4.11.2 Multi-Partitioning

The second result of a small λ_k is discussed in this section. If λ_k of the Laplacian matrix is close to zero, then there are k disjoint subsets $S_1, S_2, ..., S_k$ such that $\phi(S_i)$ is small for every i [55].

There exists a geometric heuristic for graph partitioning: embed each node i to a k-dimensional point $(\boldsymbol{v_1}(\boldsymbol{i}), \boldsymbol{v_2}(\boldsymbol{i}), ..., \boldsymbol{v_k}(\boldsymbol{i}))$ where $\boldsymbol{v_1}, \boldsymbol{v_2}, ..., \boldsymbol{v_k}$ are the first k eigenvectors, then some geometric clustering algorithm is employed to partition the vertices. This *spectral embedding* provides a nice presentation of graphs (see Section 4.12).

If λ_k is close to zero, then there are k orthonormal eigenvectors with small Rayleigh quotient. In view of these orthonormal eigenvectors, it is expected that each of them defines a unique sparse cut. Therefore, k disjoint sparse cuts can be found.

Theorem (Lee, Oveis Gharan, Trevisan) [55]: There is a polynomial time algorithms to find k disjoint subsets $S_1, S_2, ..., S_k \subseteq V$ such that $\phi(S_i) \leq O(k^3)\sqrt{\lambda_k}$.

In a quick comparison to the previous theorem, this theorem provides a more functional results since it gives many sparse cuts (which one of them is small) and it is also works for small k. On the other hand, the previous theorem presents a stronger bound on the conductance which can be important for small-set expansion.

Proof Outline

- The most important fact is that $\boldsymbol{v_1}, \boldsymbol{v_2}, ..., \boldsymbol{v_k}$ are orthonormal which is noticed in the isotropy property

$$\sum_{v \in V} < x, F(v) >^2 = 1 \text{ for any unit vector } x \in \mathbb{R}^k \tag{4.56}$$

where $F(i) = (v_1(i), v_2(i), ..., v_k(i))$ is the spectral embedding of the vertex i. Furthermore, it is demonstrated that $\sum_{v \in V} \parallel F(v) \parallel^2 = k$, where $\parallel F(v) \parallel^2$ is defined as ℓ^2 mass. Accordingly, the isotropy property implies that the ℓ^2 mass cannot concentrate on less than k directions; therefore, there exist k disjoint clusters.

- The distance of two vertices is defined as

$$d_F(u, v) = \left\| \frac{F(u)}{\parallel F(u) \parallel} - \frac{F(v)}{\parallel F(v) \parallel} \right\| \tag{4.57}$$

The isotropy property entails that close vertices cannot contain too much mass. Thus, it is reasonable to put the close points to the same cluster until the cluster has enough mass. In a visual manner, points are grouped based on their directions

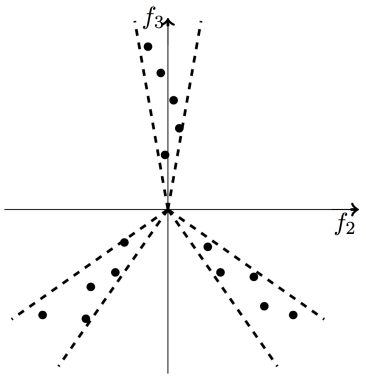

Figure 4.1: Partitioning according to the radial distance [55]

Assume there are k clusters, $S_1, ..., S_k$, each formed by close points, and each has enough mass, supposedly

$$\sum_{v \in S_i} \parallel F(v) \parallel^2 \geq \frac{1}{k} \sum_{v \in V} \parallel F(v) \parallel^2 \tag{4.58}$$

It is also defined

$$\psi_i(v) = \begin{cases} F(v), & \text{if } v \in S_i \\ 0, & \text{otherwise} \end{cases} \tag{4.59}$$

Then $\psi_1, \psi_2, ..., \psi_k$ would have disjoint supports. Consider the Rayleigh quotient

$$\frac{\sum\limits_{(u,v)\in E} \parallel \psi_i(u) - \psi_i(v) \parallel^2}{\sum\limits_{v\in V} \parallel \psi_i(v) \parallel^2} = \frac{\sum\limits_{(u,v)\in E, u\in S, v\in S} \parallel F(u) - F(v) \parallel^2 + \sum\limits_{(u,v)\in E, u\in S, v\notin S} \parallel F(u) \parallel^2}{\sum\limits_{v\in S} \parallel F(u) \parallel^2} \tag{4.60}$$

Besides,

$$\frac{\sum\limits_{(u,v)\in E} \parallel F(u) - F(v) \parallel^2}{\sum\limits_{v\in V} \parallel F(v) \parallel^2} \leq \lambda_k \tag{4.61}$$

Then, according to the fact

$$\sum_{v\in S} \parallel F(v) \parallel^2 \geq \frac{1}{k} \sum_{v\in V} \parallel F(v) \parallel^2, \tag{4.62}$$

it can be inferred that the denominator is still large. Therefore, it is only needed to demonstrate that the numerator is small. The first term of the numerator is bounded by $\sum_{(u,v)\in E} \parallel F(u) - F(v) \parallel^2$, but the second term is not bounded. The main idea is that if clusters are far apart from each other, then it is possible to define a smooth cut-off so that the edge-by-edge ratio is bounded and the property that the supports are disjoint is still maintained.

- Employing a technique in metric embedding, called low-diameter decomposition, one can partition the points into different groups, in such a manner that each group is of low diameter, and most of the points are not close to the boundary. In this case, the points are randomly partitioned based on their directions and dropped them close to the boundaries, and taking groups until there is enough mass to be called a cluster. Then, the procedure will be repeated until achieving k clusters that are far apart from each other. In this way, k functions will be obtained that have disjoint supports, each with small Rayleigh quotient. Afterwards, applying the randomized rounding procedure provides k disjoint sparse cuts.

The proof employs four prominent lemmas, which are briefly described in the following:

1. **Isotropy and Energy Spreading**

   F is considered (Δ, η)-spreading, if for all subsets S of V,

   $$diam(S, d_F) \leq \Delta \Rightarrow \sum_{v\in S} \parallel F(v) \parallel^2 \leq \eta \sum_{v\in V} \parallel F(v) \parallel^2 \tag{4.63}$$

   Lemma 1 Let $\boldsymbol{v_1}, \boldsymbol{v_2}, ..., \boldsymbol{v_k}$ be orthonormal vectors and $F(u) = (\boldsymbol{v_1}(u), \boldsymbol{v_2}(u), ..., \boldsymbol{v_k}(u))$, then F is $\left(\Delta, \frac{1}{k(1-\Delta)^2}\right)$-spreading.

2. **Smooth Localization**

   For every pair $u, v \in V$, using the definition of $d_F(u, v)$ as the distance for an edge (u, v), the shortest path distance from u to v is defined by $\hat{d}_F(u, v)$. By this definition, the ϵ-neighborhood of S is

$$N_\epsilon(S) = \left\{ x \in V : \hat{d}_F(x, S) < \epsilon \right\} \tag{4.64}$$

   The following lemma demonstrates how to perform the smooth cut-off in such a way that the edge-by-edge contribution is bounded if it is allowed to go ϵ-far from S.

   Lemma 2 For any subset $S \subseteq V$, it can be defined a mapping $\psi : V \to \mathbb{R}^k$ with the following properties:

   (a) $\psi(v) = F(v)$ for all $v \in S$.

   (b) $supp(\psi) \subseteq N_\epsilon(S)$.

   (c) $\| \psi(x) - \psi(y) \| \leq \left(1 + \frac{2}{\epsilon}\right) \| F(x) - F(y) \|$ for $x, y \in E$.

3. **Disjoint Separated Regions**

   Now, if disjoint subsets can be found such that each has enough mass, and each pairs is far apart, then the following lemma will complete the puzzle.

   lemma 3 Let $S_1, S_2, ..., S_k$ be disjoint subsets of V, such that $\hat{d}_F(S_i, S_j) \geq \beta$ and

$$\sum_{v \in S_i} \| F(v) \|^2 \geq \delta \sum_{v \in V} \| F(v) \|^2, \tag{4.65}$$

   then, there are $\psi_1, \psi_2, ..., \psi_k : V \to R$ such that

$$\frac{\sum\limits_{(u,v)\in E} \| \psi_i(u) - \psi_i(v) \|^2}{\sum\limits_{v \in V} \| \psi_i(v) \|^2} \leq \frac{1}{\delta}\left(1 + \frac{4}{\beta}\right)^2 \frac{\sum\limits_{(u,v)\in E} \| F(u) - F(v) \|^2}{\sum\limits_{u \in V} \| F(u) \|^2} \leq \frac{1}{\delta}\left(1 + \frac{4}{\beta}\right)^2 \lambda_k \tag{4.66}$$

4. **Random Partitioning**

   A random partition is called (Δ, α, δ)-padded, if $diam(S) \leq \Delta$ for each $S \in P$, and for each point x, the probability that the distance from x to the boundary is at least Δ/α. Accordingly, the following theorem from metric embedding is applied.

   Theorem: For every $\Delta > 0$ and $\delta > 0$, there is a $(\Delta, O(k/\delta), 1 - \delta)$-padded random partition.

   Employing this theorem, the points close to the boundaries can be eliminated, with only a small loss in mass. Then, each group in the partition is far apart

from the other groups. Therefore, k clusters with enough mass can be obtained by taking disjoint unions of groups.

Lemma 4 Let $F : V \to \mathbb{R}^k$ be $\left(\Delta, \frac{1}{k} + \frac{1}{8k^2}\right)$-spreading. Let P be a $\left(\Delta, \alpha, 1 - \frac{1}{4k}\right)$-padded random partition. Then, there exist k disjoint subsets $S_1, S_2, ..., S_k \subseteq$ such that

$$\hat{d}_F(S_i, S_j) \geq 2\frac{\Delta}{\alpha} \text{ for } i \neq j \tag{4.67}$$

and

$$\sum_{v \in S_i} \parallel F(v) \parallel^2 \geq \frac{1}{2k} \sum_{v \in V} \parallel F(v) \parallel^2 \tag{4.68}$$

Summarizing the Procedure

The proof of the theorem can be obtained by putting the lemmas together:

- Let $\boldsymbol{v_1}, ..., \boldsymbol{v_k}$ be the first k eigenvectors. Define $F(x) = (\boldsymbol{v_1}(x), ..., \boldsymbol{v_k}(x))$.
- Set $\Delta \approx \frac{1}{\sqrt{k}}$, such that

$$\left(1 - \frac{\Delta^2}{2}\right)^{-1} \leq 1 + \frac{1}{8k} \tag{4.69}$$

- By Lemma 1, F is $\left(\Delta, \frac{1}{k} + \frac{1}{8k^2}\right)$-spreading. Set $\alpha \approx k^2 \quad (\delta = \frac{1}{k})$. Apply the theorem on low-diameter decomposition to achieve the random partition P required by Lemma 4, with

$$\hat{d}_F(S_i, S_j) \geq \frac{\Delta}{k^2} \tag{4.70}$$

- Apply Lemma 3 with $\delta = \frac{1}{2k}$ and $\beta = \frac{\Delta}{k^2} \approx \frac{1}{k^{2.5}}$ to obtain k disjointly supported functions $\psi_1, \psi_2, ..., \psi_k : V \to R$, such that

$$\frac{\sum\limits_{(u,v) \in E} \parallel \psi_i(u) - \psi_i(v) \parallel^2}{\sum\limits_{v \in V} \parallel \psi_i(v) \parallel^2} \leq \frac{1}{\delta}\left(1 + \frac{4}{\beta}\right)^2 \lambda_k = O(k^6).\lambda_k \tag{4.71}$$

- Employing the randomized rounding procedure, k disjoint subsets are obtained, each with conductance $O(k^3).\sqrt{\lambda_k}$

4.12 k-way Partitioning

The spectral graph partitioning methods for bipartitioning problems has thoroughly discussed. In this section, the spectral methods for partitioning a graph into k subgraphs will be presented.

The most common approach to partition a graph into more than two subgraphs is recursive partitioning that the main approach is generally explained in Section 3.4.2. But, this method is not the most reliable option because the final solution is highly dependent on the first cut.

There exist different other methods for k-way partitioning, such as vector partitioning, spectral clustering by spectral vector directions, and by spectral vector rotation.

The vector partitioning method is the most flexible approach and it will be studied in this section. In essence, this approach employs an arbitrary number of eigenvectors to cut a graph into an arbitrary number of subgraphs.

