UNIVERSITY OF CAGLIARI

Ph.D. PROGRAM IN MATHEMATICS AND
COMPUTER SCIENCE
Cycle XXXII

Numerical Linear Algebra applications in Archaeology:

the seriation and the photometric stereo problems

Scientific Disciplinary Sector: MAT/08 Numerical Analysis

Ph.D. Candidate: Anna Concas

Ph.D. PROGRAM COORDINATOR:
Prof. Michele Marchesi

SUPERVISORS:
Prof. Giuseppe Rodriguez
Dr. Caterina Fenu

Final exam
Academic Year 2018-2019
Thesis defence: January-February 2020 Session

A.C. gratefully acknowledges Sardinia Regional Government for the financial support of her PhD scholarship (P.O.R. Sardegna F.S.E. - Operational Programme of the Autonomous Region of Sardinia, European Social Fund 2014-2020 - Axis III Education and training, Thematic goal 10, Investment Priority 10ii), Specific goal 10.5.

The proper place of Archaeology is the faculty of Mathematics.

David L. Clarke, 1968

Acknowledgments

Everything I have done during these three years, would not have been possible without the support that I have received from many people.

Firstly, I wish to express my deepest gratitude to my supervisor, Professor Giuseppe Rodriguez, for being a patient guide in this research experience, for his huge help and immense knowledge. I could not have imagined having a better mentor! A very special thank to my co-supervisor Dr. Caterina Fenu for being also a real friend! Without her constant support and valuable advice, this research would not have been possible.

My gratitude is also extended to Prof. Raf Vandebril, who supervised my research visit in Leuven, for his suggestions and for giving me the opportunity to join the Leuven group. I would like to express my sincere gratitude to Prof. Lothar Reichel for the ongoing collaboration and for the inspiring discussions during my stay in Kent. A very special thank to Prof. Matteo Sommacal, my supervisor during my visit in Newcastle upon Tyne, for the countless ideas and stimulating discussions, for the careful review of this thesis and for having made me feel at home together with his beautiful family.

I gratefully acknowledge the Northumbria University and in particular the Department of Mathematics, Physics and Electrical Engineering for the funding received that supported my research visit in Newcastle.

A sincere thank to my high school Math teacher, Prof.ssa Perra. Without her, I surely would never have the courage to start a scientific academic carrier after classical studies.

I am also extremely grateful to the whole group of people who were based in Viale Merello, especially to the member of the CaNA (Cagliari Numerical Analysis) Group I haven't mentioned yet: Alessandro who was extremely helpful during my stay in Kent, Federica, Luisa and Patricia.

My sincere appreciation to all the people I had the pleasure of meeting during this three years and, more than anyone, the people I met in Newcastle with whom the time spent has been memorable. In particular I wish to thank the italian group and especially Marta who was a very good friend. I really hope to see all of them again!

I am incredibly thankful to all my friends and particularly to Sara for always comforting me. Her sincere friendship is essential for me!

A heartfelt thank you to Francesco, my anchor in this experience and in life. His determination is an ispiration for me!

Finally, I would like to deeply thank my parents Bruno and Teresa and my sister Daniela (and also my two feline sisters Maia and Gioia) for being always by my side and for supporting me throughout these years and the experiences far from home. Un ringraziamento anche a mia nonna Giovanna, a tutto il resto della mia famiglia e alla famiglia di Francesco che considero parte della mia.

Abstract

The aim of this Thesis is to explore the application of Numerical Linear Algebra to Archaeology. An ordering problem called the *seriation problem*, used for dating findings and/or artifacts deposits, is analysed in terms of *graph theory*. In particular, a Matlab implementation of an algorithm for *spectral seriation*, based on the use of the Fiedler vector of the Laplacian matrix associated with the problem, is presented. We consider *bipartite graphs* for describing the seriation problem, since the interrelationship between the units (i.e. archaeological sites) to be reordered, can be described in terms of these graphs. In our archaeological metaphor of seriation, the two disjoint nodes sets into which the vertices of a bipartite graph can be divided, represent the excavation sites and the artifacts found inside them.

Since it is a difficult task to determine the closest bipartite network to a given one, we describe how a starting network can be approximated by a bipartite one by solving a sequence of fairly simple optimization problems.

Another numerical problem related to Archaeology is the 3D reconstruction of the shape of an object from a set of digital pictures. In particular, the *Photometric Stereo* (PS) photographic technique is considered.

Contents

List of Figures

List of Tables

Introduction

Mathematics is widely used in Archaeology and it represents an almost indispensable tool at any stage of archaeological procedures such as documentation, data recording during excavations, and digital archiving, just to mention some of the several phases in archaeological investigations.

A crucial part of the archaeologists' work consists of dating findings and/or artifacts deposits. The deposits may be graves in a prehistoric cemetery and may contain artifacts such as pottery, jewellery or utensils that have been classified into types by considering their material, shape, style of decoration, etc. When archaeological deposits cannot be dated by deciphering inscriptions or applying physical or chemical techniques, the use of mathematical and statistical techniques is crucial for solving the problem of sequencing the archaeological sites in a chronological order.

A fundamental ordering problem of particular interest to archaeologists, is the so-called *seriation problem*. By asking to find the best enumeration order of a set of elements according to a given correlation function, so that elements with higher similarity are closer in the resulting sequence, the seriation problem consists of ordering the archaeological sites in a chronological order. It is based on the concept that archaeological units having similar contents, as determined by the arrangement into types, should be closer in time than deposits with dissimilar findings.

The pioneer of seriation is considered to be the Archaeologist Petrie who introduced seriation to sequence, in a chronological order, tombs found in the Nile area at the end of XIV century [152]. He used a cross tabulation of graves and objects found inside them. Initially, the rearrangement of rows and columns of this contingency table was done manually. Later, many different algorithms have been proposed to solve the seriation problem. In particular, a matrix method was first developed in the 50s with the works of the archaeologist Brainerd [25] and the statistician Robinson [158]. The proposed method is based on the ordering of the *similarity matrix* that contains the correlations between the sites to be chronologically reordered. Later, Kendall [108] proposed a procedure for directly ordering the *data matrix* which is the simplest representation of archaeological data. See [165] for the description of fairly recent development of the use of matrices and networks to solve the seriation problem.

A spectral algorithm, developed first by Barnard et al. [12] for computing an envelope-reducing reordering of sparse matrices, was considered for the solution of the seriation problem by Atkins et al. [6]. In this Thesis, we present a Matlab implementation of the above algorithm for *spectral seriation*. The algorithm is based on the use of the Fiedler vector of the Laplacian matrix associated with the problem, which encodes the set of admissible solutions into a compact data structure called *PQ-tree*. In particular, we describe a Matlab toolbox for spectral seriation and for manipulating the PQ-tree data structure containing all the possible reorderings that solve the seriation problem in case of *consistent data*. Therefore, we point

out that the presence of a multiple Fiedler value may have a substantial influence on the computation of an approximated solution, in the presence of inconsistent data sets.

The seriation problem is modelled here in terms of *graph theory*. Indeed, in the archaeological application, seriation consists in finding the best arrangement of a set of the units, to be reordered, whose interrelationship can be defined by a *bipartite graph*. A graph is said to be bipartite if its nodes can be divided into two nonempty sets such that there are no edges between nodes in the same set. In our archaeological metaphor of seriation, the two disjoint nodes sets represent the excavation sites and the artifacts found inside them.

Bipartivity is a fundamental topological property of graphs and networks and determining if a graph is bipartite or not, is equivalent to observe if the graph is *two-colourable* or not [20]. This perfect separation into two colours (or classes more in general) is rare in most of the real-world situations. Hence, in the literature there have been some efforts to measure the quantity of bipartivity a non-bipartite graph has [68, 95]. Some spectral approaches, based on the eigedecompostion of the adjacency matrix or of other matrices that describe the network, have been proposed by various authors for detecting approximately bipartite structures in given networks [44, 51].

With the aim of finding an approximate solution to the the difficult task of determining the "closest" bipartite network to a given one, we describe how a starting network can be approximated by a bipartite one by solving a sequence of fairly simple optimization problems. The proposed *spectral "bipartization" algorithm* detect an approximate bipartization and produces a node permutation which highlights the possible bipartite nature of the initial adjacency matrix identifying the two sets of nodes expected in a bipartite structure. The same computational procedure can also be used to detect the presence of a large *anti-community*, that is a group of nodes loosely connected internally but with many external links, in a network and identify it.

Another application of Numerical Linear Algebra in Archaeology, considered in this Thesis, is a classical problem in Computer Vision consisting of the reconstruction of the 3D shape of an object from a set of 2D digital pictures. To solve this *inverse problem* many different techniques have been used [97]. We consider a particular photographic technique called photometric stereo (PS). PS exploits shading information when several light sources illuminate an object and requires that the object is observed from a fixed point but under different lighting conditions. The fact that PS uses only a fixed camera and a movable light to acquire a set of digital pictures to generate 3D scans of the observed object, makes it an efficient tool for 3D recording engravings and relieves.

While the traditional approach requires that the position of the light sources and the illumination intensity used for taking the digital images are accurately known, we explore photometric stereo under unknown lighting.

The Thesis is divided into 4 chapters and a brief description of the content of each one is reported below.

Chapter 1 introduces preliminary notions of Numerical Linear Algebra and recall various mathematical tools used in the rest of the Thesis. We remind some fundamental matrix factorizations recurrent in various parts of the whole work. Few classes of matrices are briefly analysed, together with basic concepts of graph theory. A brief outline on permutations is given and we also consider some notions of computational complexity theory since some problems considered are characterised to be *difficult* to solve.

Chapter 2 discusses how the seriation problem can be described using the terminology typical of graph theory. Focusing on the archaeological application of seriation, we describe a Matlab implementation of a spectral algorithm together with the toolbox that provides the functions to manipulate and visualize a compact data structure, called *PQ-tree*, into which the set of admissible solutions of the seriation problem are encoded.

Chapter 3 illustrates a spectral method for approximating bipartite networks which models the interaction between two different types of object. The "bipartization" method exploits the spectral structure of bipartite graphs and determines a node permutation that separates the nodes of a given network into two disjoint set, in order to approximate a connected undirected graph by a bipartite one. An application of the algorithm to the detection of a large anti-community is considered.

Chapter 4 describes the photometric stereo problem, with application to archaeology, in the 3D reconstruction of the shape of an object starting from a set of 2D digital pictures. While the classical approach requires that the position of the light sources is accurately known, we illustrate an approach for solving the photometric stereo problem under unknown lighting.

Chapter 1

Mathematical Preliminaries

With the aim of making the Thesis self-contained, in this Chapter we briefly review some notions and results of Linear Algebra that will be useful in the rest of the Thesis for delineating how Numerical Linear Algebra can be used in the framework of Archaeology.

Although the definitions and results below are given in general for complex matrices, we are concerned about matrices having real entries.

1.1 Spectral decomposition, singular value decomposition and polar decomposition

Before recalling some of the most important factorizations, we quickly analyse few fundamental concepts of Linear Algebra.

Given a set of vectors $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ in R^m, the vectors $\mathbf{v}_j$ are *linearly independent* if $\sum_{i=1}^{n} \alpha_i \mathbf{v}_i = 0$ implies $\alpha_1 = \cdots = \alpha_n = 0$. On the contrary, the set of vectors is said to be *linearly dependent* if at least one of the vectors in the set can be defined as a linear combination of the others, i.e., a nontrivial combination of the vectors $\mathbf{v}_i$ is zero.

Let $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ vectors in R^m; the set of all linear combinations of these vectors is a subspace called *span* of $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$

$$\text{span}\{\mathbf{v}_1, \ldots, \mathbf{v}_n\} = \{\sum_{j=1}^{n} \beta_j \mathbf{v}_j : \beta_j \in \mathbb{R}\}.$$

A set of vectors in a vector space $\mathcal{V}$ is a *basis* if its elements are linearly independent and every element of $\mathcal{V}$ is a linear combination of elements of the considered set.

1.1.1 Null space, range and rank

The *range* of a $m \times n$ matrix A, is the set of vectors, denoted here as $\text{ran}(A)$, defined by

$$\text{ran}(A) = \{\mathbf{y} \in \mathbb{R}^m : \mathbf{y} = A\mathbf{x} \quad \text{for some} \quad \mathbf{x} \in \mathbb{R}^n\}.$$

The fact that any vector given by the product $A\mathbf{x}$ can be written as

$$\mathbf{b} = A\mathbf{x} = \sum_{j=i}^{n} x_j \mathbf{a}_j,$$

where $\mathbf{a}_j$ denotes the j-th column of A, leads to the characterization of $\mathrm{ran}(A)$ expressed by the following result; see [176].

Theorem 1.1.1. $\mathrm{ran}(A)$ *is the space spanned by the columns of A, i.e.,*

$$\mathrm{ran}(A) = \mathrm{span}\{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$$

if $A = [\mathbf{a}_1 \cdots \mathbf{a}_n]$ *is a column partitioning of A.*

The *nullspace* of A, indicated with $\mathrm{null}(A)$ here, is the set of vectors defined by

$$\mathrm{null}(A) = \{\mathbf{x} \in \mathbb{R}^n : A\mathbf{x} = \mathbf{0}\},$$

where $\mathbf{0}$ is the null vector in $\mathbb{R}^m$.

The *column rank* of a matrix is the dimension of its space spanned by its columns and, similarly, the *row rank* of a matrix is the dimension of the row space. We may also define the column rank of a matrix A as the maximum number of linearly independent columns and, in the same way, we may say that the row rank is the maximum number of linearly independent rows of A. Column rank equals row rank for every $m \times n$ matrix A; a proof of this statement can be found for example in [118]. We refer to this number as the *rank* of the considered matrix A. Hence, the rank of a matrix A is defined by

$$\mathrm{rank}(A) = \dim(\mathrm{ran}(A)),$$

i.e., the rank of a matrix is the dimension of its column (or rows) space.

1.1.2 Eigenvalues and eigenvectors

Definition 1.1.1. (Eigenvalues and eigenvectors). *Let* $A \in \mathbb{C}^{n\times n}$ *be a square matrix. An eigenvector of A is a vector* $\mathbf{v} \neq \mathbf{0} \in \mathbb{C}^n$ *such that*

$$A\mathbf{v} = \lambda\mathbf{v}, \quad \lambda \in \mathbb{C}.$$

The scalar λ *is called the eigenvalue associated to the eigenvector* $\mathbf{v}$.

From the definition it follows that the eigenvectors of a matrix A are the non-trivial solutions of the linear system $(A - \lambda I)\mathbf{v} = \mathbf{0}$ and hence the eigenvalues are the roots of the *characteristic polynomial* of A defined by $p_A(z) = \det(A - zI)$. The set of these roots, solutions of the characteristic equation $\det(A - zI) = 0$, is the so-called *spectrum* of A and denoted here as $\sigma(A)$. The *spectral radius* $\rho(A)$ is defined by

$$\rho(A) = \max_{i=1,\ldots,n} \{|\lambda_i| : \lambda_i \in \sigma(A)\}.$$

By the fundamental theorem of algebra it follows that the characteristic polynomial has exactly n roots, counted with multiplicity, i.e., every square matrix of order n has n, possibly not distinct, eigenvalues. The *algebraic multiplicity* of an eigenvalue λ of A is defined to be its multiplicity as a root of p_A. An eigenvalue with algebraic multiplicity 1 is said to be *simple*.

If an eigenvalue of a given matrix A of order n is known, then it is possible to find a matrix of dimension $n - 1$ such that its spectrum contains the same eigenvalues of A except for the one we are using for reducing the problem. This process is called *deflation*.

Note that both the determinant and the trace of a matrix can be characterised in terms of its eigenvalues. Indeed, if $\sigma(A) = \{\lambda_1, \ldots, \lambda_n\}$ is the spectrum of a matrix A of order n, then the determinant and the trace, denoted by $\det(A)$ $\mathrm{trace}(A)$, are equal to the product and the sum of the eigenvalues of A respectively, counted with algebraic multiplicity

$$\det(A) = \prod_{i=1}^{n} \lambda_i \quad \text{and} \quad \mathrm{trace}(A) = \sum_{i=1}^{n} \lambda_i,$$

where the trace function is the sum of the diagonal entries of the considered matrix, i.e.,

$$\mathrm{trace}(A) = \sum_{i=1}^{n} a_{ii}.$$

The set of eigenvectors corresponding to an eigenvalue λ_i of a matrix $A \in \mathbb{C}^{n\times n}$, together with the zero vector, is a subspace of $\mathbb{C}^n$ called *eigenspace* and denoted here by E_{λ_i}. The *geometric multiplicity* of λ_i is the dimension of the corresponding eigenspace E_{λ_i}, i.e. the number of linearly independent eigenvectors associated with λ_i. The geometric multiplicity can also be defined as the dimension of $\mathrm{null}(A - \lambda_i I)$ since this nullspace is again the eigenspace E_{λ_i}.

In general, the algebraic multiplicity of an eigenvalue is at least as great as its geometric multiplicity. An eigenvalue whose algebraic multiplicity exceeds its geometric multiplicity is said to be a *defective* eigenvalue. A matrix with at least one defective eigenvalue is referred to as a defective matrix. Nondefective matrices are also known as *diagonalizable*.

Theorem 1.1.2. (Spectral decomposition). *A matrix $A \in C^{n\times n}$ is diagonalizable if and only if there exists an invertible matrix V of order n and an $n \times n$ diagonal matrix Λ such that*

$$A = V\Lambda V^{-1}. \tag{1.1.1}$$

Proof. See [176]. □

The factorization (1.1.1) is called *eigendecomposition* or *spectral decomposition* of the matrix A.

Hence, a matrix A of size $n \times n$ is diagonalizable if it has n linearly independent eigenvectors and viceversa. If A not only has n linearly independent eigenvectors, but they can also be chosen to be orthogonal, then A is said to be *unitarily diagonalizable*. In such a case, there exists a unitary matrix Q (i.e., $Q^{-1} = Q^*$; a unitary real matrix is said *orthogonal*) such that

$$A = Q\Lambda Q^*,$$

where Q^* is the conjugate transpose of Q, i.e. $Q^* = \overline{Q}^T$. A class of matrices unitarily diagonalizable is given by the Hermitian matrices. Recalling that a complex square matrix A is *Hermitian* if $A = A^*$ (if A is real, then A is called *symmetrix* if $A = A^T$), we have the following well-known result, a proof of which can be found in [176].

Theorem 1.1.3. *A Hermitian matrix is unitarily diagonalizable and its eigenvalues are real.*

The following result states that the eigenvalues of a Hermitian matrix of order n are interlaced with those of any submatrix of A obtained by removing rows and columns of A.

Theorem 1.1.4. (Cauchy's interlacing theorem). *Let A be an $n \times n$ Hermitian matrix and let B be a principal submatrix of A of order $m \leq n$. If $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_{n-1} \leq \lambda_n$ are the eigenvalues of A and $\mu_1 \leq \mu_2 \leq \cdots \leq \mu_m$ are those of B then*

$$\lambda_j \leq \mu_j \leq \lambda_{n-m+j}, \quad \textit{for all} \quad j \leq m.$$

We recall that if $\mathbf{x}$ is an eigenvector of a given matrix, then the Rayleigh quotient, defined by

$$\frac{(A\mathbf{x}, \mathbf{x})}{(\mathbf{x}, \mathbf{x})},$$

where $(\cdot, \cdot)$ denotes the Euclidean inner product on $\mathbb{C}^n$, gives the corresponding eigenvalue. The eigenvalues of Hermitian matrices have a characterization that is concerned with the quadratic form expressed through the Rayleigh quotient, as the following theorem states. See [83] for a proof.

Theorem 1.1.5. (Courant-Fischer Minimax Theorem). *Let $A \in \mathbb{C}^{n \times n}$ be a Hermitian matrix and S be any k-dimensional subspace. Then*

$$\lambda_k(A) = \max_{\dim(S)=k} \quad \min_{\mathbf{0} \neq \mathbf{y} \in S} \quad \frac{\mathbf{y}^T A \mathbf{y}}{\mathbf{y}^T \mathbf{y}},$$

for $k = 1, \ldots, n$.

The Hermitian matrices are not the only ones that are unitarily diagonalizable. Other examples include unitary matrices and circulant matrices, just to mention some of them.

Considering that by definition, a matrix A is said to be *normal* if $AA^* = A^*A$, the class of unitarily diagonalizable matrices can be characterised in terms of the normal matrices, as expressed by the following result.

Theorem 1.1.6. *A matrix is unitarily diagonalizable if and only of it is normal.*

In this brief recap of well-known definitions and results, we do not describe more general factorizations involving the eigenvalues and eigenvectors such as the Schur and Jordan decompositions. The interested reader can consult, for example, references [83, 176].

In some situations one may need only the approximation of few eigenvalues or wishes to have a rough idea of where the eigenvalues lies in the complex plane by considering the so-called *localization* theorems. A simple well-known localization result uses any matrix norm to state that

$$\|\lambda_i\| \leq \|A\|, \quad i = 1, 2, \ldots, n$$

i.e., any eigenvalue of an $n \times n$ matrix belongs to the disc centered at the origin with radius given by $\|A\|$.

In the framework of the eigenvalues localization, the following theorem, due to Gershgorin, may be useful to bound the spectrum of a square matrix in a more precise way.

Theorem 1.1.7. (Gershgorin's Circle Theorem). *Let A be a complex square matrix of order n. Then the spectrum of A is such that*

$$\sigma(A) \subseteq \bigcup_{i=1}^{n} D_i$$

where D_i is the i-th disc of the complex plane defined as

$$D_i = \left\{ z \in \mathbb{C} : |z - a_{ii}| \leq \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}| \right\} \qquad i = 1, \ldots, n. \tag{1.1.2}$$

The previous result, the first of three Gershgorin's theorems, defines the so-called *row circles*. Given a square matrix A of order n, the *column circles* are the sets defined by

$$C_j = \left\{ z \in \mathbb{C} : |z - a_{jj}| \leq \sum_{\substack{i=1 \\ i \neq j}}^{n} |a_{ij}| \right\} \qquad j = 1, \ldots, n \tag{1.1.3}$$

and, applying the above theorem to A^T, it follows that

$$\sigma(A) \subseteq \bigcup_{j=1}^{n} C_j.$$

Denoting by $\mathcal{S}_r$ and $\mathcal{S}_c$ the union of the row discs (1.1.2) and column discs (1.1.3), respectively, i.e.

$$\mathcal{S}_r = \bigcup_{i=1}^{n} D_i, \quad \mathcal{S}_c = \bigcup_{j=1}^{n} C_j,$$

it then follows that

$$\sigma(A) \subseteq \mathcal{S}_r \cap \mathcal{S}_c.$$

The result below is knows as *the second Gershgorin's theorem*.

Theorem 1.1.8. *Suppose that there are k Gerschgorin discs (1.1.2) whose union $\mathcal{S}_1$ is disjoint from all other $n - k$ discs, so that $\mathcal{S}_r$ is given by the union of*

$$\mathcal{S}_1 = \bigcup_{i=1}^{k} D_i, \quad \mathcal{S}_2 = \bigcup_{i=k+1}^{n} D_i, \quad \mathcal{S}_1 \cap \mathcal{S}_2 = \emptyset.$$

Then $\mathcal{S}_1$ contains exactly k eigenvalues, (counted with their multiplicities) and $\mathcal{S}_2$ the remaining $n - k$.

1.1.3 Singular value decomposition

Another fundamental decomposition, that will be useful in different parts of the Thesis, is the *singular value decomposition*, shortened here as SVD. The existence of the SVD for matrices was established in a linear algebra context independently by Beltrami [15], Jordan [103] and Sylvester [171]. Schmidt [161] and Weyl [182] approached the decomposition from the framework of the integral equations, by developing an infinite-dimensional generalization. For a survey on the history of the singular value decomposition see [169].

Definition 1.1.2. (Singular values and singular vectors). *Given a matrix $A \in \mathbb{C}^{m \times n}$, a number $\sigma \geq 0$ is called a singular value for A if and only if there exist unit vectors $\mathbf{u} \in \mathbb{C}^m$ and $\mathbf{v} \in \mathbb{C}^n$ such that*

$$A\mathbf{v} = \sigma\mathbf{u} \quad \textit{and} \quad A^*\mathbf{u} = \sigma\mathbf{v}.$$

The vectors $\mathbf{u}$ and $\mathbf{v}$ are called left-singular vectors and right-singular vectors of A, respectively.

Theorem 1.1.9. (Singular value decomposition). *Let $A \in \mathbb{C}^{m\times n}$ be a complex matrix. Then there exist two unitary matrices*

$$U = [\mathbf{u}_1 \quad \cdots \quad \mathbf{u}_m] \in \mathbb{C}^{m\times m} \qquad \text{and} \qquad V = [\mathbf{v}_1 \quad \cdots \quad \mathbf{v}_n] \in \mathbb{C}^{n\times n}$$

such that

$$A = U\Sigma V^* \tag{1.1.4}$$

where $\Sigma = \operatorname{diag}(\sigma_1, \ldots, \sigma_p) \in \mathbb{R}^{m\times n}$ with $p = \min\{m, n\}$.

The identity (1.1.4) is called the Singular Value Decomposition (SVD) of the matrix A. The diagonal matrix Σ contains the singular values of A conventionally ordered as $\sigma_1 \geq \sigma_2 \geq \ldots \sigma_p$ and the columns of U and V are the corresponding (properly normalized) left and right singular vectors, respectively.

The SVD has a geometric meaning. Specifically, given a matrix A, the semiaxis directions of the hyperellipsoid E defined by $E = \{A\mathbf{x} : \|\mathbf{x}\|_2 = 1\}$ are given by the left singular vectors and their lengths are the singular values of A. See [176] for a complete description of the geometric interpretation of the SVD of a real matrix.

If $U^*AV = \Sigma$ is the singular value decomposition of $A \in \mathbb{C}^{m\times n}$ and $m \geq n$, then, from Definition 1.1.2, it follows immediately that, for $i = 1, \ldots, n$

$$A^*A\mathbf{v}_i = \sigma_i^2\mathbf{v}_i, \tag{1.1.5}$$

$$AA^*\mathbf{u}_i = \sigma_i^2\mathbf{u}_i. \tag{1.1.6}$$

This shows that there is a connection between the SVD of A and the eigensystem of the Hermitian matrices A^*A and AA^*. Specifically, from (1.1.5) it follows that the right singular vectors of A are eigenvectors of A^*A and the identity (1.1.6) ensures that the left singular vectors of A are eigenvectors of AA^*. Therefore, the nonzero singular values of A are the square roots of the nonzero eigenvalues of both A^*A and AA^*. If $A = A^*$, then the singular values of A are the absolute values of the eigenvalues of A.

Assume that A has dimension $m \times n$. The number of zero singular values of A is $p - r$ where $p = \min\{m, n\}$ and $r \leq p$, the number of nonzero singular values of A, is such that $r = \operatorname{rank}(A)$. It can be shown that

$$\operatorname{ran}(A) = \operatorname{span}\{\mathbf{u}_1, \ldots, \mathbf{u}_r\} \quad \text{and} \quad \operatorname{null}(A) = \operatorname{span}\{\mathbf{u}_{r+1}, \ldots, \mathbf{u}_p\}.$$

In a wide range of applications, the factorization (1.1.4), that is also known as the *full* SVD, is rarely required especially for the fact that its computation is expensive. It is often sufficient and, above all, faster to compute a reduced version of the SVD. The following decomposition of a matrix considers only the columns of the matrices U and V associated with nonzero singular values and hence it contains the essential singular value decomposition information.