4.12.1 Vector Partitioning

The main idea of vector partitioning method is using multiple eigenvectors, and not just the Fiedler vector. In this approach, each vertex will be assigned to a spectral coordinates in a k-dimensional space by utilizing k eigenvectors of the Laplacian matrix. As a simple instance, Figure 4.2 illustrates a sample 4-regular graph that its vertices are positioned in a 2-dimensional space by using two eigenvectors of the Laplacian matrix as their spectral coordinates. After assigning spectral coordinates, the points can be partitioned by a geometric partitioning algorithm. This method is also functional to obtain a better bipartitioning solution.

Employing eigenvectors has been proposed to reduce the graph partitioning problem to vector partitioning, which is a discrete optimization problem, as well. It is also NP-hard, but heuristic approaches efficiently work on this problem.

For a bipartitioning problem let the indicator vector $\boldsymbol{x}$ be a binary vector in which $x_i = 1$ means vertex i belongs to subgraph G_1. Consider $\boldsymbol{V} = [\boldsymbol{v_1}, ..., \boldsymbol{v_n}]$ as a $n \times n$ matrix including the eigenvectors as its columns and $\boldsymbol{\Lambda}$ as a diagonal $n \times n$ matrix including the a corresponding eigenvalues. In this way, the eigensystem can be described as the follows,

$$\begin{aligned} \boldsymbol{LV} &= \boldsymbol{MV\Lambda} \\ \boldsymbol{V}^T\boldsymbol{MV} &= \boldsymbol{I} \end{aligned} \tag{4.72}$$

By writing the indicator vector as a linear combination of eigenvectors, $\boldsymbol{x} = \boldsymbol{Va}$, and

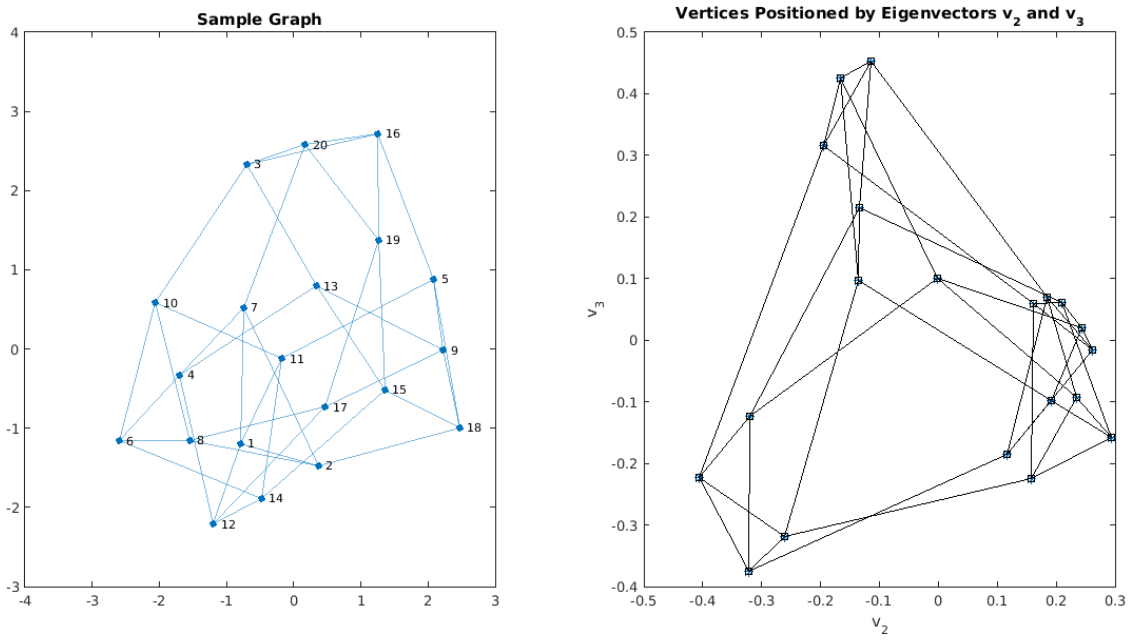

Figure 4.2: A sample regular graph (left) illustrated by the spectral coordinates of its vertices in a 2-dimensional space (right)

recalling Equation (4.8), the objective function can be re-written as

$$\begin{aligned} \boldsymbol{x}^T \boldsymbol{L}\boldsymbol{x} &= \left| \boldsymbol{\Lambda}^{1/2} \boldsymbol{a} \right|^2 \\ &= \left| \sum_{j \in G_1} s_j \right|^2 , \end{aligned} \tag{4.73}$$

where s_j is defined as

$$s_j = m_j \left[\sqrt{\lambda_1} v_{j1}, \sqrt{\lambda_2} v_{j2}, ..., \sqrt{\lambda_n} v_{jn} \right]^T , \tag{4.74}$$

and is known as the n-coordinate spectral vector of vertex j. Therefore, the problem is actually choosing a subset of G minimizing Equation (4.73) that is actually the length of the sum of its spectral vectors. As mentioned, this problem is NP-hard, as well. However, the problem can be re-written in a maximization form and can be solved by some well-known heuristic algorithms. Let μ be the desired mass of G_1,

$$\mu = \text{Mass}(G_1) = \boldsymbol{x}^T \boldsymbol{M}\boldsymbol{x} = \boldsymbol{a}^T \boldsymbol{a} \tag{4.75}$$

Then, for an arbitrary constant ξ, the equivalent maximization problem will be

$$\begin{aligned} \xi\mu - \text{Cut}(G_1, G_2) &= \xi \boldsymbol{x}^T \boldsymbol{M}\boldsymbol{x} - \boldsymbol{x}^T \boldsymbol{L}\boldsymbol{x} \\ &= \left| \sum_{j \in G_1} \hat{s}_j \right|^2 , \end{aligned} \tag{4.76}$$

where the spectral vector $\hat{s}_j$ is

$$\hat{s}_j = m_j \left[\sqrt{\xi - \lambda_1} v_{j1}, \sqrt{\xi - \lambda_2} v_{j2}, ..., \sqrt{\xi - \lambda_n} v_{jn} \right]^T, \tag{4.77}$$

and the value of ξ is recommended to be $\lambda_2 + \lambda_n$ [6].

In practice, computing all the eigenvectors is expensive or even infeasible. Hence, it is suggested to compute only the first k eigenvectors for some small k. Therefore, each vertex has a k-dimensional spectral vector.

By truncating $\boldsymbol{V}$ to a $n \times k$ matrix, a binary indicator vector may not be found as a linear combination of these k eigenvectors, so Equations (4.73) and (4.76) do not hold, consequently. A possible way is projecting the indicator vector into the subspace spanned by the computed eigenvectors. Algebraically, the vector $\boldsymbol{a} = \boldsymbol{V}^T \boldsymbol{M} \boldsymbol{x}$ specifies the projection of $\boldsymbol{x}$ onto subspace spanned by columns of $\boldsymbol{V}$. Hence, by considering the projection, $\bar{\boldsymbol{x}} = \boldsymbol{V}\boldsymbol{a}$, the objective function (4.76) can be recast as

$$\left| \sum_{j \in G_1} \hat{s}_j \right|^2 = \bar{\boldsymbol{x}}^T (\xi \boldsymbol{M} - \boldsymbol{L}) \bar{\boldsymbol{x}} \tag{4.78}$$

An algorithm approximately maximizes Equation (4.73), if $\bar{\boldsymbol{x}}$ approximates $\boldsymbol{x}$ well. In one of the fundamental greedy algorithms proposed by Alpert et al. [6], the procedure begins by placing the longest vector in G_1. Then, the vector maximizing the summation is repeatedly added until the desired mass μ is achieved. In this approach if just one eigenvector associated to the second smallest eigenvalue is used, it is actually the standard spectral bipartitioning described in Section 4.7.

The vector partitioning scheme can be implemented for other objective functions, such as minimum cut, sparsity and the isoperimetric ratio.

4.12.2 k-Subgraph Partitions

For spectral partitioning a graph into more than two subgraphs multiple indicator vectors is required. Therefore, an indicator matrix $\boldsymbol{X} \in \mathbb{R}^{n \times k}$ has been defined in which

$$X_{ji} = \begin{cases} c_{yes}, & \text{if vertex } j \in G_i \\ 0, & \text{otherwise} \end{cases} \tag{4.79}$$

The columns of $\boldsymbol{X} = [\boldsymbol{X_{*1}}, \boldsymbol{X_{*2}}, ..., \boldsymbol{X_{*n}}]$ are indicator vectors for k subgraphs. Accordingly, for a k-way partitioning, Equation (4.8) can be re-written as

$$\text{Cut}(G_i, G - G_i) = \frac{\boldsymbol{X_{*i}}^T \boldsymbol{L} \boldsymbol{X_{*i}}}{c_{yes}^2} \tag{4.80}$$

Hence, the total weight of a k-way cut is

$$\text{Cut}(G_1, G_2, ..., G_k) = \sum_{i=1}^{k} \frac{\boldsymbol{X_{*i}}^T \boldsymbol{L} \boldsymbol{X_{*i}}}{2c_{yes}^2} = \frac{\text{trace}(\boldsymbol{X}^T \boldsymbol{L} \boldsymbol{X})}{2c_{yes}^2} \tag{4.81}$$

This k-way partitioning problem can be expressed as a k-way vector partitioning problem [6]. By choosing $c_{yes} = 1$ and following Equations (4.73) and (4.76), minimizing the cut weight (4.81) is equivalent to maximizing

$$\xi \text{Mass}(G) - \sum_{i=1}^{k} \boldsymbol{X_{*i}}^T \boldsymbol{L} \boldsymbol{X_{*i}} = \sum_{i=1}^{k} \left| \sum_{j \in G_1} \hat{s}_j \right|^2 \tag{4.82}$$

This objective function is actually the sum of the length of k vectors, each associated to one subgraph. Therefore, the partitioning method essentially aims at making these vectors as long as possible. The same greedy algorithm described in Section 4.12.1 is also applicable for this case.

4.13 Summary

In summary, this chapter has reviewed all the key features of spectral graph theory. The spectral partitioning methods have been explained for both bipartitioning and k-way partitioning problems. Moreover, the overall spectrum of a graph has been scrutinized and the related problems have been discussed in this chapter.

Chapter 5

Spectral Graph Clustering

5.1 Overview

Clustering is an unsepervised technique dealing with grouping related objects of a set. It is expected that objects of each cluster are more similar to each other rather than the objects of other clusters. Graph clustering algorithms are an especial family of clustering method that aim at finding groups of related vertices in a graph. In this section spectral clustering algorithms are briefly discussed.

5.2 Graph Cut and Problems

The majority of the graph clustering methods are on the basis of the graph partitioning approaches. Spectral clustering algorithms are mostly based on the solution of the graph partitioning problems in which at least one eigenvector of the Laplacian matrix of the graph that its partitions are the solutions of some graph cut problems [63]. One of the most popular graph partitioning problems is the k-way partitioning problem in which k connected sub-graphs will be produced by eliminating edges from the initial graph (Section 4.12). The produced sub-graphs are actually a representation of well separated clusters. Therefore, the key problem is actually the elimination of some edges. To investigate the available criteria of edge elimination, at first, consider $W(C_r, C_t) = \sum_{v_i \in C_r, v_j \in C_t} w_{ij}$, $\bar{C}_i$ is every cluster like C_j so as $j \neq i$. Furthermore, Π^k is the set of possible k-way partitions of G.

5.2.1 Minimum Cut Problem

The first problem to be investigated is the k-way minimum cut problem:

$$min\ cut(\pi^k),\ \ \pi^k \in \Pi \tag{5.1}$$

where:

$$cut(\pi^k) = \frac{1}{2}\sum_{i=1}^{k} W(C_i, \bar{C}_i) \tag{5.2}$$

This problem aims at minimizing the sum of the weights of the edges whose nodes come from different clusters. One of the limitation observed in this approach is that the solutions of this problem are often partitions with isolated nodes in clusters which can be a drawback in some applications such as VLSI domain. Therefore, in the further formulations, additional aspects have been taken into account to resolve such limitations. One of these aspects is considering some lower and upper bounds to delimit the size of the clusters of a partition π^k. Such consideration adds some further constrains to the k-way min-cut formulation:

$$l_i \leq |C_i| \leq u_i,\ 1 \leq i \leq k \tag{5.3}$$

where l_i and u_i are, respectively, the lower and upper bounds for the size of cluster i and $|C_i|$ is the number if nodes in cluster C_i.

5.2.2 Minimum Ratio Cut Problem

An approach to avoid finding partitions with isolated nodes in clusters is dividing Equation (5.3) by the number of elements in each cluster [63]. This approach was introduced to solve bipartitioning problems and also known as two-way ratio cut problem [56, 92].

The method is also extended for the k-way ratio cut problems [14]:

$$\min_{\pi^k \in \Pi^k} ratiocut\left(\pi^k\right), \tag{5.4}$$

where

$$ratiocut\left(\pi^k\right) = \frac{1}{2}\sum_{i=1}^{k} \frac{W(C_i, \bar{C}_i)}{|C_i|} \tag{5.5}$$

5.2.3 Minimum Normalized Cut Problem

This approach is proposed as an alternative to the k-way minimum cut problem. In this method the summation of $W(C_i, \bar{C}_i)$ is divided by the sum of the degrees of the vertices inside the cluster C_i, $vol(C_i) = \sum_{j \in C_i} d_j$. This problem is also called k-way $ncut$ problem [76]:

$$\min_{\pi^k \in \Pi^k} mincut\left(\pi^k\right), \tag{5.6}$$

where

$$ncut\left(\pi^k\right) = \frac{1}{2}\sum_{i=1}^{k} \frac{W(C_i, \bar{C}_i)}{vol(C_i)} \tag{5.7}$$

This problem was originally derived from the relation between the normalized association and dissociation measures of a partition [76]. The association measure indicates the average connectivity of nodes in each cluster of a partition. The dissociation measure reflects the cut cost of a partition in terms of percentage of the cut edge connections regarding all the graph nodes. It has been concluded that the problem of finding a partition maximizing the dissociation inter-clusters is identical to the problem of finding a partition maximizing the association inside each cluster [76, 77].

5.2.4 Min-max Cut Problem

Another well-defined cut problem is creating min-max cut problem for clustering [22, 21]. The main purpose of the min-max is indeed minimizing the inter-cluster similarities,

$$\min_{\pi^k \in \Pi^k} MinMaxCut\left(\pi^k\right), \tag{5.8}$$

where

$$MinMaxCut\left(\pi^k\right) = \frac{1}{2}\sum_{i=1}^{k} \frac{W(C_i, \bar{C}_i)}{W(C_i, C_i)} \tag{5.9}$$

5.2.5 Modularity Maximization Problem

Modularity is a measure of quality validation in graph clustering [64]. The modularity denoted by q, in a partition π, measures the difference between the sum of the edge weights connecting vertices within the cluster and the expected sum of edge weights connecting vertices inside each cluster of the same partition in a random graph:

$$q(\pi) = \frac{1}{2m}\sum_{i=1}^{n}\sum_{j=1}^{n}\left(w_{ij} - \frac{d_i d_j}{2m}\right) y_{ij}, \tag{5.10}$$

where

$$m = \frac{\sum_i^n d_i}{2}, \tag{5.11}$$

and

$$y_{ij} = \begin{cases} 1, & \text{if } v_i \text{ and } v_j \text{ belongs to the same cluster} \\ 0, & \text{otherwise} \end{cases} \tag{5.12}$$

The Equation is multiplied by $1/2$ because the sum of intra-cluster edge weights is counted twice. Moreover, the multiplication by $1/m$ is in view of normalizing to ensure its value is less than or equal to 1. The portion $\frac{d_i d_j}{2m}$ is also denoted by p_{ij} demonstrating the probability of having the edge (v_i, v_j) in a random graph with the same node degree sequence as the original graph G. The closer value of $q(\pi)$

to 1 means the better connectivity of the partition. To conclude, the modularity maximization problem can be presented as the following formula:

$$\max_{\pi \in \Pi} q(\pi), \tag{5.13}$$

where Π is the set of all possible k-way partitions of G, with $k = 1, ..., n$.

5.3 Spectral Clustering Algorithms

According to the previous section, spectral clustering algorithms are actually based on and closely related to the spectral partitioning methods. Accordingly, these algorithms are categorized into two main group: recursive two-way spectral clustering algorithms and direct k-way spectral clustering algorithms. The former is almost identical to the approach explained in Section 4.7.2 and the later is actually the algorithm described in Section 4.12.

5.3.1 Two-way Partitioning Algorithms

As it explained in the previous chapter, the two-way partitioning approach is actually an answer to the min cut problem. The main scheme of theses algorithms is demonstrated in the following algorithm

Algorithm 4: The main scheme of two-way spectral clustering algorithm [40]

Data: $G(V, E)$, Laplacian matrix L, Threshold value r
Result: $graph\ clusters$

```
Procedure Two-way Clustering
    employing numerical algorithms to calculate the second eigenvalue λ2 of
     L and its associated eigenvector v2;
    for all i ∈ V do
        if v2(i) > r then
            put i in the first partition (i ∈ C1);
        else
            put i in the second partition (i ∈ C2);
        end
    end
    return the resulting partitions;
end
```

Various algorithms are distinguished by the threshold value. For instance, in general spectral bipartitioning, it is suggested that $r = 0$. But, it is also suggested to be the median of $\boldsymbol{v_2}$ or the largest gap between two consecutive elements in the sorted $\boldsymbol{v_2}$

5.3.2 k-way Partitioning Algorithms

The basis of these clustering algorithms is vector partitioning method described in Section 4.12.1. To clarify this clustering approach, consider a fundamental case of the k-way min cut problem, which is essentially an especial case of the Equation (4.82), if the mass of all vertices is 1. In this case, the k-way min cut problem can be reformulated as a max-sum vector partitioning problem

$$\max_{\pi^k} \xi n - cut(\pi^k), \tag{5.14}$$

where $\xi \geq \lambda_n$ and $l_i \leq |C_i| \leq u_i$ for $1 \leq i \leq k$

Afterwards, due to Equation (4.77), a matrix of scaled eigenvectors is created such that

$$\boldsymbol{V_d} = [\hat{v}_{ij}]_{n \times d} \tag{5.15}$$

where

$$\hat{v}_{ij} = v_{ij}\sqrt{\xi - \lambda_j} \tag{5.16}$$

Moreover, $\boldsymbol{U} = [u_{ij}]_{n \times d}$ is defined as the matrix including the first d eigenvectors of the Laplacian matrix $\boldsymbol{L}$. Defining a set $P = \left\{\boldsymbol{y_1^n}, ..., \boldsymbol{y_n^n}\right\}$, where $\boldsymbol{y_i^n}$ is the i-th row of $\boldsymbol{V_n}$, the procedure proposed by Alpert et al. [6] demonstrated in Algorithm 5. This algorithm is called multiple eigenvector linear orderings (MELO).

Algorithm 5: The MELO algorithm [6, 63]

Data: $G(V, E)$, the number of desired clusters k, the number of eigenvectors to be use d
Result: $graph\ clusters$

1 **Procedure** *MELO*
2 construct the matrix of scaled eigenvectors $\boldsymbol{V_d}$ by employing the Equation 5.16, (set $P = \emptyset$);
3 create $\boldsymbol{Y} = \left[\boldsymbol{y_i^d}\right]_{n \times d}$, for $1 \leq i \leq n$, where $\boldsymbol{y_i^d}$ is the i-th row of $\boldsymbol{V_d}$;
4 **for** *j = 1 to n* **do**
5 find $\boldsymbol{y_i^d},\ 1 \leq i \leq n$, such that it maximizes $\| \sum_{\boldsymbol{y} \in P} \boldsymbol{y} + \boldsymbol{y_i^d} \|$;
6 add $\boldsymbol{y_i^d}$ to P and remove it from $\boldsymbol{Y}$;
7 label i as the j-th vertex in the ordering;
8 **end**
9 find the final k-way partition by linear ordering and applying the approach proposed by Alpert ans Kahng [5];
10 **return** the resulting partitions;
11 **end**

In the step 9, the k-way final partition will be generated by employing a dynamic programming procedure [5]. It is also worthwhile to note that the complexity of this algorithm is $O(dn^2)$.

Chan et al. [14] proposed a spectral relaxation of the k-way ratio cut problem (Equation (5.4)). The proposed spectral heuristic method is based on the orthonormality among the eigenvectors of the Laplacian matrix. In this approach, an indicator matrix is defined such that $\boldsymbol{X} = [x_{ij}]_{n \times k}$, where

$$x_{ij} = \begin{cases} \frac{1}{\sqrt{|C_j|}}, & \text{if } i \in C_j \\ 0, & \text{otherwise} \end{cases} \tag{5.17}$$

Furthermore, a rotation partitioned matrix $\boldsymbol{R} = [r_{ij}]_{n \times n}$ is defined such that

$$r_{ij} = \begin{cases} \frac{1}{\sqrt{|C_l|}}, & \text{if } i, j \in C_l \\ 0, & \text{otherwise} \end{cases} \tag{5.18}$$

Accordingly, it can be demonstrated that $\boldsymbol{x_j}^T \boldsymbol{L} \boldsymbol{x_j} = ratiocut(C_j, \bar{C_j}) = \left(\boldsymbol{X}^t \boldsymbol{L} \boldsymbol{X}\right)_{jj}$. It can also be deduced that

$$ratiocut\left(C_1, C_2, ..., C_k\right) = \sum_{j=1}^{k} \boldsymbol{x_j}^T \boldsymbol{L} \boldsymbol{x_j} = trace\left(\left(\boldsymbol{X}^t \boldsymbol{L} \boldsymbol{X}\right)_{jj}\right) \tag{5.19}$$

Consequently, It can be obtained that

$$\min_{\pi^k \in \Pi^k} \frac{1}{2} \sum_{i=1}^{k} \frac{W(C_i, \bar{C_i})}{|C_i|} = trace\left(\boldsymbol{U}^T \boldsymbol{L} \boldsymbol{U}\right) = \sum_{i=1}^{k} \lambda_i \tag{5.20}$$

In other words, $\sum_{i=1}^{k} \lambda_i$ is a lower bound for the k-way ratio cut problem.

As the solution of a relaxed form of the k-way ratio cut problem, $\boldsymbol{U}$ is an approximation for $\boldsymbol{X}$ and $\boldsymbol{Z} = \boldsymbol{U}\boldsymbol{U}^T$ is an approximation for the matrix $\boldsymbol{R}$. Considering $\boldsymbol{U}^T = [\boldsymbol{u}'_1, ..., \boldsymbol{u}'_n]$, it can be obtained $z_{ij} = \boldsymbol{u'}_i^T \boldsymbol{u}'_j$.

The main idea of the heuristic proposed for this clustering problem is to measure how close node are to each other by evaluating the cosine between their associated rows in the matrix $\boldsymbol{U}$. Hence, the elements of the directional cosine matrix of the rows of this matrix, denoted by $\hat{\boldsymbol{P}}$ is defined as

$$\hat{p}_{ij} = \frac{\boldsymbol{u}'_i \boldsymbol{u}'_j}{\| \boldsymbol{u}'_i \| \| \boldsymbol{u}'_j \|} \tag{5.21}$$

The procedure is depicted by Algorithm 6.

Algorithm 6: The KP algorithm [14]

Data: $G(V, E)$, the number of desired clusters k
Result: $graph\ clusters$

Procedure *KP*
  calculate the first k eigenvectors of L and sort them in the columns of the matrix $\boldsymbol{U}$;
  select k nodes to represent each of the k prototype clusters by using their magnitude $\| u_i' \|$ and the relation of orthogonality between them;
  `/* calibrating the k prototypes */`
  for *iter = 1 to 4* **do**
    calculate the average of the prototypes from the previous iteration;
    choose the seed for each prototype by the posterior selection of the closest node;
  end
  for *i = 1 to n* **do**
    calculate the cosine between $\boldsymbol{u_i'}$ and each of the prototypes;
    if *one of the calculated cosines* $> \cos(\pi/8)$ **then**
      assign node i to the prototype with the largest cosine;
    end
  end
  define S as the set if outsiders (non-allocated nodes);
  for *all* $s \in S$ **do**
    find the largest weight cut between s and all existing clusters;
    mark the cluster with the largest cut value (key) as the target;
  end
  while $S \neq \emptyset$ **do**
    insert the outsider with the largest key to its target;
    recalculate all targets and keys of the remaining neighbor outsiders;
  end
  return the resulting partitions;
end

Sometimes, it might be the case that there exist more edges between outsiders than the clusters. An improvement to this algorithm also proposed by Chan et al [14] suggests that after allocation of an outsider, the solution of merging all the remaining outsiders to the cluster with the best ratio cut is calculated and if it is better than merging only one element, then all of them will be assigned to the cluster with the best ratio cut [63].

Another approach grouping the rows of the matrix $\boldsymbol{U}$ is the Unnormalized k-means algorithm which is also known as Ukmeans. This procedure is summarized in Algorithm 7.

Algorithm 7: The Ukmeans algorithm [90, 63]

Data: $G(V, E)$, the number of desired clusters k
Result: graph partitions

Procedure *Ukmeans*
  calculate the first k eigenvectors of L and sort them in the columns of the matrix U;
  `/* The` i`-th row of` U `is corresponded to the node` v_i `*/`
  apply the k-means algorithm to U and obtain a k-way partition $\pi'^k = \{C'_1, ..., C'_n\}$;
  `/* Form the final partitions */`
  for *i = 1 to n* **do**
    if *the* i*-th row of the matrix* U *belongs to* C'_l *in the* π'^k **then**
      assign v_i to the cluster C_l;
    end
  end
  return the resulting partitions;
end

5.4 Summary

Briefly, this chapter has discussed the prominent graph clustering algorithms and highlights the connections between this class of clustering methods and spectral partitioning approaches by providing a summary of graph cuts and the algorithm proposed to solve these problems.