Definition 1.1.3. (Compact SVD). *The* compact *SVD is the factorization*

$$A = U_r\Sigma_r V_r^* \in \mathbb{C}^{m\times n}$$

where U_r and V_r are a $m \times r$ and $n \times r$ matrices respectively, with orthonormal columns that are the left and right singular vectors, respectively, corresponding to the r nonzero singular values of A.

An even more "economical" reduced version of the singular value decomposition is obtained by considering only the first k singular values.

Definition 1.1.4. (Truncated SVD). *The* truncated *SVD of $A \in \mathbb{C}^{m\times n}$ is defined as the matrix*

$$A_k = U_k \Sigma_k V_k^*, \qquad \Sigma_k = \operatorname{diag}(\sigma_1, \ldots, \sigma_k) \in \mathbb{R}^{k\times k}$$

where Σ_k is obtained by replacing with zeros the smallest $n-k$ singular values with $k \leq r$. The matrices U_k and V_k are of size $m \times k$ and $n \times k$, respectively.

The matrix A_k is a rank-k matrix and, in particular, it it the best rank-k approximation of A in both the spectral and the Frobenius norm. This result is known as the Eckart-Young-Mirsky theorem [62, 137]. The matrix A_k generally differs from the original one in all its entries. See [82] for a generalization of the theorem for obtaining a best approximation of lower rank such that a specified set of columns of the matrix remains fixed.

Even though there is a connection between the SVD and the spectral decomposition, they have some fundamental differences. First of all, we saw that not all the matrices have an eigendecomposition, whereas all matrices (even rectangular ones) have a singular value decomposition; see Theorem 4.1 in [176]. Another difference between the two factorizations is that while the eigenvalue decomposition uses just one basis (given by the set of the eigenvectors) that is not orthogonal in general, the SVD uses two different orthonormal bases the sets of left and right singular vectors.

1.1.4 Polar decomposition

The *polar decomposition* is the generalization to matrices of the polar representation of a complex number $z = a + ib$, given by $z = re^{i\theta}$ where $r = \sqrt{a^2+b^2}$ and $e^{i\theta}$ is defined by $(\cos(\theta), \sin(\theta)) = (a/r, b/r)$.

The polar decomposition is closely related to the SVD and it is the representation of a matrix as a product of a symmetric positive definite matrix and a unitary matrix as the following result states.

Theorem 1.1.10. (Polar decomposition). *Let $A \in \mathbb{C}^{m\times n}$ with $m \geq n$. Then there exists a matrix $U \in \mathbb{C}^{m\times n}$ with orthonormal columns and a unique Hermitian positive semidefinite matrix $H \in \mathbb{C}^{n\times n}$ such that $A = UH$.*

Proof. See [91]. □

The matrix U is called the *orthogonal polar factor* and we refer to H as the *symmetric polar factor*.

The polar decomposition states that any square matrix A can be factorized in the form $A = UH$ where U is orthogonal and H is symmetric and positive semidefinite. Note that if $A = U\Sigma V^T$ is the SVD of A, then $A = (UV^T)(V\Sigma V^T)$ is its polar decomposition. See also [83] for further discussion.

1.2 Some classes of matrices

This section collects a variety of definitions that are needed in the following Chapters.

Definition 1.2.1. (Permutation matrix). *A permutation matrix P is a square binary matrix obtained by permuting the rows of an identity matrix.*

Permutation matrices represent permutation of elements and, in particular, the pre-multiplication to a matrix A, to give the matrix PA, results in permuting the rows of the considered matrix A. One obtains a columns permutation of A when considering instead the product AP.

1.2.1 Reducible and nonnegative matrices

Definition 1.2.2. (Reducible matrix). *A square matrix A of order n is reducible if there exists a permutation matrix P such that the matrix $B = PAP^T$ has the upper block triangular form*

$$B = PAP^T = \begin{bmatrix} B_{11} & B_{12} \\ 0 & B_{22} \end{bmatrix},$$

where B_{11} and B_{22} are nonempty square submatrices of size $k \times k$ and $(n-k) \times (n-k)$, respectively. A matrix is said to be irreducible if it is not reducible.

The reducibility of a matrix is a fundamental property and it is strictly related to the connectivity of a graph as it will be explained below.

Definition 1.2.3. (Definiteness of a matrix). *An $n \times n$ Hermitian matrix A is said to be* positive definite *if $\mathbf{x}^* A\mathbf{x} > 0$ for all nonzero vectors $\mathbf{x} \in \mathbb{C}^n$ and* positive semidefinite *if $\mathbf{x}^* A\mathbf{x} \geq 0$ for all nonzero $\mathbf{x} \in \mathbb{C}^n$, where $\mathbf{x}^*$ denotes the conjugate transpose of the vector $\mathbf{x}$. Equivalent conditions for a Hermitian matrix A to be positive (semi)definite is that $\lambda_i(A) > 0$ for all i ($\lambda_i(A) \geq 0$ for all i). Negative definite and negative semidefinite matrices are defined analogously.*

Definition 1.2.4. *A matrix $A \in \mathbb{R}^{m \times n}$ is* nonnegative *if all its elements are nonnegative, i.e. $a_{i,j} \geq 0$ for all i and j. If all the elements are strictly greater than zero, then A is said to be* positive.

The following classical result in matrix theory, proved by Perron and Frobenius, summarizes fundamental spectral properties of nonnegative matrices.

Theorem 1.2.1. (Perron-Frobenius Theorem). *Let A be a real, nonnegative matrix. If we denote with $\rho(A)$ the spectral radius of A, then*

a) $\rho(A)$ is an eigenvalue of A;

b) there is a nonnegative vector $\mathbf{x}$ such that $A\mathbf{x} = \rho(A)\mathbf{x}$.

The following formulation of the Perron-Frobenius theorem consider an extension to irreducible matrices.

Theorem 1.2.2. (Perron-Frobenius Theorem). *If $A \in \mathbb{R}^{n \times n}$ is a nonnegative and irreducible matrix then*

a) $\rho(A) > 0$;

b) $\rho(A)$ is an eigenvalue of A;

c) there is positive eigenvector $\mathbf{x}$ *corresponding to* $\rho(A)$*;*

d) $\rho(A)$ *has algebraic multiplicity 1.*

For further details on Perron-Frobenius theory see [16].

Regarding the eigenvalues localization for irreducible matrices the following theorem can be mentioned.

Theorem 1.2.3. (3rd Gershgorin's theorem or Taussky's theorem). *Let* A *be a* $n \times n$ *irreducible matrix. If an eigenvalue of* A *lies on the boundary of of the union* $\mathcal{S}_r = \bigcup_{i=1}^{n} D_i$ *of all the Gershgorin's disc, then it lies on the boundaries of each* D_i*,* $i = 1, \ldots, n$*.*

1.2.2 Housholder transformations and QR factorization

In this subsection we quickly review some notions useful in various parts of the whole Thesis.

Definition 1.2.5. (Householder matrix). *Let* $\mathbf{v} \in \mathbb{R}^n$ *a nonzero vector. A Householder matrix is an* $n \times n$ *matrix of the form*

$$H = 1 - \beta \mathbf{v}\mathbf{v}^T, \qquad \beta = \frac{2}{\mathbf{v}^T\mathbf{v}}.$$

The vector $\mathbf{v}$ in the previous definition is the *Householder vector*. Often, Householder matrices are also known as Householder *transformations* or *reflections* since, if a vector $\mathbf{x}$ is multiplied by a Householder matrix H, then it is reflected in the hyperplane given by the orthogonal complement of $\mathrm{span}\{\mathbf{v}\}$. Householder matrices are orthogonal and symmetric and are extensively used in numerical linear algebra for tridiagonalizing symmetric matrices. In particular, Householder triangularization is one of the principal methods for computing the QR decomposition of a matrix.

Theorem 1.2.4. (QR factorization). *Let* $A \in \mathbb{R}^{m\times n}$ *be a rectangular matrix. Then there exist an orthogonal matrix* $Q \in \mathbb{R}^{m\times m}$ *and an upper triangular matrix* $R \in \mathbb{R}^{m\times n}$ *so that*

$$A = QR. \tag{1.2.1}$$

The identity (1.2.1) is referred to as the *QR factorization* and it is the basis for a particular eigenvalue algorithm, the *QR algorithm*, which is an iterative method for approximating all the eigenvalues of (not too large) matrices; see [83] for more details. The QR factorization has a central role also in solving linear least squares problems. Just for completeness, we briefly recall that a least squares problem consists of solving an overdetermined ($m > n$) rectangular system of equations $A\mathbf{x} = \mathbf{b}$, with A of size $m \times n$, so that the 2-norm of the *residual* $\mathbf{r} = \mathbf{b} - A\mathbf{x}$ is minimized.

The QR factorization is strictly related to the well known Gram–Schmidt process for orthonormalising a set of vectors. Indeed, the Gram-Schmidt iteration is the basis of a numerical algorithm for computing QR factorizations. It is a process of making the columns of a matrix orthonormal through a sequence of matrix operations that can be interpreted as multiplication on the right by upper-triangular matrices; see [176] for further discussion on the Gram–Schmidt QR decomposition. The Householder algorithm, based on the use of Householder matrices, for determining the QR factorization of a matrix is numerically more stable than the Gram-Schmidt orthogonalization. The Householder algorithm is a process

of making a matrix triangular by a sequence of unitary matrix operations. Given a matrix A, the Householder method consists of applying a succession of Householder orthogonal matrices H_j on the left of A, so that the resulting matrix

$$H_n H_{n-1} \cdots H_2 H_1 A = R$$

is upper triangular. The product $Q = H_1 H_2 \cdots H_{n-1} H_n$ is an orthogonal matrix too, and therefore $A = QR$ is the QR factorization of the given matrix A.

1.2.3 Toeplitz and circulant matrices

A *Toeplitz* matrix T of order n is a matrix whose entries are constant along each diagonal, that is

$$t_{ij} = t_{i-j}, \quad i, j = 1, \ldots, n.$$

Toeplitz matrices are symmetric with respect to their antidiagonal, i.e., they belong to the class of the so-called *persymmetric* matrices. A matrix $B \in \mathbb{R}^{n \times n}$ is persymmetric if

$$Z_n B Z_n = B^T$$

where Z_n is the commonly named *flip* or n-by-n *exchange* matrix, given by

$$Z_n = \begin{bmatrix} O & & 1 \\ & \iddots & \\ 1 & & O \end{bmatrix}.$$

A *circulant matrix* is a special type of Toeplitz matrix in which each row vector is a cyclic permutation one to the right relatively to the row above. Specifically, a circulant matrix C of order n whose entries satisfy the relations

$$c_{ij} = c_{i-j}, \quad i, j = 1, \ldots, n$$

$$c_{k-n} = c_k, \quad k = 1, \ldots, n-1$$

and it takes the form

$$C = \begin{bmatrix} c_0 & c_{n-1} & \ldots & c_2 & c_1 \\ c_1 & c_0 & c_{n-1} & & c_2 \\ \vdots & c_1 & c_0 & \ddots & \vdots \\ c_{n-2} & & \ddots & \ddots & c_{n-1} \\ c_{n-1} & c_{n-2} & \ldots & c_1 & c_0 \end{bmatrix}. \tag{1.2.2}$$

Hence, a circulant matrix is fully specified by the vector $\mathbf{c}$ that appears as its first column. The remaining columns are cyclic permutations of $\mathbf{c}$ with an offset equal to the column index. The main property of circulant matrices is that they are diagonalized by a discrete Fourier transform, i.e., by the normalized Fourier matrix $\mathcal{F}$ defined by

$$\mathcal{F}_{k\ell} = \frac{1}{\sqrt{n}} \omega^{-(k-1)(\ell-1)}, \quad k, \ell = 1, \ldots, n$$

where $\omega := e^{\frac{2\pi \mathbf{i}}{n}}$ is the minimal phase nth root of unit and "i" denotes the imaginary unit.

Another property of a $n \times n$ circulant matrix is that its j-th normalized eigenvector is given by

$$\mathbf{v}_j = \frac{1}{\sqrt{n}}(1, \omega^j, (\omega^j)^2, \dots, (\omega^j)^{n-1})^T \tag{1.2.3}$$

where $\omega^j = \mathrm{e}^{\frac{2\pi \mathbf{i}}{n} j}$ is the jth power of ω. The corresponding eigenvalues are given by

$$\lambda_j = c_0 + c_1(\omega^j)^{n-1} + c_2(\omega^j)^{n-2} + \dots + c_1(\omega^j).$$

For a real symmetric circulant matrix, the real and imaginary parts of the eigenvectors are themselves eigenvectors and they lead to the discrete cosine transform (DCT) and the discrete sine transform (DST), respectively. See [157] where a Matlab toolbox for operating with Toeplitz and circulant matrices is presented.

1.3 Graphs and complex networks

In this Section we review some mathematical concepts fundamental in graph and complex network theories.

Relations between discrete quantities such as genes, people, proteins, or streets can be described by networks which consist of nodes, representing the entities of the analysed complex system, that are connected by edges describing the relations or interactions between these entities. Networks arise in many applications, including genetics, ecology, epidemiology, energy distribution, sociology, economy and telecommunication; see, e.g., [64, 67, 144] for discussions on networks and their applications. They are represented by graphs which are defined as follows.

Definition 1.3.1. (Graph). *Formally, a* graph *(or network)* $\mathcal{G}$ *is a pair of sets* (V, E) *where* $V = \{v_i\}_{i=1}^n$ *is a finite set whose elements are called* nodes *or* vertices *and* $E \subseteq V \times V = \{e_k\}_{k=1}^m$. *The elements of the set* E *are called* edges *or* arcs *and* $e_k = (i_k, j_k)$ *represents an edge from vertex* v_{i_k} *to vertex* v_{j_k}.

A graph is said to be *weighted* if each edge e_k has a weight w_k, *unweighted* if the weights are either 0 or 1. A large value of the weight $w_k > 0$ may indicate that edge e_k is important in the considered graph. For instance, in a road network, the weight w_k may be proportional to the amount of traffic on the road that is represented by the edge e_k.