Chapter 6

Practical Application: Group Testing

6.1 Overview

In this chapter, a novel application of graph partitioning in graph-constrained group testing has been investigated. Typically, a group testing problem is dealing with obtaining a method to split up the task of identifying elements of a set which have some specific properties into tests on subsets, which are also known as groups, rather than on each element, individually. By taking this approach, the number of required tests is significantly reduced. The main idea of the proposed approach in this chapter is that by employing spectral graph partitioning and by holding some specific condition in this method to partition a given graph into expander subgraphs can take advantage of group testing schemes in a data set represented by a graph.

6.2 Graph-Constrained Group Testing

To understand the application, at first, a summary of the problem has been presented. In general, in a non-adaptive group testing, n items of a given set are arbitrary grouped into different pools. Each pool is then tested to identify defective items. The purpose of this problem is minimizing the number of pools, and accordingly tests, needed to identify at most d defective items. Motivated by applications in network tomography, sensor networks and infection propagation, the graph-constrained group testing has been introduced [16] in which the given set is necessarily represented by a graph. In this applications, unlike the conventional group testing, the graph imposed some constraints on the group formation.In this case, a test is usually associated with a random walk.

6.2.1 Network Tomography

One of the most important applications illustrating the graph-constrained group testing problem is network tomography in which identification of congested links in a

given network is a key problem [25]. Practically, in an end-to-end path, most of the packet losses only happen in a few links along the path. The graph-constraint group testing approach can be employed to find the location of such congested links [16].

In this context, the network can be modelled as a graph $G = (V, E)$ where each vertex associated to a router/host on the network and the edges demonstrate the communication links. For instance, Figure (6.1) illustrates a network in which the route $1 \to 4 \to 6 \to 7 \to 8 \to 9$ is valid. In practice, to test a network, there exists

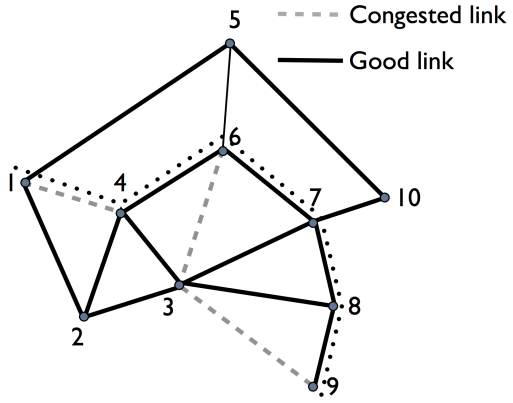

Figure 6.1: Graph representation of a network [16]

a monitoring system consisting of some nodes known as vantage points which can send and receive packets. The packets are sent through the network by assigning the routes and the end hosts. To identify the congested links, all measurement results on whether a packet has reached its destination will be reported to and analysed in a central server. In this case, the graph-constrained group testing is actually aiming at minimizing the number of required measurements. In this application, the graph constrains the group forming in such a way that a vantage point can only assign the routes which are in a path in the graph G [16].

6.2.2 Problem Statement

Let $G = (V, E)$ be a graph with at most d defective vertices . The goal is locating the set of defective items by conducting as least as possible measurements. Each measurement determines whether the set of item examined along a path on the graph includes a defective item or not. The problem of finding defective vertices is known as vertex group testing and the problem of finding defective edges is called edge group testing.

It should be noted that not all sets of vertices can be grouped together, but only those that share a path on the graph G can be regarded as a pool.

By considering $T(n)$ as the mixing time of the graph G, it is demonstrated that [16]:

- The conventional group testing problem is actually a special case of graph-constrained group testing problem in which the set of items can be represented by a complete graph K_n.
- In a general graph, by conducting $m = O(d^2T^2(n)\log(n/d))$ non-adaptive tests, the defective items can be identified.
- In the case of the Erdős-Rényi random graph denoted by $G(n,p)$ and also the expander graphs, the number of non-adaptive tests required to locate the defective items is

$$m = O(d^2 \log^3(n)) \tag{6.1}$$

Therefore, according to Equation (6.1), in a set with a huge number of items, the defective items can be identified by much less tests, if the representative graph is an expander.

Accordingly, it can be inferred that if a given graph is not expander but it can be partitioned into expander subgraphs, then Equation (6.1) will still be valid in each subgraph and the number of required tests will significantly be less than the case of a general graph.

6.3 Expander Graphs

Expander graph is a very useful object with many applications in the computer science.

There are different possible ways to define expander graphs:

- Spectrally: Expander graphs are graphs with a large spectral gap, i.e. $\lambda_1 - \lambda_2$ is large.
- Combinatorially: Expander graphs are graphs with very good connectivity, i.e. $|\delta(S)|/|S|$ is large for all $S \subseteq V$ (edge expansion), or $|N(S)|/|S|$ is large for all $S \subseteq V$ (vertex expansion).
- Probabilistically: From this point of view, expander graphs are graphs for which random walk mixes very rapidly.

As described before, Cheeger's inequality shows that a graph has a large spectral gap if and only if the edge expansion is large. It is also explained that the random walk mixes rapidly if and only if there is a large spectral gap.

According to these definitions, complete graphs are the best expander graphs. However, in the most of applications, the sparse expander graphs are interested in which there are only linear number of edge, e.g. d-regular graphs for constant d.

6.4 Partitioning into Expanders

As mentioned earlier, in Section 4.11, given an undirected graph $G = (V, E)$, the k-th smallest eigenvalue of the (normalized) Laplacian matrix of G is zero if and only if the graph has at least k connected components. In other words, $\lambda_k > 0$ if and only if G has at most $k - 1$ connected components.

Accordingly, it can easily be inferred that if $\lambda_k = 0$ and $\lambda_{k+1} > 0$ then G has exactly k connected components. Although, these equivalent statements give some useful information about the existence of k connected components, no information about the quality of subsets can be achieved. Such information can essentially be useful in graph-constrained group testing. The aim of this section is discussing the robust versions of the abovementioned fact and trying to implement this information in application of group testing scheme.

To the best of knowledge, most of the works on the sparsest cut problem are only dealing with the partitioning graph into sets of small outside conductance of the subsets in the partition [9, 58, 56, 55]. there exist only two conducted researches studying the inside conductance of the subsets.

6.4.1 Definitions

To study the quality of the subsets and consequently the quality of a k-way partitioning, several combinatorial measures have been proposed, such as k-center, k-median, and conductance. The latter defined in Section 4.2 is one of the best objective functions for measuring the quality of a cluster. But after finding a k-way partitioning, although a set S might have a small conductance, it can be loosely-connected or even disconnected inside. Kennan et al. [46] proposed a bicriteria measure which has also been used by Oveis-Gharan and Trevisan [36] to evaluate the quality of a k-clustering based on the both inside and outside conductance of sets.

Basically, for $P \subseteq V$, the inside conductance of P denoted by $\phi(G[P])$ is defined as the conductance of the induced subgraph of G on the vertices of P. Suggested by Kennan et al., a k-partitioning into $P_1, ..., P_k$ is good if for all $1 \leq i \leq k$, $\delta(P_i)$ is small but $\phi(G[P_i])$ is large [46]. Accordingly, Oveis-Gharan and Trevisan proposed (ϕ_{in}, ϕ_{out})-clustering as a new measure of clustering quality [36]. By their definition, k disjoint subsets $A_1, ..., A_k$ of V are a (ϕ_{in}, ϕ_{out})-clustering, if for all $1 \leq i \leq k$,

$$\begin{aligned} \phi(G[A_i]) &\geq \phi_{in} \text{ and} \\ \phi_G(A_i) &\leq \phi_{out} \end{aligned} \tag{6.2}$$

In this way, the graphs that contain a k-partitioning such that $\phi_{in} \gg \phi_{out}$ will be of interest.

6.4.2 Theorems

The first theoretical result which guarantees a (ϕ_{in}, ϕ_{out}) partitioning is based on the theorem proposed by Tanaka [85, 86]. Consider $\rho(k)$ as the maximum conductance of any k disjoint subsets of G, which for any $k \geq 2$ is defined as

$$\rho(k) = \min_{\text{disjoint} A_1,..,A_k} \max_{1 \leq i \leq k} \phi(A_i) \tag{6.3}$$

Therefore, $\rho(2) = \phi(G)$. Accordingly, a higher order variant of Cheeger's inequality has been proposed by Lee et al. [55] such that for any graph G and $k \geq 2$,

$$\frac{\lambda_k}{2} \leq \rho(k) \leq O(k^2)\sqrt{\lambda_k} \tag{6.4}$$

By this definitions, according to Tanaka's theorem, if $\rho_G(k+1) > 3^{k+1}\rho_G(k)$ for some k, then G has a k-partitioning that is a $(\rho(k+1)/3^{k+1}, 3^k\rho(k))$-clustering. In other words, if there is a large gap between $\rho(k)$ and $\rho(k+1)$ then there exists a k-partitioning that is a $(\exp(k)\rho(k), \rho(k+1)/\exp(k))$-clustering. The large exponential gap required between $\rho(k)$ and $\rho(k+1)$ is actually a drawback of this theorem.

Another existential theorem proposed by Oveis-Gharan and Trevisan [36] is the following. For any $k \geq 2$ if $\lambda_k > 0$, then for some $1 \leq l \leq k-1$ there exists a l-partitioning of V into subsets $P_1, ..., P_l$ that is a $(\Omega(\rho(k)/k^2), O(l\rho(l))) = (\Omega(\lambda_k/k^2), O(l^3)\sqrt{\lambda_l})$-clustering. This theorem shows that between the order k conductance and the order $k+1$ conductance can really be arbitrarily small. But this result is not algorithmic. With some loss of parameters, an algorithmic theorem can be obtained [36]:
Algorithmic Theorem: There is a simple local search algorithm which for any $k \geq 1$ if $\lambda_k > 0$ finds a l-partitioning of V into sets $P_1, ..., P_l$ that is a $(\Omega(\lambda_k^2/k^4), O(k^6)\sqrt{\lambda_{k-1}})$ where $1 \leq l \leq k$. For an unweighted graph G the algorithm runs in polynomial time in the size of G. This theorem leads to a functional corollary that there exists a universal constant $c > 0$ such that for any graph G if $\lambda_{k+1} \geq c.k^2\sqrt{\lambda_k}$, then a k-partitioning of G can be obtained that is a $(\Omega(\lambda_{k+1}/k), O(k^3\sqrt{\lambda_k}))$-clustering.

Therefore, having a polynomial $O(k^2)$ gap between λ_k and λ_{k+1} guarantees the existence of a partitioning of V into k sets of small conductance, each inducing an expander subgraph of large conductance.

6.5 Extension to the Group Testing Scheme

According to the corollary resulted by the algorithmic theorem mentioned in the previous section, a graph-constrained group testing problem whose graph is not an expander but there exists a polynomial $O(k^2)$ gap between its k-th and $(k+1)$-th eigenvalues can take advantage of the efficient group testing algorithms on expander graphs.

In this case, the problem of graph-constrained group testing is reduced to finding a partitioning of G into k expanders. Based on the proposed algorithmic theorem, Oveis-Gharan and Trevisan [36] presents a polynomial time algorithm for partitioning graph G into k expander subgraphs (Algorithm 8).

Algorithm 8: A polynomial time algorithm for partitioning G into k expanders [36]

Data: $k > 1$ such that $\lambda_k > 0$
Result: A $(\phi_{in}^2/4, \phi_{out})$ l-partitioning of G for some $1 \leq l \leq k$

Procedure *k-expander*
  initialize $l = 1$, $P_1 = B_1 = V$;
  while $\exists\ 1 \leq i \leq l$ *such that* $w(P_i - B_i \to B_i) < w(P_i - B_i \to P_j) <$ *for* $j \neq i$, *or Spectral Partitioning finds* $S \subseteq P_i$ *such that* $\phi_{G[P_i]}(S), \phi_{G[P_i]}(P_i - S) < \phi_{in}$ **do**
    let $S_B = S \cap B_i$;
    let $\bar{S}_B = \bar{S} \cap B_i$;
    let $S_P = S \cap (P_i - B_i)$;
    let $\bar{S}_P = \bar{S} \cap (P_i - B_i)$;
    if $\max\{\phi(S_B), \phi(\bar{S}_B)\} \leq (1 + 1/k)^{l+1}\rho^*$ **then**
      let $B_i = S_B,\ P_{l+1} = B_{l+1} = \bar{S}_B$ and $P_i = P_i - \bar{S}_B$;
      $l++$;
      goto step 3;
    end
    if $\max\{\phi(S_B, B_i), \phi(\bar{S}_B, B_i)\} \leq 1/3k$ **then**
      update B_i to either of S_B or $\bar{S}_B$ with the smallest conductance;
      goto step 3;
    end
    if $\phi(S_P) \leq (1 + 1/k)^{l+1}\rho^*$ **then**
      let $P_{l+1} = B_{l+1} = S_P$ and $P_i = P_i - S_P$;
      $l++$;
      goto step 3;
    end
    if $w(P_i - B_i \to P_i) < w(P_i - B_i \to B_j) <$ *for* $j \neq i$ **then**
      update $P_j = P_j \cup (P_i - B_i)$, and let $P_i = B_i$;
      goto step 3;
    end
    if $w(S_P \to P_i) < w(S_P \to P_j) <$ *for* $j \neq i$ **then**
      update $P_i = P_i - S_P$, and merge S_P with $\text{argmax}_{P_j} w(S_P \to P_j)$;
    end
  end
  return the resulting partitions $P_1, ..., P_l$;
end

In this proposed algorithm, ρ^* is defined as

$$\rho^* = \min\left\{\frac{\lambda_k}{10}, 30k^5\sqrt{\lambda_k - 1}\right\} \tag{6.5}$$

Moreover, it has been defined that $\phi_{in} = \lambda_k/140k^2$ and $\phi_{out} = 90k^6\sqrt{\lambda_{k-1}}$. Besides, it should be noted that for $S \subseteq P \subseteq V$ $\phi_{G[P]}(S)$ denotes the conductance of S in the induced subgraph $G[P]$. For $S, T \subseteq V$,

$$w(S \to T) = \sum_{u \in S, v \in T-S} w(u, v). \tag{6.6}$$

In this definition, S and T are not necessarily disjoint. Finally, for $S \subseteq B_i \subseteq V$ it has been defined

$$\phi(S, B_i) = \frac{w(S \to B_i)}{\frac{vol(B_i - S)}{vol(B_i)}.w(S \to V - B_i)} \tag{6.7}$$

After employing the algorithm on a graph with the mentioned polynomial gap, k expander subgraph will be obtained and an efficient group testing algorithm can find the defective items. In this case, the number of non-adaptive tests required to identify the defective items in Equation (6.1) will be increased by a factor of k.

6.6 Summary

In this chapter, a novel strategy has been suggested to use the spectral graph partitioning concepts to extend the applicability of graph-constrained group testing methods.

The strategy is simply dealing with partitioning the graph representing a set of items into expander subgraphs and then employing an efficient group testing algorithm on each subgraphs, independently. This approach can be computationally cheaper than applying the algorithm on the main general graph, especially when the number of items is much more than the number of defective items.

Chapter 7

Programming and Visualization

7.1 Overview

In addition to conducting a systematic research in the graph partitioning topics, employing programming tools to implement the discussed algorithms can be a good way to improve the understanding in this area. It can also be useful to evaluate novel algorithms. Therefore, in this chapter, the implementation of graph partitioning algorithms has been discussed. Accordingly, at first, some of the available open source packages are briefly introduced. Then, even though it was not a requirement in this project, the design and development of a functional MATLAB toolbox implementing spectral partitioning algorithms is presented in this chapter.

7.2 Available Packages

There are a lot of computational packages implementing graph partitioning algorithms. Table 7.1 has described some of the well-known graph partitioning packages or the packages using graph partitioning methods.

Table 7.1: List of packages implementing graph partitioning methods

No.	Pakage name	Programming language	Description and Features
1	CDTB	MATLAB	The Community Detection Toolbox contains several functions, such as graph generator, clustering algorithms and evaluation functions
2	grPartition	MATLAB	A MATLAB function to partition large graphs efficiently by implementing a graph partitioning algorithm based on spectral factorization [42]
3	Chaco	C	A graph partitioning software package allowing the recursive application of several methods to find small edge separators in weighted graphs. The most noticeable implemented algorithm families are as follows: Inertial, spectral, Kernighan-Lin, and multilevel methods [41] There is also a mex file interface in MATLAB to communicate with Chaco and using its routines.
4	KMETIS	C	A graph partitioning package using the METIS library [3]
5	Octave Network Toolbox	MATLAB	A toolbox generally used for network and graph analysis. The original package has been developed by Strategic Engineering Research Group (SERG), MIT [91].

According to the above table, MATLAB (matrix laboratory) has been employed in most of these packages. As a matrix-based programming language, with the strong built-in graphics and visualization tools, and the ability of the integration with the other programming language, MATLAB has been introduced and optimized as one of the most functional tools for simulation and solving engineering and scientific problems [2]. Therefore, in this project it has been employed for studying spectral graph partitioning methods.

7.3 MATLAB Implementation

There exist lots of functions and tools in MATLAB which can be employed in graph theory applications. Generally, graphs are used in MATLAB to model pairwise relations between objects using vertices and edges.

To start working with graphs and implementing the algorithms in MATLAB, graph objects are employed as the fundamental component. Basically, graph objects represent undirected graphs in which the nodes are connected by direction-less edges. There are also some built-in methods which can be used to modify the object or work with it.

The algorithms are implemented as MATLAB functions which can be called in Command Window or in other MATLAB functions. These functions are actually the building blocks of the final toolbox. To facilitate employing the toolbox, a Graphical User Interface (GUI) is created. In this project, MATLAB GUI development environment which is also known as GUIDE has been employed to create this front-end. Figure 7.1 demonstrates the main interface.

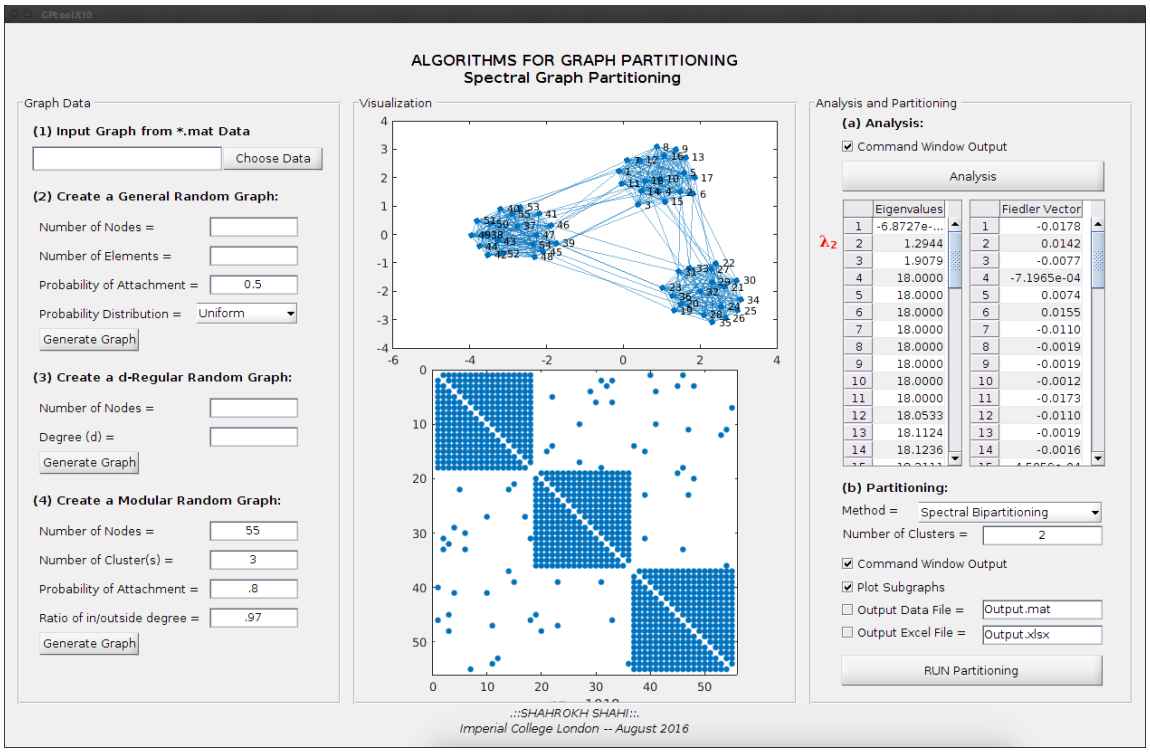

Figure 7.1: The Graphical User Interface of the toolbox

The graphical user interface consists of three main parts:

Graph Data: This panel is used to import a graph object by loading a MATLAB data file (*.mat) or generating a random graph. It is also possible to generate a random graph to work with. To generate a random graph three different options are available which will be explained.

Visualization: This panel provides the user with a graphical presentation of the inputted graph. Moreover, the adjacency matrix of the graph will be represented by its sparsity pattern in which the non-zero elements of this matrix will be displayed by small blue points. This can also give user a better sense of the graph connectivity.

Analysis and Partitioning: In this panel, at first, the user can execute the Analysis module to calculate some important parameters, such as the Laplacian matrix and

its eigenvalues and eigenvectors. By calculating the spectrum, some of the important and functional theorems explained in the previous chapters can be investigated for an inputted graph. By pressing the RUN button the partitioning method chosen by the user will be applied on the graph and the resulted subgraphs will be displayed. The user can also store the results as a MATLAB data file or a spreadsheet file. This can be helpful for further investigations or researches.

7.3.1 Generating Random Graphs

As mentioned earlier, there exist three different options to generate random graphs in the toolbox. The first one is generating a general random graph in which after generating the nodes, each pair of them will randomly be attached. The probability of attachment and the probability distribution is can be selected by the user.

The general random graph model is based on the Erdős-Rényi model in which a complete graph on n vertices is produced and the edges independently will then be removed with probability $1 - p$. The resulting graph is known as Erdős-Rényi random graph and denoted by $G(n, p)$. Figure 7.2 illustrated a general random graph generated in the GUI.

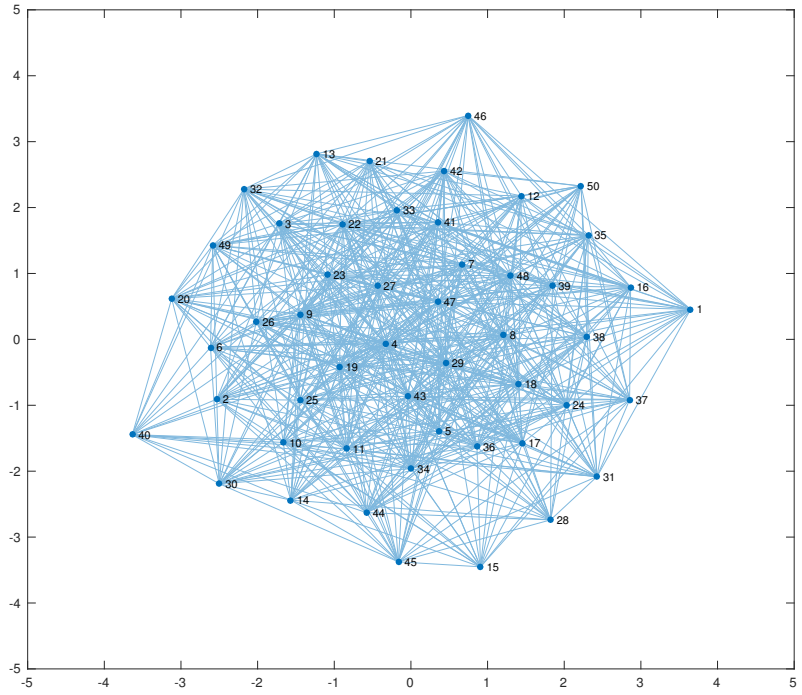

Figure 7.2: A general random graph created with $n = 50$ and $p = 0.5$

The second option is generating a regular random graph. In this part, a regular random graph will be produced by employing the algorithm proposed by Steger and

Wormald [82, 1] which produces an asymptotically uniform random regular graph in a polynomial time. It is also demonstrated that the algorithm works in practical quadratic time for relatively large d [50].

In many researches and in this project as well, regular graphs are of interest. Thanks to their special algebraic properties, this type of graphs can be important for evaluating some basic conjectures in general theorems. For instance, in this project, regular graphs have been considered as one of the fundamental forms of expander graphs. An example of a random 4-regular graph generated in the GUI is demonstrated in Figure 7.3.