In this thesis we won't consider complex networks, which are graphs satisfying particular properties related to their topology.

Two vertices are said to be *adjacent* if there exists an edge connecting them and an arc is *incident* to the nodes that connects. The number of edges connected to a node is called *degree* of the considered vertex. A graph is called *undirected* if $(i_k, j_k) \in E \iff (j_k, i_k) \in E$, and *directed* otherwise. In an undirected graph, the weights associated with these "two-way streets" edges are assumed to be the same while, as mentioned above, in unweighted graphs all weights are set to one. In a directed graph edges have instead an orientation and an edge starting from a node is an *out-edge* at that node, although an *in-edge* is an edge that arrives at that node.

Networks highlight direct connections between nodes but frequently there are also fundamental implicit connections in a network. If nodes are not explicitly linked by an edge, but there are several intermediate nodes in the way between them or there are nodes in

common, is it usual to assume that information can be passed from one to the other. The following notions are related to these implicit connections.

A *walk* of length k is a sequence of nodes $v_1, v_2, \ldots, v_k$ such that there is an edge between vertex v_i and vertex v_{i+1} for $i = 1, 2, \ldots, k-1$. Vertices and edges may be repeated in a walk. An oriented walk is a walk in which every edge of the sequence is oriented from vertex v_i to vertex v_{i+1}. A closed walk is a walk in which the last node of the sequence coincides with the first one. A *path* is a walk with all vertices distinct.

The notion of path is the basis of the concept of *connectivity* which is a significant property of a network. In fact, a graph is *connected* if there is a path connecting any two nodes. In an undirected network it is simple to identify connectivity: if it is not connected then there will be a part of the graph that is completely separated from the rest. For a directed graph the connectivity can be strong or weak: a directed network is *strongly* connected if there is an oriented path that connects any pair of nodes, *weakly* connected if the edges in the path do not follow the same orientation. If a network is not connected then it can be divided into components each of which is connected and therefore they are called *connected components*. An algorithm for identifying them will be presented in the following.

Given a matrix A, is it possible to associate to it a graph having the indices of A as nodes and oriented edges (i, j) for every entry $a_{ij} \neq 0$. Remembering that a matrix A is irreducible if does not exist a permutation matrix P such that the matrix PAP^T is block triangular (see Definition 1.2.2), the following theorem emphasizes that there is a correlation between the irreducibility of a matrix and the connectivity of the associated graph.

Theorem 1.3.1. *A matrix A is irreducible if and only if the directed graph, related to it, is strongly connected.*

In this thesis we will consider *simple graphs* that are undirected networks without self-loops (edges connecting nodes to themselves) and multiple edges (arcs between the same pair of vertices).

There are some matrices devoted to represent a graph. The most commonly used is the *adjacency matrix*, defined as follows.

Definition 1.3.2. (Adjacency matrix). *The adjacency matrix A of an undirected weighted simple graph $\mathcal{G}$ with n nodes is the $n \times n$ matrix $A = \{a_{i,j}\}_{i,j=1}^{n}$, where*

$$a_{i,j} = \begin{cases} w_k, & \text{if there is an edge } e_k \text{ between the nodes } v_i \text{ and } v_j \text{ with weight } w_k, \\ 0, & \text{otherwise}. \end{cases}$$

Since $\mathcal{G}$ is undirected and the weights associated with each direction of an edge are the same, the matrix A is symmetric. The diagonal entries of A are zeroes since $\mathcal{G}$ does not contain any self-loop. The largest possible number of edges of an undirected graph with n nodes without self-loops is $n^2 - n$, but typically the actual number m of edges of such graphs that arise in applications, is much smaller. The adjacency matrix A, therefore, is generally very sparse. If $\mathcal{G}$ is unweighted, then the entries of the adjacency matrix are defined as

$$a_{i,j} = \begin{cases} 1, & \text{if } (i,j) \in E, \\ 0, & \text{otherwise}. \end{cases}$$

Then, it is easy to see that in an undirected network the ℓth row or column of A has exactly k_ℓ entries. The number k_ℓ is the degree of node ℓ and represents the number of nearest

neighbors that ℓ has. From the definition of adjacency matrix it is easy to see that for $\ell \geq 1$, the (i, j) entry of A^ℓ gives the number of walks of length ℓ starting at node i and ending at node j.

A matrix that can be used together with the adjacency matrix to construct the *graph Laplacian*, that will be defined in Chapter 2, is the *degree matrix* that is a diagonal matrix which contains information about the degree of each node.

Definition 1.3.3. (Degree matrix). *Given a graph $\mathcal{G}$ with n nodes, the degree matrix D of $\mathcal{G}$ is a $n \times n$ diagonal matrix whose entries are defined as*

$$d_{i,j} = \begin{cases} d_i, & \text{if } i = j, \\ 0, & \text{otherwise} \end{cases}.$$

In particular, if $\mathcal{G}$ is a weighted graph, d_i equals the sum of the weights of the edges starting from node i in the undirected network defined by A, that is $d_i = \sum_{j=1}^{n} a_{ij}$. In the case of an unweighted graph, d_i is the number of nodes connected to it, that is the number of times an edge is incident to that vertex.

In an undirected graph, this means that each loop increases the degree of a vertex by two.Vertex degrees are more complicated in directed networks. In a directed graph, the term degree may refer either to *in-degree* or *out-degree* that is, the number of nodes that can reach one node or that can be reached from that node, respectively. Bearing in mind that the adjacency matrix of a directed network has element $a_{i,j} = 1$ if there is an edge from node i to node j, in- and out-degrees can be written, respectively, as

$$d_i^{\text{in}} = \sum_{j=1}^{n} a_{j,i}, \qquad d_i^{\text{out}} = \sum_{j=1}^{n} a_{i,j}.$$

The degree is used for defining the *degree centrality*. The notion of *centrality* of a node first arose in the context of social sciences and is used for determining the most "important" nodes in a network. The degree of a node can be used indeed to characterize and measure the importance of a node in terms of connection with the rest of the network. This *centrality measure* quantifies the ability of a node to communicate directly with others. Despite that, this metric does not give global information on the graph, since it only counts the number of neighbors of each node and, hence, it fails in taking into account the importance of the nodes connected to the considered one, e.g., how well-connected they are in the network.

Besides the degree, there are other important quantities that describe global properties of a given graph, such as the importance of a particular node within the network, or the ease of traveling from one node to another. Since we are not interested in the computation of centrality measures, the interested reader can refer to [64, 67].

By taking a different point of view of a graph, one can come up with another way of representing a network in matrix form in terms of the *incidence matrix* whose elements indicate if pairs node-edge are incident or not. The columns of this matrix are labeled by the arcs and the rows are labeled by the nodes.

Definition 1.3.4. (Incidence matrix). *Let $\mathcal{G}$ be a simple graph with $|V| = n$ nodes and $|E| = m$ edges. The incidence matrix B of $\mathcal{G}$ is the $n \times m$ matrix such that*

$$b_{i,j} = \begin{cases} 1, & \text{if node } v_i \text{ and edge } e_j \text{ are incident}, \\ 0, & \text{otherwise}. \end{cases}$$

B can be considered as a *signless/unsigned incidence matrix* to distinguish it from the incidence matrix of a directed graph whose entries are such that $b_{i,j} = -1$ if the edge e_j leaves vertex v_i, 1 if it enters vertex v_i and 0 otherwise.

The following type of graphs are used in Chapter 2 for describing the seriation problem and they are in particular analyzed in Chapter 3.

The membership of nodes in groups can be represented by special graphs, namely the *bipartite graphs* also called "two-mode" networks in the sociology literature. In a bipartite network there are two kind of vertices, one representing the original nodes and the other type representing the groups to which they belong with edges running only between vertices of unlike kinds. In the case of affiliation networks, for example, the two types of vertex represent people and the groups they belong to. In the case of a metabolic network the different types of vertices represent metabolites and metabolic reactions, with edges joining each metabolite to the reactions in which it participates. See [144] for further examples. Formally, bipartite networks can be defined as follows.

Definition 1.3.5. (Bipartite graph). *A graph G is a bipartite graph if its vertices can be divided into two disjoint sets U and V containing n and m nodes, respectively, such that every edge connects a node in U to one in V.*

The following definition regards the matrix representation of a bipartite graph.

Definition 1.3.6. (Adjacency matrix of a bipartite graph). *The adjacency matrix of a bipartite graph whose parts contains n and m nodes, respectively, is of the form*

$$A_B = P \begin{bmatrix} 0_n & A \\ C & 0_m \end{bmatrix} P^T, \tag{1.3.1}$$

for a permutation matrix P, where 0_n and 0_m are null matrices of order n and m respectively. In an undirected network $C = A^T$ and $A \in \mathbb{R}^{n \times m}$ describes the connections in the graph.

There are many networks in real applications that are exactly or nearly bipartite; see Chapter 3 for additional details.

1.4 A brief overview on permutations

In this Section we review a few notions of Combinatorics focusing, in particular, on permutations. Specifically, we will consider only the concepts that will be useful in the rest of the Thesis. The reader interested in an extensive analysis of elementary combinatorics, can refer to [27].

The following definition is only one of the possible definitions of permutations. For example, in algebra and particularly in group theory, a permutation of a set S is defined as a *bijection* from S to itself.

Definition 1.4.1. (Permutation). *A permutation is a linear ordering of the elements of a set $\{1, 2, \ldots, n\}$.*

When one wants to stress the fact that the elements to be arranged are n, the act of organizing the members of the considered set into a sequence or order is called n-permutation.

In other words, a permutation lists all the elements of a set so that each element is listed exactly once. For example if $n = 3$, all the possible permutations written as the following tuples, i.e., finite ordered sequences of elements, which represent all the possible orderings of the three-element set $\{1, 2, 3\}$

$$(1, 2, 3), \quad (1, 3, 2), \quad (2, 1, 3), \quad (2, 3, 1), \quad (3, 1, 2), \quad \text{and} \quad (3, 2, 1).$$

Note that permutations differ from *combinations*, which are collections of objects of a set regardless of order. Hence, one could say that a permutation is an *ordered* combination.

The following simple statement is probably the best-known fact about permutations; see [19].

Theorem 1.4.1. *The number of permutations of n distinct objects is $n!$, i.e., the product of all positive integers less than or equal to n.*

Permutations are the elements π of the *symmetric group* $\mathcal{S}_n$ consisting of all bijections from the set $\{1, 2, \ldots, n\}$ to itself, using composition as multiplication. Hence, for example, the permutation $\pi\sigma$ is the bijection obtained by first applying σ and then π.

The analysis of permutations of finite sets is a fundamental topic in the fields of combinatorics and group theory; however, permutations appear and are studied in almost every branch of mathematics and in various other fields of science. For example, in computer science, they are used for analyzing sorting algorithms, in quantum physics are studied for describing states of particles, in biology for describing RNA sequences and in telecommunications are used in the allocation of telephone numbers to subscribers around the world.

In certain situations, additional restrictions may be imposed. In these cases, the problem is related to permutations with constraints. Generally, the restriction consists in the fact that only a small number of the total objects need to be ordered. Other common types of constraints include restricting the type of objects that can be adjacent to one another, or changing the ordering from linear to another topology (e.g. a round table instead of a line, or a key-chain instead of a ring). Restrictions to few objects is equivalent to consider n given distinct objects and searching for how many ways there are to place $k < n$ of them into an ordering. In this case we consider the k-permutations, or *partial* permutations, that are the ordered arrangements of k distinct elements selected from the given set with n objects. And the total number of partial orderings is given by the following result.

Theorem 1.4.2. *Given n distinct objects, the number of possible different ways to place k of them into an ordering is denoted here by P_k^n and its value is given by the product of k factors as follows*

$$n(n-1)(n-2)\cdots(n-k+1) = \frac{n!}{(n-k)!}.$$

The product is known as the *Pochhammer symbol* also called *falling* or *descending* factorial.

If $\pi \in \mathcal{S}_n$ is a permutation of the set $\{1, 2, \ldots, n\}$, there are different notations for representing it in a compact way. The *two-line* notation is an array consisting in two rows, the first one lists the elements of the considered set while the second row contains for each element the image below it. For example, if $\pi \in \mathcal{S}_5$ is a permutation given by

$$\pi(1) = 2, \quad \pi(2) = 3, \quad \pi(3) = 1, \quad \pi(4) = 5, \quad \pi(5) = 4,$$

then, its two-line form is

$$\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 1 & 5 & 4 \end{pmatrix}.$$

Since the top line is fixed by considering the natural order or the elements of $\{1, 2, \ldots, n\}$, one can drop the first row of the two-line notation and write the permutation π in *one-line* notation as

$$\pi = (2 \quad 3 \quad 1 \quad 5 \quad 4).$$

For this kind of notation, the parentheses are generally omitted and used, instead, for the *cycle* notation that expresses the permutation as a composition of cycles. The cycle $(i, j, k, \ldots, l)$ means that π sends i to j , j to k, ..., and l back to i. The process is then iterated, selecting an element not in the cycle containing i, until all the elements of the considered set have been used. Hence, cycle notation describes the effect of repeatedly applying the permutation on the elements of the set. Then, the previous example in cycle notation becomes

$$\pi = (1, 2, 3)(4, 5). \tag{1.4.1}$$

The cycle notation allows us to represent the so-called *cyclic permutations*. A cyclic permutation (or cycle) is a permutation of the elements of a given set X which maps the elements of a subset U of X to each other in a cyclic way and leaves all the remaining elements of the set in question unchanged. If the subset U contains k elements, the cyclic permutation is called k-cycle. For instance, given $X = \{1, 2, 3, 4, 5\}$. the permutation $(1, 3, 2, 5, 4)$ that sends 1 to 3, 3 to 2, 2 to 5, 5 to 4 and 4 to 1 is a 5-cycle. The permutation $(1, 3, 2, 5)$, that fixes the element 4, is a 4-cycle. Conversely, the permutation (1.4.1) is not a cyclic permutation since it separately permutes the triple $\{1, 2, 3\}$ form the pair $\{2, 4\}$.

A permutation of the elements of the set $\{1, 2, \ldots, n\}$ can also be represented as an $n \times n$ matrix. Hence, if $\pi \in \mathcal{S}_n$ we can associated to this permutation the matrix $X(\pi)$ whose entries $x_{i,j}$ are

$$x_{i,j} = \begin{cases} 1, & \text{if} \quad i = \pi(j), \\ 0, & \text{otherwise.} \end{cases}$$

The resulting matrix is a permutation matrix since it contains only zeros and ones with exactly a unique entry 1 in every column and in each row; see 1.2.1.

1.5 Computational complexity

In this Section we concisely consider some notions of the *computational complexity theory*, a subfield of theoretical computer science.

The primary goal of computational complexity theory is to classify and compare problems which have similar complexity with respect to a specific computation model and complexity measure. A computational problem may be solvable by a mechanical application of mathematical steps, such as an algorithm, and it is regarded as inherently difficult if its solution requires significant resources, no matter which algorithm is used. Discussion of the complexity of a computational problem implicitly refers to the complexity of an algorithm for solving that problem and to the measure of complexity for evaluating the algorithm's performance.

Complexity theory attempts to formalise the distinction of computational problems depending on their difficulty and to make it precise by introducing mathematical models of

computation to study the problems and by quantifying their computational complexity, i.e., the amount of resources needed to solve them, such as time and storage.

One of the most commonly used mathematical models in complexity theory is the *Turing machine* developed by the logician Alan Turing in 1935. Many types of Turing machines, such as deterministic, probabilistic, non-deterministic and quantum Turing machines, are used to define complexity classes and hence for classifying the computational problems according to their inherent difficulty.

Central to the development of computational complexity theory is the notion of *decision problem*. Such a problem can be formalised as a yes-no question of the input values and it corresponds to decide if the input belongs or not to a given set X. For instance, the problem of deciding whether a given natural number is prime is a simple example of a decision problem.

Computational problems can be classified according to their *decidability*. The resources needed to realise an algorithm in order to decide an instance of a problem are typically measured in terms of the number of processor cycles (i.e. elementary computational steps), the amount of memory space (i.e. storage for auxiliary computations) and the number of processors used in parallel computing required to give a solution. The methods of complexity theory can be then useful not only in deciding how we can efficiently use such supplies, but also for distinguishing which effectively decidable problems can be solved with efficient decision methods. In this regard, it is customary to theoretically distinguish between the class of *feasibly decidable* problems – i.e. those which can be solved in practice by an efficient algorithm – and the class of *intractable* or *undecidable* problems, i.e. those which lack an effective procedure that always leads to a correct yes-or-no answer and they may thus be regarded as intrinsically difficult to solve.

Many of the problems studied in complexity theory are decision variants of optimization problems.

1.5.1 Complexity classes

A complexity class can be defined as the set of decision problems for which there exists a decision procedure with a given running time or running space complexity. Indeed computational problems can be classified in terms of how much time or storage space is needed to solve them as a function of the length n of the input. This first distinction allows the definition of the time-complexity and space-complexity classes denoted by TIME(f(n)) and SPACE(f(n)) respectively. The first one denotes the class of problems with time complexity $f(n)$, while the second class includes all the decision problems that can be solved by a deterministic Turing machine using an amount of memory space of the order of $f(n)$.

Considering the time-complexity class we briefly consider some of the classifications of computational problems based on time bounds.

A feasibly decidable computational problem can be solved by a deterministic Turing machine in *polynomial time*, i.e. in a number of steps which is proportional to a polynomial function of the size of the input. The class of problems with this property is known as *class P*, or *polynomial time* complexity class. Hence, the class P includes all the problems for which there exists a decision algorithm whose time complexity is of polynomial order of growth. The importance of such a class derives from the fact that it includes many "simple" problems such as, for instance, the computation of the greatest common divisor, finding a maximum matching and the problem of determining if a number is prime. Although the

class P is usually associated with the class of computationally tractable problems, it also includes many problems which cannot be solved in practice by computers due to the high degree of the polynomial; see [24] for further details.

The P class can be formally shown to be distinct from other classes such as the *class EXP*, or *exponential time*, which includes all the decision problems that have exponential runtime, i.e., that are solvable by a deterministic Turing machine in $O(2^p)$ time, where $p = p(n)$ is a polynomial function of the length n of the input.

In this thesis some of the considered problem are included in another important complexity class called *class NP* or *non-deterministic polynomial time*. This class includes problems that are harder to solve than problem of P and hence, it contains highly intractable problems. The NP class is a natural extension of class P obtained by replacing deterministic algorithms with non-deterministic ones. In fact, it consists of those computational decision problems which can be solvable by a non-deterministic Turing machine in polynomial time, i.e. in a number of steps which is a polynomial function of the size of the input. Examples of problems belonging to the NP complexity class, arise in numerous areas such as graph theory, network design, algebra and number theory, game theory, logic, automata and language theory.

It is easy to see that the complexity class P, containing all the problems deterministically solvable in polynomial time, is contained in the NP class. A famous conjecture, namely the *P versus NP problem*, states that P $\subsetneq$ NP, i.e. P is properly contained in the NP class. Basically this problem asks whether every problem whose solution can be verified in polynomial time, can also be solved in polynomial time. The P versus NP question, raised in mathematical logic and theoretical computer science during the middle of the twentieth century, is still unsolved and proving or disproving the non-coincidence P$\neq$NP of these complexity classes remain an important open problem in contemporary complexity theory and it is one of the seven Millennium Prize Problems. See [166] for a survey on the history of the topic.

The NP class contains more problems than the P one, the hardest of which are called *NP-complete* problems. An algorithm solving such a problem in polynomial time is also able to solve any other NP problem in polynomial time. Formally, a decision problem is NP-complete if it is NP and every problem in the NP class can be reduced to the considered one, in polynomial time.

The most important P versus NP problem, asks whether polynomial time algorithms exist for solving NP-complete, and by corollary, all NP problems. All known algorithms for NP-complete problems require time that is *super-polynomial* in the input size, and it is unknown whether there are any faster algorithms. A significant subclass of NP-complete problems includes polynomial time approximation algorithms which search for a solution that is within a certain constant factor of optimality.

Other computational problems are classified as belonging to the *NP-hard* complexity class. NP-hard problems do not have to be elements of the NP class and indeed, they may not even be decidable. For this kind of problems the most efficient known decision algorithm has exponential time complexity in the worst case. Informally NP-hard (non-deterministic polynomial time hard) problems are at least as hard as the hardest problems in the NP complexity class. A more precise definition is that decision problem H is NP-hard when for every NP problem L, there is a polynomial-time reduction from L to H [178].

An example of an NP-hard problem is the subset sum problem, that consists of the decision if any non-empty subset of a given set of integers adds up to zero. Another example is given by the optimization problem of finding the least-cost cyclic route through all nodes

of a weighted graph. This is commonly known as the *traveling salesman problem* and it will be described more in details in Chapter 2 in relation to the seriation problem.

Many classical results and important open questions in complexity theory concern the inclusion relationships which hold among the classes; see [151] for a complete description of them as well as for a report on other time-complexity classes and all the space-complexity classes. In Figure 1.1 the inclusion relationships among the considered major complexity classes are depicted under the hypothesis of the non-coincidence and coincidence of the P and NP classes in the P versus NP issue.

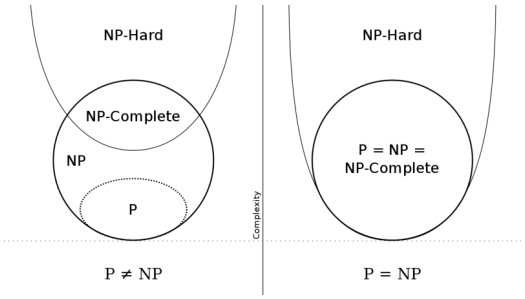


Figure 1.1: Euler diagram for P, NP, NP-complete, and NP-hard complexity classes of computational problems. On the left the representation valid under the assumption that P≠NP, while the right side is valid under the assumption that P=NP.

Chapter 2

The seriation problem

2.1 The seriation problem: an overview on the historical context and applications

Seriation is a ubiquitous, fundamental combinatorial ordering problem, asking to find the best enumeration order of a set of units, according to a given correlation function, so that elements with higher similarity are close to each other in the resulting sequence. The desired order can be characteristic of the data, a chronological order, a gradient or any sequential structure of the data. As it will be made clear hereinafter, we will state the seriation problem from the mathematical point of view by considering it as the arrangement of units in a sequence according to a "similarity" function and we will focus on chronological seriation in archaeology.

The concept of seriation has been formulated in many different ways and the seriation problem, along with seriation methods and algorithms, emerges and finds application, as explained more in details below, in a wide range of contexts spanning from archaeology, anthropology, to genomics and DNA sequencing, and, within mathematics, from complexity theory to combinatorial optimisation, graph theory and operational research.

In this thesis, we will use the archaeological setting as a metaphor for the seriation problem considering also the fact that seriation actually arose for the first time in the context of archaeological studies, where it is typically formulated as the problem of dating excavation sites (e.g., deposits, such as tombs in a necropolis) on the basis of the finds discovered inside them (e.g., assemblage of artefacts in the deposits, such as grave goods in each tomb) according to how these contents are related, and of determining their relative chronology i.e., a dating which indicates if a given site is chronologically preceding or subsequent to another. Archaeological finds are in turn classified according to their manufacturing style, technical characteristics or typology, with the hypothesis that different types were produced and "fashionable" only for a limited period of time. In general, relative chronologies are devoid of a direction, in the sense that the units are placed in a sequence which can be read in both directions. Relative dating methods, such as seriation, are vital when absolute dating methods consisting of deciphering inscriptions or applying physical or chemical techniques such as carbon dating, cannot be applied.

The first systematic formalisation of the seriation problem in archaeology was made by an English Egyptologist, Sir W. M. Flinders Petrie, in 1899 [152]; his work on the prehistoric Egyptian necropoleis in Naqada and Abydos, in the Nile area, represents the

first example of a temporal ordering of burial places obtained from a combinatorial analysis of archaeological data. Since common modern dating techniques such as stratigraphy or radio carboning, were not available at that time, Petrie designed a relative dating procedure to sequence the graves. Specifically, he examined about 900 burial sites and classified the potteries found in the tombs according to their manufacturing style assigning them sequence dates. Hence, he tried to reorder the artifacts according to their similarity, with the idea that similar potteries would belong to ages closer in time under the assumption that different objects continuously come into and go out of fashion. Then, the order of the tombs was obtained by linking each tomb to the corresponding sequenced pottery element.

Observations and methods presented by Petrie are recognized for pioneering the idea of sequencing objects. Indeed, Petrie's work influenced several prominent American anthropologists and archaeologists and his ideas have been pivotal in establishing a whole line of research, with a reprise of interest from the 1950s with the works of Brainerd [25], Robinson [158], who also proposed a practical method for the seriation problem solution, and in particular by Kendall who pioneered the dialogue between archaeologists and statisticians and wrote several papers on the research of mathematical properties of the matrix analysis used in archaeology; see [107–109]. See [153] for a review on the seriation problem in archaeology with a mathematical perspective. This paper has inspired the work on seriation explained in this thesis. A comprehensive description of the development of seriation in archeology is presented by Ihm in 2005 [101]. Another review on seriation is given in [146], where application of seriation in stratigraphy is discussed. An extensive survey of the problem from an archaeologist's point of view appears in [128], while overviews on the application of mathematics to archaeology can be found in [11, 92, 110, 183].

As mentioned before, seriation has several applications in a wide range of different fields. Although it was originally introduced for applications arising in archaeology, seriation can also be used in paleontology for ordering the excavation sites on the basis of mammal species whose remains are found in the sites [130]. Furthermore, seriation can be used for data visualization and data combinatorial analysis as a method for studying the relevant data sets in which is important the arrangement of a collection of objects [28, 100], with applications in biology and bioinformatics [126]. In particular, in the bioinformatics setting, seriation is used for large-scale expression pattern discovery in SAGE (Serial Analysis of Gene Expression) data [141] and for identifying coherent local clusters with global patterns in microarray gene expression profiles [174]. In this area of interest another significant application of seriation, in use since the early 1990s, is in genomics and DNA sequencing [63, 92, 136]. In such field of study, seriation is applied in the construction of physical maps by hybridisation in the context of genome sequencing [84]. Given a DNA sequence, a set of so-called "clones" (each clone being a copy of a second DNA sequence containing a single, specific fragment of the original DNA sequence), and a set of so-called "probes" (each probe being a marker of a specific site on the original DNA sequence), a series of "hybridisation" experiments is run to determine whether a certain clone contains a certain probe, namely if the fragment carried by the clone contains the site indicated by the probe. Clones are chosen to carry overlapping fragments of the original DNA sequence. Through multiple hybridisation experiments, an hybridisation matrix is built, namely a (0,1)-matrix indicating which are the probes included by each clone. Then, the seriation problem is that of constructing a physical map of the original DNA sequence by ordering the probes, namely the sites on the sequence.

In sociology, and more specifically in sociometry, seriation is used to rearrange data

coming from sociometric tests for describing and evaluating social status and structure and, hence, for finding group assemblage in sociograms [75]. In machine learning, seriation can be adopted for determining the possible number of clusters in a set of objects or their cluster tendency [87], with also applications to text data mining, for example, in order to derive high-quality information from online newsgroup articles [55]. Techniques related to seriation are also popular in several other fields: for example, in ecology where seriation techniques are used, under the name *ordination*, for the arrangement or "ordering" of species and/or sample units along gradients [2, 3]. Further applications of seriation include cartography and graphics, psychology and psychometry for the study of confusion data [29].