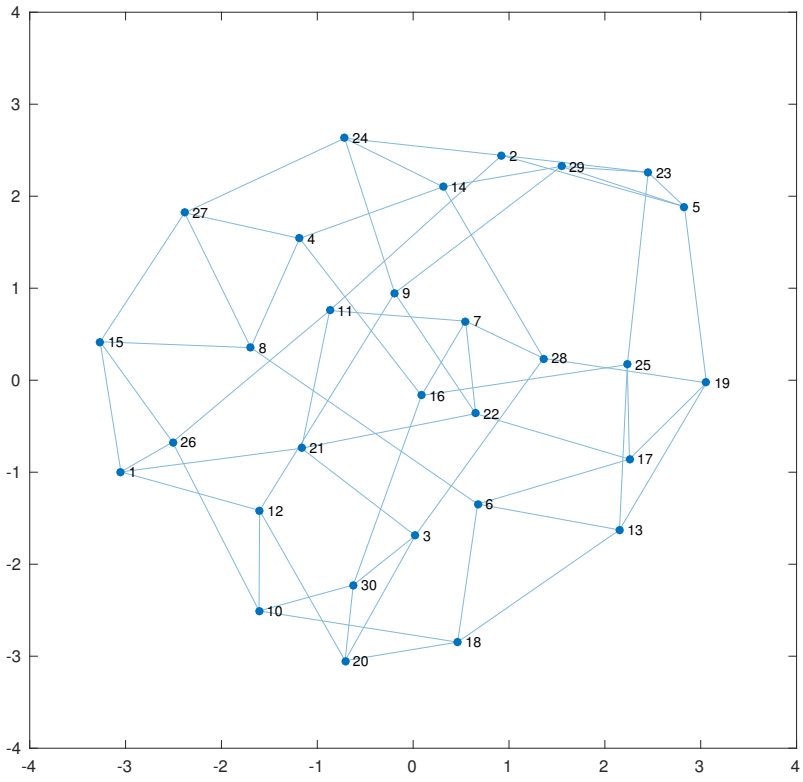

Figure 7.3: A random regular graph created with $n = 30$ and $d = 4$

The last option is generating a modular random graph by which the user can create a graph with a particular number of communities. In some applications, it needs to have a random graph in which some communities are intentionally created. This type of random graph can be useful in studying community detection algorithms or graph partitioning methods. Hence, a random modular graph generator has been implemented in this part.

In this research, this type of random graphs has been used to study the spectral graph partitioning algorithms and also evaluate the toolbox. A simple method to create such graphs follows these steps:

- Creating a list of vertices by using randperm function. The resulted array contains a random permutation of integers in the range of 1 to $|V|$.
- To form the communities, the array is broken into c sub-arrays, where c is number of required communities.
- Then the edges are randomly added due to a predefined probability values for existence of an edge between nodes within a community denoted p_{in} and the nodes in different communities denoted by p_{out}. In the GUI, the former has been called the probability of attachment and the latter will be calculated by using a ratio describing the relation between the average degree within a community to the average degree outside degree.

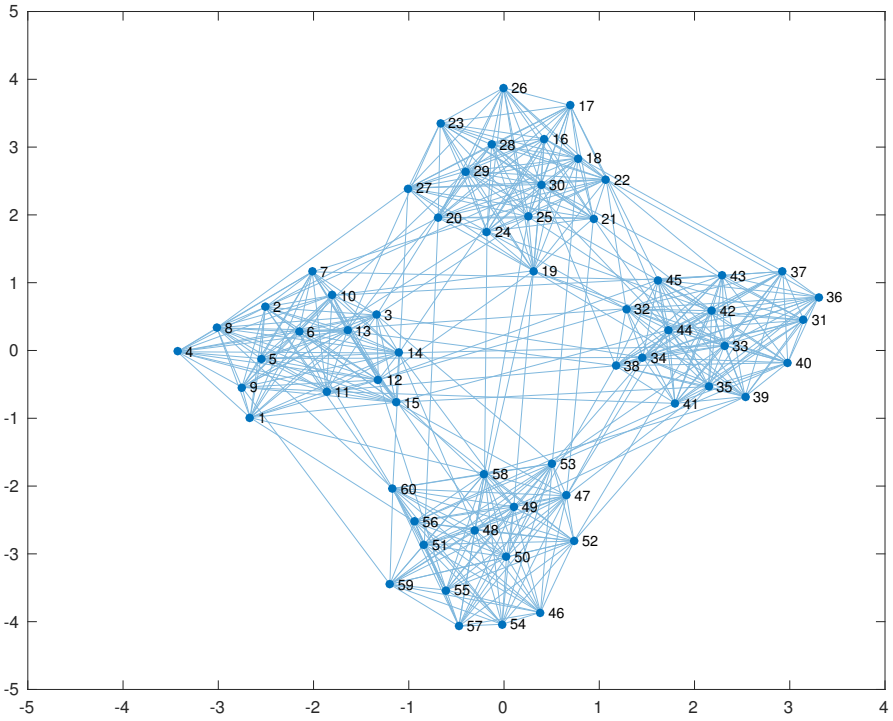

Figure 7.4: A random modular graph created with 60 nodes, 4 communities and the probability $p = 0.8$

7.3.2 MATLAB Toolbox

After creating a set of functions and tools in MATLAB, all files can be packaged to create a toolbox to share with others. These files might include MATLAB code, data, apps, examples, and documentation. By creating a toolbox, a single installation file is generated (.mltbx) that enables the user to install the toolbox within the MATLAB.

When the end user install the toolbox, the .mltbx file manages the details such as setting the MATLAB path and other installation details.

One of the fundamental ways for sharing a MATLAB toolbox is uploading the installation file to MATLAB Central File Exchange. In this way, the users will be able to download the toolbox within the MATLAB. It should be noticed that there are some limitation on file type for submission to MATLAB File Exchange. In this case, the package cannot include MEX-files or other binary executable files, such as DLL or ActiveX controls [2].

7.4 Evaluation

In this section an example of spectral partitioning is solved by employing the developed toolbox.

7.4.1 Example1: Bisection Problem

At first, a sample random graph with two communities is created in the GUI. Figure (7.5) illustrates the generated random graph.

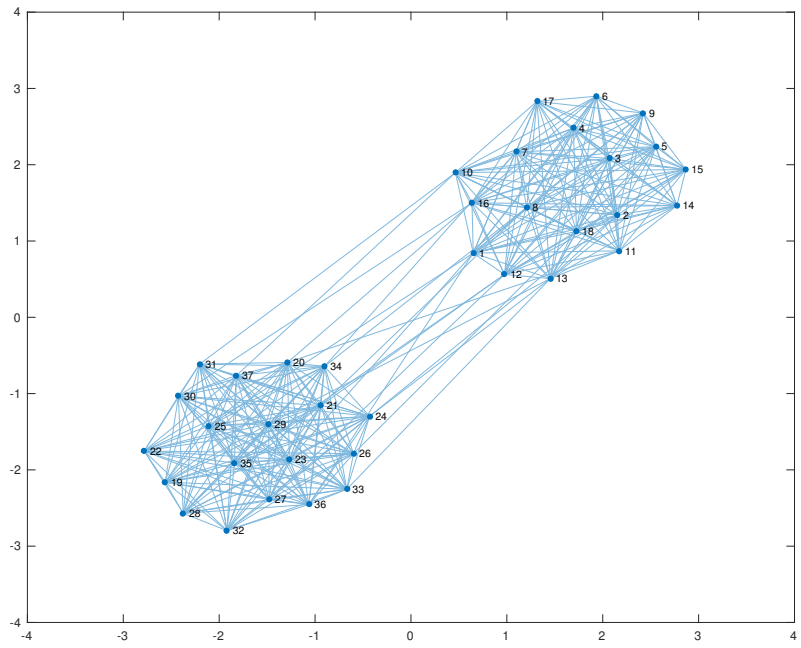

Figure 7.5: The initial random graph with 37 nodes and two communities

Moreover, the sparsity pattern of its adjacency matrix showed in the following figure.

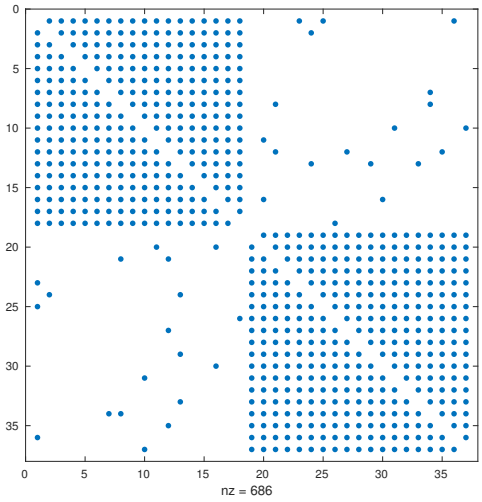

Figure 7.6: The visualization of the sparsity pattern of the adjacency matrix

By running the Analysis module, the graph spectrum and Fiedler vector is calculated and demonstrated in the GUI (Figure 7.7)

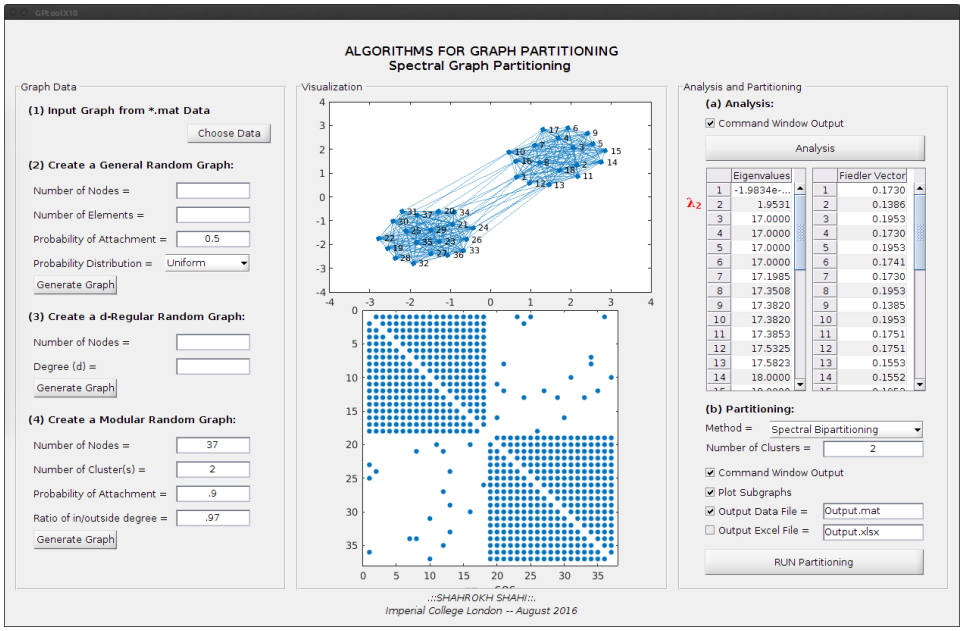

Figure 7.7: The GUI after running the Analysis module

It is also possible to plot the values of Fiedler vector to observe the gap between the values of different clusters (Figure 7.8).

Finally, the resulted subgraphs can be obtained after running Partitioning module (Figure 7.9).