In the mathematical discipline, uses of seriation are in the reordering of sparse matrices to reduce their so-called envelope size [12], in recognition of interval graphs in graph theory and network analysis [21, 116], in matrix reordering [131] and in ranking for ordering a set of items given pairwise comparisons between these objects [73]. For an application of seriation methods in operational research and optimization for rearranging data arrays see [53]. A comprehensive historical overview of the development of seriation techniques and a more exhaustive and complete list of applications can be found in a review article by Liiv [123].

Given this variety of applications, some software packages have been developed in the past to manipulate seriation data. Some of these packages have not undergone regular maintenance, and do not seem to be easily usable on modern computers, like the Bonn Archaeological Software Package (BASP) (`http://www.uni-koeln.de/~al001/`) that includes functions for seriation, clustering, correspondance analysis and mapping tools for archaeologists. Another seriation software for dating archaeological artifacts or assemblages of objects, called *OptiPath*, is capable of performing a wide range of seriation techniques including occurrence seriation, frequency seriation and shortest path seriation. Continually in development and testing, it runs under Microsoft Windows and is freely available for installation at `http://terevaka.net/optipath/publish.htm`. A package called "seriation", providing an infrastructure for seriation, is implemented in the open source statistical software R [85] and uses different algorithms depending on the seriation measure chosen to model the seriation problem. A software specifically designed for the seriation problem in bioinformatics has been developed by Caraux and Pinloche [32]. In this bioinformatics software package, called *PermutMatrix* and available for free download, a data analysis of gene expression profiles is performed using various seriation methods. In [124], Liiv et al. introduce a web-based tool for exploratory visual analytics, called Visual Matrix Explorer (VME), that enables dynamic evaluation and visualization of matrices, and allows the linking of different seriation results using some explicative examples coming from psychology and paleontology. A seriation method, based on an algorithm described by Brower and Kile in [26], is integrated in the statistical analysis software, called PAST (PAleontological STatistics) presented in [86] and available for free download at `https://folk.uio.no/ohammer/past/`. PAST integrates functions developed for executing a range of standard numerical analysis computation used in quantitative paleontology and ecology, and the seriation technique included in it, is applied to reordering taxa (species) according to given samples.

In this Chapter, we present a Matlab implementation of a spectral method for the solution of the seriation problem which appeared in [6]. This technique is based on the use of an eigenvector, called *Fiedler vector* and associated to the eigenvalue, called *Fiedler value* of the Laplacian matrix associated to the problem and it describes the results in terms of a

particular compact data structure called PQ-tree.

We further develop some numerical aspects of the algorithm in [6], concerning the detection of equal components in the Fiedler vector and the computation of the eigensystem of the Laplacian associated to a large scale problem. We also provide a parallel version of the method. The package, named the `PQser` toolbox, described in Section 2.3 also defines a data structure to store a PQ-tree and provides the Matlab functions to manipulate and visualize it. The toolbox is compatible with Octave, except for the parallel implementation. The results of our implementation of the spectral algorithm are then compared in Subsection 2.4.2 with those provided by a specific function in the R package mentioned above and available at `http://cran.r-project.org/web/packages/seriation/`. Finally, in Section 2.5 we discuss the implications of the presence of a multiple Fiedler value, an issue which has been disregarded up to now, and we illustrate the discussion with some numerical experiments considering a graphic method for finding the admissible permutations in some particular cases involving graphs having double Fiedler value. In fact as it will be made clear in the following, we will explain how the seriation problem can be expressed in terms of finding the best, or an approximate, ordering of a set of units whose interrelationship can be defined in terms of a *bipartite graph*.

2.1.1 The data matrix

In all the applications, seriation data are usually given in terms of a matrix, called *data matrix*, whose rows or columns (or both) represent the elements to be ordered. Considering the archaeological application, the rows of the data matrix are the archaeological units (e.g., the sites, deposits, tombs) and the columns represent the types of the archaeological finds detected inside the units. Each unit is characterized by the presence of certain artifacts, which are systematized in types. See [153] for an explanation on how to practically realize this correlation and in what manner is it possible to construct the $m \times n$ data matrix representing units/types.

In general, authors refer to the data matrix as either *incidence matrix* or *abundance matrix*, depending on the archaeological data representation. In the first case, the data are reported by using a binary representation, i.e., an element in position (i, j) is equal to 1 if type j is present in the unit i, and 0 otherwise. In the second case, the data matrix reports the number of objects belonging to a certain type in a given unit, or its percentage. So each entry of an abundance data matrix represents the (relative) frequency or quantity of an artifact type in the considered site. See [101] for a comparison between incidence and abundance matrices in archeology.

In this thesis, we follow the typical terminology used in *complex networks theory* and we refer to a binary representation of seriation data as an *adjacency matrix*; see Section 2.2 for further details. An example of an adjacency matrix illustrating seriation data, is given in Table 2.1 that represents archaeological data acquired from female sepulchers of the first Iron Age rediscovered in the Germanic site of Bornholm; see [153] and the references therein.

Under the hypothesis that the sites (units) have been assembled in a particular moment or in a short temporal interval, fixed a certain typology, the information obtained from the study of the archaeological units are reported in the data matrix. And therefore, if the data matrix represents types of found objects as columns, and the locations in which they are found (graves, pits, etc.) as rows, we can find a chronological order for the locations by

Table 2.1: Adjacency matrix for archaeological data originated from female burials at the Germanic Bornholm site, Denmark [104, 105]. The rows report the names of the tombs, the columns the identification codes of the found *fibulae*.

	G3	F27	S1	F26	N2	F24	P6	F25	P5	P4	N1	F23
Mollebakken 2	1	1	1	1	0	0	0	0	0	0	0	0
Kobbea 11	0	1	1	0	1	1	0	0	0	0	0	0
Mollebakken 1	1	1	0	1	1	0	1	1	0	0	0	0
Levka 2	0	1	1	0	1	0	0	1	1	0	0	0
Grodbygard 324	0	0	0	0	1	1	0	0	0	1	0	0
Melsted 8	0	0	1	1	0	0	1	1	0	1	0	0
Bokul 7	0	0	0	0	0	0	1	1	0	0	1	0
Heslergaard 11	0	0	0	0	0	0	0	1	0	1	0	0
Bokul 12	0	0	0	0	0	0	0	1	1	0	0	1
Slamrebjerg 142	0	0	0	0	0	0	0	0	0	1	0	1
Nexo 6	0	0	0	0	0	0	0	0	0	1	1	1

assuming that the types were fabricated, or were "in vogue" only for a short time, under the hypothesis that different objects continuously come into and go out of fashion. In light of this assumption, the purpose of determining a relative chronology for the excavation sites results in obtaining an ordering of the rows and columns of the data matrix that places the nonzero entries close to the main diagonal of the data matrix.

2.2 The seriation problem in terms of graph theory

In the first part of this Section, some definitions and results of *spectral graph theory* are given. Subsequently, the rest of the section aims at delineating how the seriation problem can be described using the terminology and definitions typical of graph theory; see Section 1.3.

The adjacency and incidence matrices defined in Chapter 1 (see Section 1.3) are not the only useful algebraic representations of a graph, especially in the study of spectral graph theory, that involves the investigation of the relationships between the topological properties of a graph and the algebraic properties of the spectra of certain matrices associated with it. This duality between topology and spectral domain has been widely studied in the field of mathematics called *algebraic graph theory*. Several books and surveys on the topic have already appeared, for example by Cvetkovic̀ et al. [45, 46], Biggs [17] and Godsil and Royle [81]. Most of the early work involved the spectrum of adjacency matrices, although we are more interested in the algebraic properties of the spectrum of the *Laplacian matrix*.

The *graph Laplacian* can be defined by means of either the adjacency matrix or the incidence matrix.

Definition 2.2.1. (Laplacian matrix). *The (unnormalized) graph Laplacian of a symmetric matrix $A \in \mathbb{R}^{n\times n}$ that represents a simple graph G formed by n nodes, is the symmetric positive semidefinite matrix*

$$L = D - A,$$

where $D = \mathrm{diag}(d_1, \ldots, d_n)$ is the degree matrix.

Explicitly, the entries of L are given by

$$\ell_{i,j} = \begin{cases} d_i, & \text{if } i = j, \\ -1, & \text{if } i \neq j \text{ and there is an edge between } v_i \text{ and } v_j \\ 0, & \text{otherwise.} \end{cases}$$

Alternatively, we can write

$$\ell_{i,j} = \delta_{i,j} d_i - a_{i,j},$$

where $\delta_{i,j}$ is the Kronecker delta.

The graph Laplacian can also be written in terms of the incidence matrix B (see Definition 1.3.4) as

$$L = B^T B$$

and it can be shown that L is indeed a discrete analogue of the continuous Laplacian operator $\Delta = \nabla^2$, see [67] for some details on this correlation. For the correspondence between the operator Δ and the Laplacian matrix of a graph which discretizes the region where the Laplace equation is studied, see also [47].

The Laplacian matrix is a very useful tool for analysing a graph. Its spectrum and, in particular, its eigenvectors can reveal significant properties of the considered network. Note that $L = D - A$ is symmetric since D and A are both symmetric; the symmetry $L = L^T$ also follows from the other definition of the graph Laplacian, $L = B^T B$, in terms of the incidence matrix B. Then, the spectrum of L is real.

Setting $\mathbf{e} = [1, \ldots, 1]^T \in \mathbb{R}^n$, it is immediate to observe that $L\mathbf{e} = \mathbf{0}$, where $\mathbf{0} \in \mathbb{R}^n$ is the zero vector. Hence, 0 is an eigenvalue of the graph Laplacian with eigenvector $\mathbf{e}$. The Gershgorin discs of L are of the form $\Delta_i = z : \|z - d_i\| \leq d_i$, where d_i is the degree of node i. Then, the Gershgorin's circle theorem 1.1.7 implies that all the eigenvalues are non-negative and it can also be shown that they lie in the interval $[0, 2d_{max}]$ where d_{max} is the maximal degree of a node in the associated network; see [67]. Then, we can order the real, non-negative eigenvalues of the Laplacian L as $\lambda_1 = 0 \leq \lambda_2 \leq \cdots \leq \lambda_n$, with corresponding eigenvectors $\mathbf{v}_1 = \mathbf{e}, \mathbf{v}_2, \ldots, \mathbf{v}_n$. The smallest eigenvalue of L with associated eigenvector orthogonal to $\mathbf{e}$ is called the *Fiedler value* of the graph described by F. The corresponding eigenvector is the so-called *Fiedler vector*.

Alternatively, the Fiedler value may be defined using the Courant-Fisher principle (see Theorem 1.1.5), by

$$\min_{\mathbf{x}^T\mathbf{e}=0,\ \mathbf{x}^T\mathbf{x}=1} \mathbf{x}^T L \mathbf{x}.$$

Then, a Fiedler vector is any vector $\mathbf{x}$ that achieves the minimum.

Disconnectivity of a network is related to the reducibility of its adjacency matrix and expresses that no communication is possible between two nodes in a different component that is a connected subgraph contained in the given graph; see Section 1.3. The Fiedler value, is also known as *algebraic connectivity* and denoted by $a(G)$. It gives information on the connectivity of the considered graph and so its importance is due to the fact that it is a good parameter to measure how well a graph is connected, as expressed by the following well-known theorem.

Theorem 2.2.1. *Let $\lambda_1 = 0 \leq \lambda_2 \leq \cdots \leq \lambda_n$ be the eigenvalues of the Laplacian matrix L of a graph G with n nodes. Then G is connected if and only if $\lambda_2 > 0$.*

Proof. See [179]. □

This theorem is a consequence of the Perron-Frobenius theorem 1.2.2 for a nonnegative, irreducible matrix. So, if the considered adjacency matrix is irreducible, that is, if the graph is connected, then the Fiedler vector corresponds to the first non-zero eigenvalue of the Laplacian matrix. Moreover, it can be proved (see [179]) that the multiplicity of the smallest eigenvalue $\lambda_1 = 0$ of the Laplacian matrix is equal to the number of connected components of the graph.

The spectrum of L can also be used for enumerating *spanning trees* of a graph according to one of the oldest theorems in Graph Theory, Theorem 2.2.2 , whose proof can be found in [17]. A spanning tree of an undirected graph G is a subgraph and in particular a tree (undirected graph in which any two vertices are connected by exactly one path) which includes all the vertices of G with the minimum possible number of edges i.e. only the edges that are necessary to connect all the nodes with only one walk. The following theorem states that the determinant of any cofactor of the graph Laplacian is equal to the number of spanning trees in the considered graph.

Theorem 2.2.2. [Kirchhoff's theorem (Matrix-tree theorem).] *Let u and v be nodes of a graph G and indicate with $L(u, v)$ the submatrix obtained from the Laplacian matrix L of G by deleting row u and column v. Let $\tau(G)$ be the number of spanning trees of G, then*

$$\det(L(u, v)) = \tau(G).$$

This result can be also used, alternatively to the Perron-Frobenius Theorem, to prove that the Fiedler value is zero if and only if the graph is not connected [49]. Therefore, from the consequent result it follows immediately that a not connected graph will not contain any spanning tree.

Corollary 2.2.3. *Let G be a connected graph having n nodes, without self-loops. Suppose that $\sigma(L) = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$ is the spectrum of the Laplacian matrix L with $\lambda_1 = 0$ and that the non-zero eigenvalues of L are ordered in increasing order $\lambda_2 \leq \lambda_3 \leq \cdots \leq \lambda_n$. Then*

$$\tau(G) = \frac{1}{n}\lambda_2\lambda_3 \ldots \lambda_n$$

Laplacian eigenvectors were first studied by Fiedler [70–72] and independently by Donath and Hoffman [57]. The Laplacian matrix of a graph and its eigenvalues can be used in several areas of mathematical research, mainly discrete mathematics and more recently combinatorial optimization problems and have an interpretation in various physical and chemical theories. See [139] for a survey on known results about the spectrum of the Laplacian matrix of graphs with emphasis on the Fiedler value and its relation to numerous graph invariants, including connectivity, maximum cut, diameter, mean distance and bandwidth-type parameters of a network. Most of the results on the connection between $a(G)$ and invariants of graphs are consequences of the well-known Courant-Fisher Theorem [71] which states that

$$a(G) = \min_{\mathbf{x}^T\mathbf{e}=0,\ \mathbf{x}^T\mathbf{x}=1} \mathbf{x}^T L\mathbf{x}.$$

For example, the relation between the algebraic connectivity and the diameter $\mathrm{diam}(G)$ of a graph G with n nodes, is represented by the lower bound

$$\mathrm{diam}(G) \geq \frac{4}{n\lambda_2(G)}$$

proved in [140].

The Fiedler value, and in particular the associated eigenvector(s), can also be considered in relation to the problem of *graph partitioning* that consists of dividing the nodes of a graph into a number of disjoint groups, also called partitions. Indeed, the Fiedler vector has a very important property given by the following Theorem from [70].

Theorem 2.2.4. [Fiedler, 1975.] *Given $G = (V, E)$ a connected graph with n nodes and suppose L to be the graph Laplacian whose smallest eigenvalue is $\lambda_2 > 0$. Indicating with $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)^T$ the eigenvector associated to $\lambda_2 > 0$, let $k \in \mathbb{R}$ and partition the nodes in V into two sets*

$$V_1 = \{i \in V | x_i \geq k\}, \quad V_2 = \{i \in V | x_i < k\}.$$

Then the subgraphs of G induced by the sets V_1 and V_2 are connected.

Therefore, this shows that a connected graph can be partitioned into two distinct connected components using the sign of the entries of the Fiedler vector for a threshold value k; this eigenvector solves a relaxation of an NP-hard discrete graph partitioning problem [39] and it can also be shown that cuts based on the eigenvector associated to the Fiedler value give an approximation to the optimal cut [39, 168].

Spectral clustering using the Fiedler vector can be applied in bioinformatics whose key task is the accurate clustering of samples. In [90] a spectral algorithm based on the use of the Fiedler vector for spectral clustering, is adopted for studying microarray datasets published by cancer researchers. In this particular example the graph summarizes similarity of gene activity across different tissue sample; see also [145, 154].

Another application of the Fiedler vector arise in *image segmentation*. In computer vision, image segmentation is the process of partitioning a digital image into regions (sets of pixels, also known as super-pixels) with the goal of obtaining a compact representation of a picture into something that is more meaningful and easier to analyze. This problem can be formulated as a partitioning problem since an image can be seen as a graph considering the pixels as nodes; see [4, 164].

In this thesis we consider the use of the eigenvector associated to the Fiedler value, in relation to the seriation problem considering the archaeological investigation. Further details will be given below, in Section 2.4. In this application to the seriation problem we consider *bipartite graphs*, since the interrelationship between the units we want to rearrange, can be defined in terms of these particular networks defined in Section 1.3.

Recalling that a graph is bipartite if its vertices can be split into two disjoint subsets such that only edges between nodes belonging to different sets can occur, in our archaeological metaphor, the two disjoint nodes sets U and V represent the excavation sites (i.e. units) and the found artifacts (i.e. the type of the findings), respectively. In particular, in this interpretation of the seriation problem, the data matrix can be interpreted as the upper-right block of the adjacency matrix (1.3.1) associated with the bipartite graph of size $n + m$. Hence, the "adjacency" matrix A associated to the seriation problem, of size $n \times m$, is obtained by setting $a_{ij} = 1$ if the unit i contains objects of type j and 0 otherwise, and therefore is exactly the upper-right block of (1.3.1).

In case the element a_{ij} takes a value different from 1, we consider it as a weight indicating the number of objects of type j contained in unit i, or their percentage. In this case, we denote A as the *abundance matrix* of the considered seriation problem.

2.2.1 The similarity matrix

Let A be the $(0,1)$-matrix from a given seriation problem. The first mathematical definition of seriation was based on the construction of a symmetric matrix S known as *similarity matrix* [25, 158], where the element s_{ij} describes, in some way, the *likeness* of the nodes $i, j \in U$ representing two archaeological units. One possible definition of the similarity matrix is through the product $S = AA^T$, being A the adjacency matrix of the problem. In this case, s_{ij} equals the number of types shared between unit i and unit j. For example, considering the adjacency matrix represented in table 2.1, the similarity matrix will be

$$S = \begin{bmatrix} 4 & 2 & 3 & 2 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 2 & 4 & 2 & 3 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 3 & 2 & 6 & 3 & 1 & 3 & 2 & 1 & 1 & 0 & 0 \\ 2 & 3 & 3 & 5 & 1 & 2 & 1 & 1 & 2 & 0 & 0 \\ 0 & 2 & 1 & 1 & 3 & 1 & 0 & 1 & 0 & 1 & 1 \\ 2 & 1 & 3 & 2 & 1 & 5 & 2 & 2 & 1 & 1 & 1 \\ 0 & 0 & 2 & 1 & 0 & 2 & 3 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 2 & 1 & 2 & 1 & 1 & 1 \\ 0 & 0 & 1 & 2 & 0 & 1 & 1 & 1 & 3 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 2 & 2 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 2 & 3 \end{bmatrix}. \tag{2.2.1}$$

Clearly, with the matrix product AA^T we are considering the similarity matrix on the rows/units while by taking the product A^TA the attention is devoted to the similarity matrix on the columns that reports the likenesses between the types of the found artifacts and therefore, each element is equal to the number of units shared by the considered types. In the similarity matrix (see the example 2.2.1) the largest value on each row is the diagonal element, which reports the number of types associated to each unit. By permuting the rows and columns of S in order to cluster the largest values close to the main diagonal, one obtains a permutation of the corresponding rows of A that places the units similar in types closer to each other. This procedure does not provide an ordering for the types, i.e. for the columns of the data matrix. Then for example, one can consider the similarity matrix on the columns given by A^TA and, by symmetrically permuting its rows and columns, obtain a permutation of the columns of the data matrix A. It is worth noting that this operation of permuting rows and columns of the similarity matrix is not uniquely defined so, in general, starting from a data matrix and applying various permutations it is possible to obtain different orderings, reasonable from an archaeological point of view, for the units.

Summarizing, one can map the set of pairwise relative measurements among the objects to a similarity matrix which represents the information of the objects to be ordered. Then, the seriation problem can be modeled as a discrete optimization problem, whose goal is to find a permutation of the rows and columns of the similarity matrix which minimizes a given *objective function*. Methods based on the similarity matrix differ from one to another on how the similarity matrix is constructed or because of different objective functions used for evaluate the found permutations of the seriation data matrix.

Another way of determining the similarity matrix is obtained by using the so-called Robinson method developed by Robinson in 1951 [158]. The *Robinson method* is a statistical technique that constructs, indeed, a similarity matrix different from the one defined through the product between the data matrix and its transpose (or in the reverse order). It is based

on the concept that each type of artifact used in a certain period eventually decreases in popularity until it becomes forgotten. This method is probably the first documented example of a practical procedure based on the use of the similarity matrix, so its description is interesting in a historical perspective. In the original first formulation, it is applicable only to abundance archaeological seriation data, although later various authors proposed several new formulations of the Robinson method, some of them applicable also to incidence seriation data; see [50, 94, 117].

The method, starting from an abundance matrix $A \in \mathbb{R}^{n\times m}$ whose entries are in percentage form so that the sum of the values in each row is 100, computes the similarity matrix S by a particular rule, leading to a symmetric matrix of order n with entries between 0 (for rows with no types in common) and 200, which corresponds to units containing exactly the same types. Specifically, the entries of S are computed as

$$s_{i,j} = 200 - \sum_{k=1}^{n} \|a_{i,k} - a_{j,k}\|$$

and the similarity measured introduced is called *Robinson's Index of Agreement*. Then, the method searches for a permutation matrix P such that PSP^T has its largest entries as close as possible to the main diagonal. The same permutation is applied to the rows of the data matrix A to obtain a chronological order for the archaeological units. Since, as already remarked, the sequence can be read in both directions, external information given by the archaeologists must be used to choose an orientation.

The procedure of finding a permutation matrix P is, again, not uniquely specified. One way to deal with this problem was defined by Robinson who described an ideal arrangement of the elements of the similarity matrix, the so called *Robinson's form*. This structure of the similarity matrix places larger values close to the main diagonal, and lets off-diagonal entries be nonincreasing when moving away from the main diagonal. Precisely, a given symmetric matrix S is in *Robinson's form*, or is an R-matrix or a Robinson (similarity) matrix, if and only if its entries nonincrease monotonically along rows and columns when moving away the main diagonal, i.e. if and only if

$$s_{ij} \leqslant s_{ik}, \quad \text{if } j \leqslant k \leqslant i, \tag{2.2.2}$$

$$s_{ij} \geqslant s_{ik}, \quad \text{if } i \leqslant j \leqslant k. \tag{2.2.3}$$

Robinson's matrices play a fundamental role in the seriation problem, since the goal of seriation to order similar objects close to each other is best achieved by this special class of similarity matrices. Even though R-matrices were introduced by Robinson to model the seriation problem, they can also be used in the problem of building a consistent ranking of a set of objects given pairwise comparisons between these items [73]. We refer the reader interested in the application of Robinson's matrices to data analysis and visualization, to [61] and the references therein.

A symmetric matrix is said to be *pre-R* if and only if there exists a simultaneous permutation of its rows and columns which transforms it into Robinson's form. Hence, pre-R matrices correspond to well-posed seriation problems.

In the literature, a distinction is made between Robinson similarities and Robinson dissimilarities (or *anti-Robinson*) matrices whose values in all rows and columns are monotone nondecreasing when moving away from the main diagonal. Formally, an $n \times n$ symmetric

matrix D is in anti-Robinson's form if and only if the following two conditions hold [100]

$$d_{ik} \leqslant d_{ij} \quad \text{if } i \leqslant k \leqslant j, \tag{2.2.4}$$
$$d_{kj} \leqslant d_{ij}, \quad \text{if } i \leqslant k \leqslant j. \tag{2.2.5}$$

In an anti-Robinson matrix the smallest dissimilarity values appear on the main diagonal since the dissimilarity between an element and itself is minimum. Since we are not interested in anti-Robinson matrices, in the rest of the thesis we will only consider similarity matrices in Robinson's form, especially in Section 2.4.

The following matrices, excerpted from [153], are examples of R, not-R and pre-R matrices:

$$\begin{bmatrix} 6 & 4 & 2 & 2 \\ 4 & 8 & 5 & 3 \\ 2 & 5 & 9 & 4 \\ 2 & 3 & 4 & 7 \end{bmatrix} R\text{-form}, \qquad \begin{bmatrix} 6 & 4 & 9 & 2 \\ 4 & 8 & 5 & 3 \\ 9 & 5 & 9 & 4 \\ 2 & 3 & 4 & 7 \end{bmatrix} \text{not-}R, \qquad \begin{bmatrix} 9 & 2 & 5 & 4 \\ 2 & 6 & 4 & 2 \\ 5 & 4 & 8 & 3 \\ 4 & 2 & 3 & 7 \end{bmatrix} \text{pre-}R.$$

Note that the last matrix, which is pre-R, is obtained by applying the permutation $P = [\mathbf{e}_2, \mathbf{e}_3, \mathbf{e}_1, \mathbf{e}_4]$ (with $\mathbf{e}_i$ the ith columns of the identity matrix) simultaneously, in order to preserve the symmetry, to the rows and columns of the first matrix. Note that, understanding if a given symmetric matrix is pre-R is often difficult, considering that not all symmetric matrices can be brought into the Robinson's form. For other interesting references on R-matrices and their detection, see [36, 119, 120, 155, 163].

Seriation techniques based on the construction of the similarity matrix, are divergent from other methods that involve directly the data matrix in finding units rearrangements; see [153] for further details and references.

2.2.2 Connections between the seriation problem and other combinatorial problems

The seriation problem is related to another combinatorial problem, namely, the NP-hard *travelling salesman problem* (TSP). The origin of the TSP is enigmatic and it was mathematically formulated by the mathematicians W.R. Hamilton and T. Kirkman who worked on similar problems. It can be formulated as the problem of finding, given a list of cities and the distances between each pair of cities, which is the shortest possible route that visits each place and returns to the starting city. The relationship between these two combinatorial problems, was first exploited by Wilkinson [184] who made rigorous the connections between the TSP and the reordering of the similarity matrix considering the seriation problem from this matrix point of view. Later some other authors examined the analogy between these two combinatorial problems; see [58, 99]. Indeed, the TSP can be formulated as the problem of finding the permutation of the rows and columns of a symmetric *connection* matrix C whose entry c_{ij} represents the distance between the ith and jth cities the salesman has to visit, that places smaller entries close to the main diagonal of C. The basics of this formulation of the TSP is the same used for rearranging the similarity matrix except for the constraint that the last visited city must coincide with the starting one and for the fact that, while in the reordering of the similarity matrix the largest values are to be placed near the main diagonal, in the TSP the smaller entries are considered; see [153] for further details and references on the travelling salesman problem and its connection to the seriation problem.

In case of incidence seriation data, the seriation problem can be rigorously defined in terms of the so-called *consecutive ones problem* [147], whose aim is to find all the permutations of the rows of a binary matrix that place the 1's consecutively in each column. If such permutations exist, then the matrix is said to have the *consecutive ones property* (C1P) for columns. The equivalent property for rows can be similarly defined. A $(0,1)$-matrix featuring the C1P is said to be in *Petrie form*, or in P-form, or to be a P-matrix (along rows, or columns, or both). If a $(0,1)$-matrix can be permuted to become a P-matrix, then it is said to be a *pre-P* matrix. The following matrices are examples of P along rows, along columns, along both rows and columns, not-P and pre-P matrices:

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

P-form along rows,

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

P-form along columns,

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

P-form along both rows and columns

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

not-P.