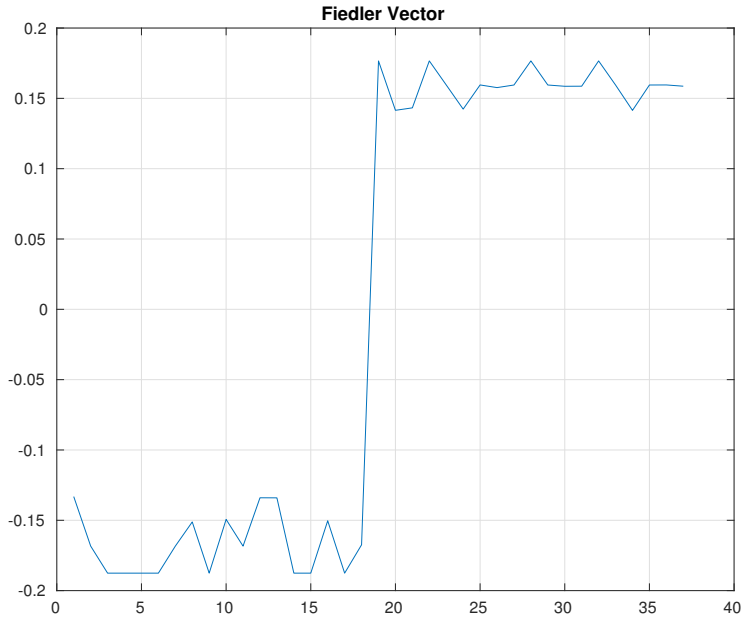

Figure 7.8: The gap in the Fiedler vector reveals the existence of two clusters

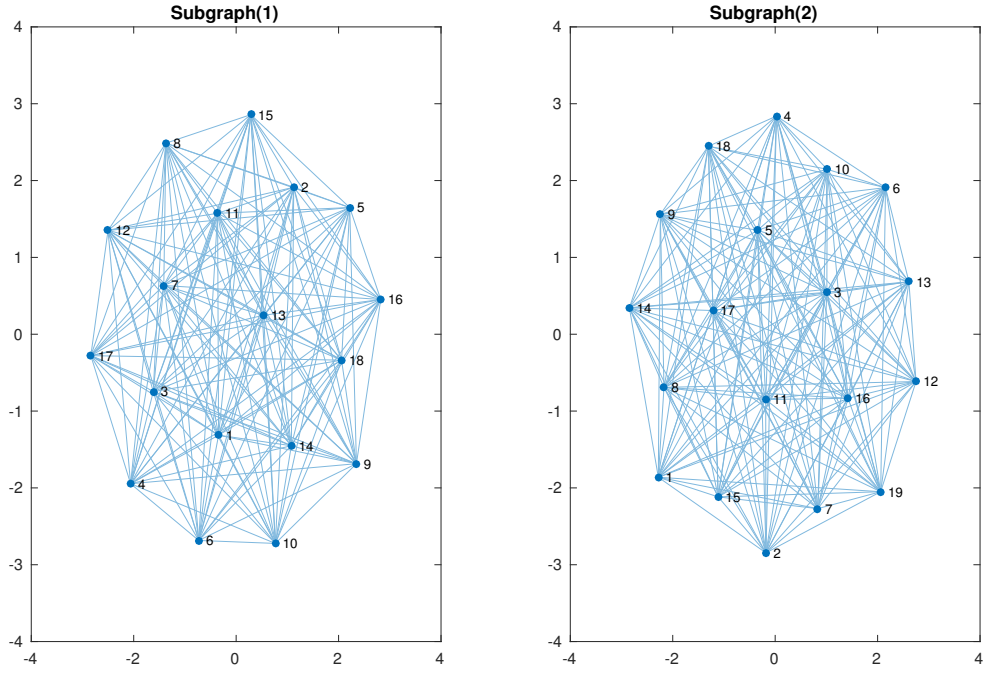

Figure 7.9: The resulted subgraphs after running Partitioning module

Chapter 8

Conclusions and Future Work

8.1 Summary of Achievements

In this research, a comprehensive study of available graph partitioning methods has been featured. Graph partitioning methods are recognized as one of the most important and functional algorithms in combinatorics and has a wide range of application in computer science. This research has provided a structured survey of graph partitioning approaches by discussing the main concepts and algorithms proposed in this area.

Initially, the research points out the prominent applications of the graph partitioning problems to demonstrate the widespread demand on these methods. Parallel processing, complex networks, image processing, clustering, and community detection are some of the most important cases in which graph partitioning algorithms are a crucial requirement.

Studying the main available graph partitioning methods reveals that although the global algorithms can work on entire the graph, the algorithms are typically expensive and only restricted to small graphs. On the other hand, the local algorithms are more flexible but a reliable solution can be achieved only by iterations. Moreover, the multilevel graph partitioning approach is introduced as one of the most accurate heuristic method.

This research has had an emphasis on spectral partitioning methods. Accordingly, all the basis of the spectral methods has been discussed in this research. Moreover, the entire of graph spectrum and the related problems has been investigated in this research.

Furthermore, the connection between spectral partitioning and spectral clustering methods is briefly discussed by describing the spectral clustering algorithms. This research illustrates that spectral clustering algorithms actually take advantages of spectral partitioning concepts.

As an extension to this research project, a novel strategy has been suggested to use some state-of-the-art spectral graph partitioning concepts to extend the applica-

bility of group testing algorithms in graph-constrained problems. The key idea in this strategy is partitioning a graph into expander subgraphs and then applying an efficient graph testing algorithms on each subset.

As an another extension to this project, a MATLAB toolbox has been developed alongside the research to improve the understanding of spectral partitioning concepts. By using this toolbox, a user can easily generate a variety of random graphs or import a graph data file and then run the analysis or partitioning modules to see how spectral methods work. Moreover, the graphical user interface provides some visualization tools to improve the user experience and comprehension.

In conclusion, this research can be used as an initial point for conducting researches on graph partitioning algorithms and can be regarded as a comprehensive study on spectral graph partitioning methods.

8.2 Future Work

Although the main objectives of this research has been achieved, there is still some potential topics for future research as it is highlighted in the following:

- The main emphasis of this project is on spectral graph partitioning methods; however, all other partitioning methods described in Chapter 3 can be a topic of research. In particular, multilevel graph partitioning method is one of the most interesting algorithms which is widely used on account of its efficiency and accuracy. Almost all of the applications requiring graph partitioning methods can take advantage of multilevel approaches.Therefore, conduction a comprehensive research on this method can be of interest.
- In this research, spectral clustering methods are briefly described to highlight the relation between spectral clustering and spectral partitioning schemes. An interesting topic for future research, spectral clustering methods can be independently studied.
- The proposed strategy in Chapter 6 is still a raw idea. It can actually be evaluated in a practical research by conducting some numerical testing.
- Finally, the MATLAB toolbox presented alongside this research can be extended for future work in the following ways:
  - Adding more random models for generating random graphs, such as Gleeson's algorithm which is proposed to generate random social network graphs.
  - Adding other partitioning methods. Currently, only bipartitioning and a simple k-partitioning algorithms have been implemented.
  - The GUI and the Analysis module can be improved adding more features to investigate other spectral parameters.

– The proposed approach for group-constrained group testing can also be implemented in this toolbox to evaluate its performance, in practice.

Bibliography

[1] Random regular generator - file exchange - MATLAB central. pages 77

[2] MATLAB - MathWorks - MathWorks United Kingdom. `http://uk.mathworks.com/products/matlab/`, 1994. [Online; accessed 25-May-2016]. pages 74, 79

[3] KMETIS - Graph Partitioning using METIS . `http://people.sc.fsu.edu/~jburkardt/c_src/kmetis/kmetis.html`, 2014. [Online; accessed 24-May-2016]. pages 74

[4] George Alan and Liu Joseph W. Computer solution of large sparse positive definite systems. *SIAM Review*, 26(2):289–291, 1984. pages 17

[5] Charles J Alpert and Andrew B Kahng. Multi-way partitioning via spacefilling curves and dynamic programming. In *Proceedings of the 31st annual Design Automation Conference*, pages 652–657. ACM, 1994. pages 61

[6] Charles J Alpert, Andrew B Kahng, and So-Zen Yao. Spectral partitioning with multiple eigenvectors. *Discrete Applied Mathematics*, 90(1):3–26, 1999. pages xi, 54, 55, 61

[7] Reid Andersen and Kevin J Lang. An algorithm for improving graph partitions. In *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 651–660. Society for Industrial and Applied Mathematics, 2008. pages 17

[8] Sanjeev Arora, Boaz Barak, and David Steurer. Subexponential algorithms for unique games and related problems. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 563–572. IEEE, 2010. pages 46, 47

[9] Sanjeev Arora, Satish Rao, and Umesh Vazirani. Expander flows, geometric embeddings and graph partitioning. *Journal of the ACM (JACM)*, 56(2):5, 2009. pages 68

[10] Eric Aubanel. Resource-aware load balancing of parallel applications. *Handbook of Research on Grid Technologies and Utility Computing: Concepts for Managing Large-Scale Applications*, pages 12–21, 2009. pages 2

[11] Lars Backstrom, Dan Huttenlocher, Jon Kleinberg, and Xiangyang Lan. Group formation in large social networks: membership, growth, and evolution. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 44–54. ACM, 2006. pages 5

[12] Thang Nguyen Bui, Soma Chaudhuri, Frank Thomson Leighton, and Michael Sipser. Graph bisection algorithms with good average case behavior. *Combinatorica*, 7(2):171–191, 1987. pages 17

[13] Aydin Buluç, Henning Meyerhenke, Ilya Safro, Peter Sanders, and Christian Schulz. Recent advances in graph partitioning. *CoRR, abs/1311.3144*, 2013. pages ix, 2, 3, 4, 16, 17, 18, 19, 20, 23, 24, 25, 26

[14] Pak K Chan, Martine DF Schlag, and Jason Y Zien. Spectral k-way ratio-cut partitioning and clustering. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 13(9):1088–1096, 1994. pages xi, 58, 62, 63

[15] J Cheeger. A lower bound for the smallest eigenvalue of the laplacian, problems in analysis: A symposium in honor of salomon bochner (1970), 195-199. *MR0402831 (53# 6645). Zbl*, 212. pages 40

[16] Mahdi Cheraghchi, Amin Karbasi, Soheil Mohajer, and Venkatesh Saligrama. Graph-constrained group testing. *IEEE Transactions on Information Theory*, 58(1):248–262, 2012. pages ix, 65, 66

[17] Michele Coscia, Fosca Giannotti, and Dino Pedreschi. A classification for community discovery methods in complex networks. *Statistical Analysis and Data Mining*, 4(5):512–546, 2011. pages 3, 5

[18] Daniel Delling and Renato F Werneck. Better bounds for graph bisection. In *European Symposium on Algorithms*, pages 407–418. Springer, 2012. pages 17

[19] Ralf Diekmann, Burkhard Monien, and Robert Preis. Using helpful sets to improve graph bisections. *Interconnection networks and mapping and scheduling parallel computations*, 21:57–73, 1995. pages xi, 20, 21

[20] Ralf Diekmann, Robert Preis, Frank Schlimbach, and Chris Walshaw. Shape-optimized mesh partitioning and load balancing for parallel adaptive fem. *Parallel Computing*, 26(12):1555–1581, 2000. pages xi, 23, 24

[21] Chris Ding, Xiaofeng He, Richard F Meraz, and Stephen R Holbrook. A unified representation of multiprotein complex data for modeling interaction networks. *Proteins: Structure, Function, and Bioinformatics*, 57(1):99–108, 2004. pages 59

[22] Chris HQ Ding, Xiaofeng He, Hongyuan Zha, Ming Gu, and Horst D Simon. A min-max cut algorithm for graph partitioning and data clustering. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 107–114. IEEE, 2001. pages 59

[23] William E Donath and Alan J Hoffman. Lower bounds for the partitioning of graphs. *IBM Journal of Research and Development*, 17(5):420–425, 1973. pages 17

[24] Wilm E Donath and Alan J Hoffman. Algorithms for partitioning of graphs and computer logic based on eigenvectors of connection matrices. *IBM Technical Disclosure Bulletin*, 15(3):938–944, 1972. pages 17, 38

[25] Nick Duffield. Network tomography of binary network performance characteristics. *IEEE Transactions on Information Theory*, 52(12):5373–5388, 2006. pages 66

[26] Shantanu Dutt. New faster kernighan-lin-type graph-partitioning algorithms. In *Computer-Aided Design, 1993. ICCAD-93. Digest of Technical Papers., 1993 IEEE/ACM International Conference on*, pages 370–377. IEEE, 1993. pages 20

[27] Peter Elias, Amiel Feinstein, and Claude Shannon. A note on the maximum flow through a network. *IRE Transactions on Information Theory*, 2(4):117–119, 1956. pages 32

[28] Charbel Farhat and Michel Lesoinne. Automatic partitioning of unstructured meshes for the parallel solution of problems in computational mechanics. *International Journal for Numerical Methods in Engineering*, 36(5):745–764, 1993. pages 18

[29] Charles M Fiduccia and Robert M Mattheyses. A linear-time heuristic for improving network partitions. In *Design Automation, 1982. 19th Conference on*, pages 175–181. IEEE, 1982. pages 20

[30] Miroslav Fiedler. Algebraic connectivity of graphs. *Czechoslovak mathematical journal*, 23(2):298–305, 1973. pages 36, 37

[31] Miroslav Fiedler. A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czechoslovak Mathematical Journal*, 25(4):619–633, 1975. pages 17, 36

[32] Maurizio Filippone, Francesco Camastra, Francesco Masulli, and Stefano Rovetta. A survey of kernel and spectral methods for clustering. *Pattern recognition*, 41(1):176–190, 2008. pages ix, 4, 5

[33] LR Ford Jr and DR Fulkerson. Maximal flow through a network. In *Classic papers in combinatorics*, pages 243–248. Springer, 2009. pages 17, 32

[34] Philippe Galinier, Zied Boujbel, and Michael Coutinho Fernandes. An efficient memetic algorithm for the graph partitioning problem. *Annals of Operations Research*, 191(1):1–22, 2011. pages 22

[35] Laurent Galluccio, Olivier Michel, Pierre Comon, and Alfred O Hero. Graph based k-means clustering. *Signal Processing*, 92(9):1970–1984, 2012. pages 5

[36] Shayan Oveis Gharan and Luca Trevisan. Partitioning into expanders. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1256–1266. Society for Industrial and Applied Mathematics, 2014. pages xi, 13, 46, 68, 69, 70

[37] John R Gilbert, Gary L Miller, and Shang-Hua Teng. Geometric mesh partitioning: Implementation and experiments. *SIAM Journal on Scientific Computing*, 19(6):2091–2110, 1998. pages ix, 18, 19

[38] Fred Glover. Tabu search-part i. *ORSA Journal on computing*, 1(3):190–206, 1989. pages 22

[39] Fred Glover. Tabu search—part ii. *ORSA Journal on computing*, 2(1):4–32, 1990. pages 22

[40] Lars Hagen and Andrew B Kahng. New spectral methods for ratio cut partitioning and clustering. *Computer-aided design of integrated circuits and systems, ieee transactions on*, 11(9):1074–1085, 1992. pages xi, 33, 60

[41] Bruce Hendrickson and Robert Leland. The chaco user's guide: Version 2.0. Technical report, Technical Report SAND95-2344, Sandia National Laboratories, 1995. pages 74

[42] Joao P Hespanha. An efficient matlab algorithm for graph partitioning. *Santa Barbara, CA, USA: University of California*, 2004. pages 74

[43] Manuel Holtgrewe, Peter Sanders, and Christian Schulz. Engineering a scalable high quality graph partitioner. In *Parallel & Distributed Processing (IPDPS), 2010 IEEE International Symposium on*, pages 1–12. IEEE, 2010. pages 20

[44] Jan Hungershöfer and Jens-Michael Wierum. On the quality of partitions based on space-filling curves. In *International Conference on Computational Science*, pages 36–45. Springer, 2002. pages 19

[45] Anil K Jain. Data clustering: 50 years beyond k-means. *Pattern recognition letters*, 31(8):651–666, 2010. pages 4, 5

[46] Ravi Kannan, Santosh Vempala, and Adrian Vetta. On clusterings: Good, bad and spectral. *Journal of the ACM (JACM)*, 51(3):497–515, 2004. pages 68

[47] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing*, 20(1):359–392, 1998. pages 17

[48] George Karypis and Vipin Kumar. Multilevelk-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed computing*, 48(1):96–129, 1998. pages 20, 22

[49] Brian W Kernighan and Shen Lin. An efficient heuristic procedure for partitioning graphs. *Bell system technical journal*, 49(2):291–307, 1970. pages 20

[50] Jeong Han Kim and Van H Vu. Generating random regular graphs. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 213–222. ACM, 2003. pages 77

[51] Suh-Ryung Kim, Jung Yeun Lee, Boram Park, and Yoshio Sano. The competition graphs of oriented complete bipartite graphs. *Discrete Applied Mathematics*, 201:182 – 190, 2016. pages ix, 9

[52] Shad Kirmani and Padma Raghavan. Scalable parallel graph partitioning. In *Proceedings of the international conference on high performance computing, networking, storage and analysis*, page 51. ACM, 2013. pages 19

[53] Ailsa H Land and Alison G Doig. An automatic method of solving discrete programming problems. *Econometrica: Journal of the Econometric Society*, pages 497–520, 1960. pages 16

[54] Kevin Lang and Satish Rao. A flow-based method for improving the expansion or conductance of graph cuts. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 325–337. Springer, 2004. pages 17

[55] James R Lee, Shayan Oveis Gharan, and Luca Trevisan. Multiway spectral partitioning and higher-order cheeger inequalities. *Journal of the ACM (JACM)*, 61(6):37, 2014. pages ix, 43, 46, 47, 48, 68, 69

[56] Tom Leighton and Satish Rao. An approximate max-flow min-cut theorem for uniform multicommodity flow problems with applications to approximation algorithms. In *Foundations of Computer Science, 1988., 29th Annual Symposium on*, pages 422–431. IEEE, 1988. pages 58, 68

[57] Juan Li and Chen-Ching Liu. Power system reconfiguration based on multilevel graph partitioning. In *PowerTech, 2009 IEEE Bucharest*, pages 1–5. IEEE, 2009. pages 3

[58] Nathan Linial, Eran London, and Yuri Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15(2):215–245, 1995. pages 68

[59] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982. pages 38

[60] László Lovász. Random walks on graphs. *Combinatorics, Paul erdos is eighty*, 2:1–46, 1993. pages 25

[61] Henning Meyerhenke and Stefan Schamberger. Balancing parallel adaptive fem computations by solving systems of linear equations. In *European Conference on Parallel Processing*, pages 209–219. Springer, 2005. pages 24

[62] Burkhard Monien and Stefan Schamberger. Graph partitioning with the party library: Helpful-sets in practice. In *Computer Architecture and High Performance Computing, 2004. SBAC-PAD 2004. 16th Symposium on*, pages 198–205. IEEE, 2004. pages 20, 24

[63] Maria CV Nascimento and Andre CPLF De Carvalho. Spectral methods for graph clustering–a survey. *European Journal of Operational Research*, 211(2):221–231, 2011. pages xi, 2, 4, 16, 29, 33, 57, 58, 61, 63, 64

[64] Mark EJ Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113, 2004. pages 59

[65] Vitaly Osipov and Peter Sanders. n-level graph partitioning. In *European Symposium on Algorithms*, pages 278–289. Springer, 2010. pages 20, 27

[66] Gergely Palla, Imre Derényi, Illés Farkas, and Tamás Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, 2005. pages 5

[67] Bo Peng, Lei Zhang, and David Zhang. A survey of graph theoretical approaches to image segmentation. *Pattern Recognition*, 46(3):1020–1038, 2013. pages 3

[68] P. Raghavendra and D. Steurer. Graph expansion and the unique games conjecture. *Proceedings of STOC. ACM, New York*, pages 755–764, 2010. cited By 6. pages 46

[69] Prasad Raghavendra and David Steurer. Graph expansion and the unique games conjecture. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 755–764. ACM, 2010. pages 46

[70] Peter Sanders and Christian Schulz. Engineering multilevel graph partitioning algorithms. In *European Symposium on Algorithms*, pages 469–480. Springer, 2011. pages 20, 23, 26

[71] Peter Sanders and Christian Schulz. High quality graph partitioning. *Graph Partitioning and Graph Clustering*, 588(1), 2012. pages 22, 23

[72] Peter Sanders and Christian Schulz. Think locally, act globally: Highly balanced graph partitioning. In *International Symposium on Experimental Algorithms*, pages 164–175. Springer, 2013. pages 22

[73] Satu Elisa Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27–64, 2007. pages 4, 5, 7, 25

[74] Stefan Schamberger. On partitioning fem graphs using diffusion. In *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*, page 277. IEEE, 2004. pages 24

[75] Jonathan Richard Shewchuk. Ladies and gentlemen, allow me to introduce spectral and isoperimetric graph partitioning. 2011. pages 29, 30, 32, 36, 38, 39, 40

[76] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pages 731–737. IEEE, 1997. pages 58, 59

[77] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000. pages 59

[78] Horst D Simon. Partitioning of unstructured problems for parallel processing. *Computing Systems in Engineering*, 2(2-3):135–148, 1991. pages 18

[79] Horst D Simon and Shang-Hua Teng. How good is recursive bisection? *SIAM Journal on Scientific Computing*, 18(5):1436–1445, 1997. pages xi, 21, 22

[80] Daniel Spielman. Spectral graph theory. *Lecture Notes, Yale University*, pages 740–0776, 2009. pages 10, 12, 13, 16, 31, 34, 35, 40, 43

[81] Isabelle Stanton and Gabriel Kliot. Streaming graph partitioning for large distributed graphs. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1222–1230. ACM, 2012. pages 19

[82] Angelika Steger and Nicholas C Wormald. Generating random regular graphs quickly. *Combinatorics, Probability and Computing*, 8(04):377–396, 1999. pages 77

[83] Mechthild Stoer and Frank Wagner. A simple min-cut algorithm. *Journal of the ACM (JACM)*, 44(4):585–591, 1997. pages 32

[84] M Szell, R Lambiotte, and S Thurner. Trade, conflict and sentiments: multi-relational organization of large-scale social networks, 2010. *arXiv preprint arXiv:1003.5137*. pages 3

[85] Mamoru Tanaka. Higher eigenvalues and partitions of a graph. Technical report, 2011. pages 69

[86] Mamoru Tanaka. Multi-way expansion constants and partitions of a graph. *arXiv preprint arXiv:1112.3434*, 2011. pages 69

[87] Luca Trevisan. Max cut and the smallest eigenvalue. *SIAM Journal on Computing*, 41(6):1769–1786, 2012. pages 44, 45

[88] Rafael Van Driessche and Dirk Roose. An improved spectral bisection algorithm and its application to dynamic load balancing. *Parallel computing*, 21(1):29–48, 1995. pages 38

[89] Jan Van Leeuwen. *Handbook of theoretical computer science (vol. A): algorithms and complexity*. Mit Press, 1991. pages 16

[90] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007. pages xi, 64

[91] Olivier L. De Weck. MIT strategic engineering research group. `http://strategic.mit.edu/downloads.php?page=matlab_networks`, 2015. [Online; accessed 29-May-2016]. pages 74

[92] Yen-Chuen Wei and Chung-Kuan Cheng. Towards efficient hierarchical designs by ratio cut partitioning. In *Computer-Aided Design, 1989. ICCAD-89. Digest of Technical Papers., 1989 IEEE International Conference on*, pages 298–301. IEEE, 1989. pages 58

[93] Douglas Brent West et al. *Introduction to graph theory*, volume 2. Prentice hall Upper Saddle River, 2001. pages 7

Appendices

A Notations and Symbols

x, y, z, X, Y, Z	Scalars (lower-case or upper-case letters)
$\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$	Vectors (lower-case bold letters)
$\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}$	Matrices (upper-case bold letters)
x_i	Entry at position i of vector $\boldsymbol{x}$
X_{ij}	Entry at position (i, j) of matrix $\boldsymbol{x}$
$\boldsymbol{x}^T, \boldsymbol{X}^T$	Transpose of vector $\boldsymbol{x}$ and matrix $\boldsymbol{X}$
$\mathbb{R}$	The set of real numbers
$\mathbb{R}^n$	The set of all vector of length n
$\mathbb{R}^{m \times n}$	The set of all real matrix of size $m \times n$
$\lvert \cdot \rvert$	Number of members of a set
$\lVert \cdot \rVert$	Norm
$\langle \cdot, \cdot \rangle$	Inner product
$\{ \cdots \}$	A set of elements
G	A general graph $G = (V, E)$
G_i	The i-th subgraph of G
C_i	The i-th cluster
V	Set of graph nodes
V_i	The i-th subset of V
n	The number of graph vertices $\lvert V \rvert$ (The order of the graph)
V	Set of graph elements
m	The number of graph elements $\lvert E \rvert$ (The size of the graph)
Δ	The maximum degree of the graph
K_n	A complete graph of order n
$\boldsymbol{A}$	The adjacency matrix of a graph
$\boldsymbol{\mathcal{A}}$	The normalized adjacency matrix of a graph
$\boldsymbol{L}$	The Laplacian matrix of a graph
$\boldsymbol{\mathcal{L}}$	The normalized Laplacian matrix of a graph
$\boldsymbol{W}$	The weight matrix of a graph
$\boldsymbol{D}$	The degree matrix of a graph
$\boldsymbol{I}$	Identity matrix
λ_i	The i-th eigenvalue
Λ	A diagonal matrix of eigenvalues
$\boldsymbol{v_i}$	The i-th eigenvectors
$\boldsymbol{V}$	A matrix of all eigenvectors
$\delta(\cdot)$	The edge separator of a cut
$vol(\cdot)$	The summation of node degrees within a subset
$\phi(\cdot)$	Conductance of a graph
$\rho(k)$	Order-k conductance of a graph
Π^k	Set of all possible k-way partitions
π^k	a k-way partition
$O(\cdot)$	Asymptotic big O notation
$\Omega(\cdot)$	Asymptotic big Ω notation

ϵ	An arbitrary small positive value
$\min\{x_1, ..., x_N\}$	The smallest among scalers
$\min_x f(x)$	The minimum value of real function f with respect to x
$\max\{x_1, ..., x_N\}$	The smallest among scalers
$\max_x f(x)$	The minimum value of real function f with respect to x
$\text{argmax}_x f(x)$	Argument of the maxima (the points at which the function values are maximized)

B User Guide

The developed graph partitioning toolbox can easily be installed into MATLAB as an add-on. All the required files to execute the toolbox including the graphical user interface and the functions have been packaged into single installation file (GPtool.mltbx).

The user can install the toolbox by navigating to the folder including the .mltbx file from within MATLAB and double click on the installation file. This will pull up an installation dialog. By clicking on Install button, the toolbox will be added as an add-on to the MATLAB. The end users do not need to be concerned with the MATLAB path or other installation details. The .mltbx installation file manages these details for end users. The user can also review and manage the toolbox by choosing Manage Add-Ons in Add-Ons menu.

To execute the toolbox, it is enough to run the main GPtool function simply by typing the following in MATLAB command window:

```
>>GPtool
```