The matrix

$$\begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

is pre-P and it is obtained by applying the permutations $P_r = [\mathbf{e}_3, \mathbf{e}_4, \mathbf{e}_1, \mathbf{e}_5, \mathbf{e}_2]$ to the rows and $P_c = [\mathbf{e}_1, \mathbf{e}_3, \mathbf{e}_4, \mathbf{e}_5, \mathbf{e}_6, \mathbf{e}_2]$ to the columns of the third matrix which is in "complete" P-form since it satisfies the C1P along rows and columns.

For more details about matrices with the consecutive ones property and algorithms for their analysis, we refer the interested reader to [56] and the references therein. The problem of verifying if a matrix possesses the consecutive ones property has applications in different fields, such as computational biology and recognition of interval graphs [21, 38].

The connection between C1P and seriation has been first investigated by Kendall who proved the following result which represents the first link between Petrie's and Robinson's matrices [109].

Theorem 2.2.5. (Kendall, 1969.) *Let A be a $(0,1)$ matrix and consider the similarity matrix $S = AA^T$. Given Π a permutation matrix, if ΠA has the consecutive ones in its columns then $\Pi S \Pi^T$ is in Robinson's form.*

From this it follows that if A is a $(0,1)$ Petrie matrix, then $S = AA^T$ is a Robinson similarity matrix, but the converse is not true in general. It can also be proved that if A

is a pre-P data matrix and $S = AA^T$ is a similarity matrix in Robinson's form, then A is a P-matrix. Hence, checking if a data matrix has the C1P reduces somehow to a special instance of Robinson similarity matrix recognition.

In an application to genetics, Fulkerson and Gross characterized pre-P matrices in terms of recognition of interval graphs and give an algorithm for solving the seriation problem in polynomial time [76]. They proved that the problem of determining whether an incidence data matrix is pre-P, is a P problem. Empirical archaeological data that lead to a pre-P data matrix are commonly called "perfect" data since, applying the algorithm of Fulkerson and Gross, it is possible to solve the seriation problem in linear time. Indeed, the seriation problem in terms of P-matrices, translates into the problem of finding the permutations transforming a given $(0, 1)$ data matrix into a P-matrix, if the matrix is pre-P, or that best "approximate" a P-form if the matrix is not pre-P, i.e. in case of "imperfect" data. As a consequence, pre-P seriation data matrices and pre-R similarity matrices correspond to well-posed instances of the seriation problem.

In presence of perfect data, the spectral algorithm from [6] returns a classification of the ordering permutations in terms of a compact data structure called PQ-tree described in Section 2.3. The algorithm devised by Atkins and collaborators is improved and implemented as a Matlab toolbox in [41], as described and discussed in Section 2.4. Specifically, given a pre-R similarity matrix, it constructs a PQ-tree describing the set of all the permutations of rows and columns that lead to an R-matrix.

However, perfect data are exceptionally rare in nature. In fact, in most cases the seriation empirical data coming from real archaeological sites are imperfect, in the sense that the data matrices are non in pre-P form. Hence, there is the problem of finding a permutation that approximate a Petrie data matrix or a Robinson similarity matrix. Further details will be given in Section 2.5.

2.3 PQ-trees

A *PQ-tree* is a data structure introduced by Booth and Lueker [21] to encode a family of permutations of a set of elements and solve problems connected to finding admissible permutations according to specific rules.

A PQ-tree $\mathcal{T}$ over a set $U = \{u_1, u_2, \ldots, u_n\}$ is a rooted, ordered tree whose leaves are in one-to-one correspondence with the elements of U and their order gives a reordering of the elements of the considered set. The internal (non-leaf) nodes of $\mathcal{T}$ are distinguished as either *P-nodes* or *Q-nodes*. The only difference between them is the way in which their children are treated and can be reordered. Namely, for a P-node, its children may be arbitrarily reordered so all possible permutations of the children leaves are permitted; for a Q-node only one order and its reverse are allowed since the children leaves may be ordered only left-to-right or right-to-left. The root of the tree can be either a P or a Q-node.

We will represent graphically a P-node by a circle, and a Q-node by a rectangle, following the same representation used in the paper by Booth and Lueker. The leaves of T will be displayed as triangles, and labeled by the elements of U. The *frontier* of T is one possible permutation of the elements of U, obtained by reading the labels of the leaves from left to right.

We recall two definitions from [21]; the first one gives a further set of restrinctions on the nodes of a PQ-tree.

Definition 2.3.1. *A PQ-tree is proper when each of the following conditions hold:*

i) every $u_i \in U$ appears precisely once as a leaf. This is because PQ-trees are supposed to represent permutations of a set, so it does not make sense for an element to appear more than once or to not appear at all;

ii) every P-node has at least two children. In this way, long chains of nodes having only a single child are ruled out;

iii) every Q-node has at least three children. This again eliminates chains but also serves a more technical purpose. Indeed, as explained below, there is no real distinction between a P-node and a Q-node in case of exactly two children, hence it is convenient to remove this redundancy.

The rearrangement of the leaves of a PQ-tree can be better formalized using the following definition.

Definition 2.3.2. *Two PQ-trees are said to be equivalent if and only if one can be transformed into the other by applying a sequence of the following two equivalence transformations:*

i) arbitrarily permute the children of a P-node;

ii) reverse the children of a Q-node.

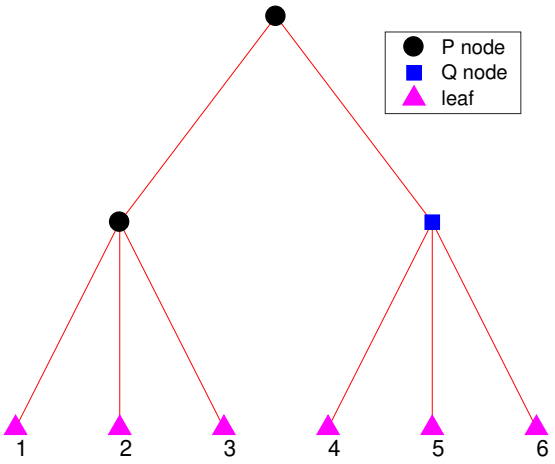
Hence, a PQ-tree represents in a compact way permutations of the elements of a set through admissible reorderings of its leaves. Each transformation in Definition 2.3.2 specifies an allowed reordering of the nodes within a PQ-tree. For example, a tree with a single P-node represents the equivalence class of all permutations of the elements of U, while a tree with a single Q-node represents both the left-to-right and right-to-left orderings of the leaves. A tree with a mixed P-node and Q-node structure represents the equivalence class of a constrained permutation, where the structure of the tree determines the constraints on the admissible permutations.

Figure 2.1 displays a PQ-tree and the admissible permutations it represents and encodes. A package for representing and manipulating PQ-trees is contained in the `PQser` toolbox. Both the figure and the permutations encoded in the tree have been obtained by applying specific functions contained in it.

The PQ-tree data structure has been exploited in a variety of applications, from archaeology and chronology reconstruction [6] to molecular biology with DNA mapping and sequence assembly [84]. The first problem to which it was applied is in an algorithm for testing the consecutive ones property (C1P) for matrices [21]; see Paragraph 2.2.2. We recall that, in this case, one seeks to find a permutation of the rows of a matrix that places the nonvanishing entries within each column consecutively in relation to the C1P on the columns. If the consecutive ones property is sought on the rows then the aim is to find a permutations of the columns of the given matrix that places the ones one after another in each row.

2.3.1 Matlab implementation of PQ-trees

In this subsection we present a Matlab implementation of PQ-trees by describing the package of our toolbox that defines a data structure to store a PQ-tree and provides also the Matlab functions to manipulate and visualize it.



1	2	3	4	5	6
1	2	3	6	5	4
1	3	2	4	5	6
1	3	2	6	5	4
2	1	3	4	5	6
2	1	3	6	5	4
2	3	1	4	5	6
2	3	1	6	5	4
3	1	2	4	5	6
3	1	2	6	5	4
3	2	1	4	5	6
3	2	1	6	5	4
4	5	6	1	2	3
4	5	6	1	3	2
4	5	6	2	1	3
4	5	6	2	3	1
4	5	6	3	1	2
4	5	6	3	2	1
6	5	4	1	2	3
6	5	4	1	3	2
6	5	4	2	1	3
6	5	4	2	3	1
6	5	4	3	1	2
6	5	4	3	2	1

Figure 2.1: On the left, a PQ-tree over the set $U = \{1, \ldots, 6\}$; on the right, the 24 admissible permutations encoded in the tree.

The `PQser` toolbox for Matlab is distributed as a compressed archive. It is available from Netlib (`http://www.netlib.org/numeralgo/`) as the na48 package and also on the web page `http://bugs.unica.it/~gppe/software/#pqser`; it is described in [41].

In the `PQser` toolbox, a PQ-tree T is a `struct` variable (i.e., a record) composed by two fields. The first field, `T.type`, specifies the type of node, i.e., P, Q, or a leaf, in the case of a trivial tree. The second field, `T.value`, is a vector which provides a list of PQ-trees, recursively defined. In the case of a leaf, this field contains the index of the unit it represents.

For example, the graph in Figure 2.1 was obtained by the following piece of code

```
v(1) = pnode([1 2 3]); % create a P-node with three leaves
v(2) = qnode([4 5 6]); % create a Q-node with three leaves
T = pnode(v); % create a P node pointing to the previous two nodes
pqtreeplot(T) % visualize the PQ-tree
```

the resulting data structure for the PQ-tree is

```
T =
  struct with fields:
     type: 'P'
    value: [1x2 struct]
```

and the permutations encoded in T and represented in the table in Figure 2.1, are extracted by using the function `pqtreeperms` whose structure is summarized in Algorithm 2

```
perms_matrix = pqtreeperms(T)
```

The functions for creating a PQ-tree and used for manipulating this data structure are listed in Table 2.2. As it is customary for graph-like data processing, most of these functions are recursive. The function `mnode` creates an additional type of node, called M-node, which is designed to deal with multiple Fiedler values; we will comment on it in the following, see Subsection 2.5.1. The function `pqtreegetnode` is designed to extract a subtree from the considered PQ-tree and allows one to plot the subtree in a a new figure, as explained below; `pqtreenodes` converts the PQ-tree to the format used by the function `treeplot` defined

`pnode`	create a P-node
`qnode`	create a Q-node
`lnode`	create a leaf
`mnode`	create an M-node
`pqtreeplot`	plot a PQ-tree
`pqtreeNperm`	number of admissible permutations in a PQ-tree
`pqtreeperms`	extract all admissible permutations from a PQ-tree
`pqtree1perm`	extract one admissible permutation from a PQ-tree
`pqtreegetnode`	extract a subtree from a PQ-tree
`pqtreenodes`	converts a PQ-tree to Matlab `treeplot` format

Table 2.2: Functions in the `PQser` toolbox devoted to the manipulation of PQ-trees.

in Matlab. Both of these two last functions are utilised by the function `pqtreeplot` and are not intended to be called by a user. All the functions are documented via the usual Matlab `help` command; as an example we report the `help` output for the function `pqtreeplot` which allows to set some attributes of the plot:

```
help pqtreeplot

pqtreeplot plot a PQ tree.
    pqtreeplot(T) plot in the current figure the PQ tree whose root
    is T. If the user clicks on one node with the left mouse button,
    the corresponding subtree is extracted, it is plotted in a new
    figure, and it is saved to a variable in the workspace.
    pqtreeplot(T,opts) optionally passes a set of options.

    options:
    opts.labelson if set to 1 (default) the values of the leaves are
                 displayed in the plot.
    opts.fontsize sets the size of the font (default 10)
    opts.markersize sets the size of the markers (default 8)
```

We now report in Algorithm 1 the structure of `pqtreeNperm`, a function which returns the number N of all the permutations contained in the tree whose root T is given as input. In the particular case of a leaf, only one permutation is possible (line 2–3). Otherwise, we consider the vector $\mathbf{c}$ of size k, containing the children nodes of the root of T (line 5). The algorithm calls itself recursively on each component of $\mathbf{c}$ (line 8). In the case of a Q-node the number of permutations is doubled, because only one ordering and its reverse are admissible, whereas for a P-node the number is multiplied by the factorial of k, since in this case all the possible permutations of the children leaves are allowed. The same procedure is applied to an M-node since this new type of node is momentary treated as a P-node; see Section 2.5 for details.

As an additional example of the functions included in our toolbox, in Algorithm 2 we summarize the structure of the function that extracts all the N admissible permutations encoded in the PQ-tree whose root T is given as input. The output of this function is a matrix containing all the possible frontiers of the tree obtained by using the MATLAB default function kron that computes the Kronecker tensor product. We recall that the Kronecker product, denoted by the symbol $\otimes$, of $A \in \mathbb{C}^{m\times n}$ and $B \in \mathbb{C}^{p\times q}$ is the $mp \times nq$ block matrix defined as $A \otimes B = (a_{ij}B)$.

Since all the admissible permutations are equivalent, in a way that will be explained later in relation to the spectral algorithm for perfect seriation data, one can extract one of the

Algorithm 1 Compute the number of admissible permutations in a PQ-tree.

```
function N = pqtreeNperm(T)
if T is a leaf
    N = 1
else
    c = T.value, k = length(c)
    p = 1
    for i = 1, ..., k
        p = p * pqtreeNperm(c_i)
    end for
    if T is a Q-node
        N = 2 * p
    else
        N = factorial(k) * p
    end if
end if
```

possible boundaries of the PQ-tree by calling the function `pqtree1perm`.

The toolbox includes an interactive graphical tool for exploring a PQ-tree T. Indeed, after displaying T by `pqtreeplot`, it is possible to extract a subtree by clicking on one node with the left mouse button. In this case, the corresponding subtree is extracted by using the function `pqtreegetnode`, it is plotted in a new figure, and it is saved to the variable `PQsubtree` in the workspace. This feature is particularly useful when analyzing a large PQ-tree, but it is not available when the toolbox is run from Octave.

2.4 Seriation in the case of *perfect* data: a spectral algorithm for the seriation problem

In this section, considering the seriation problem in the presence of *perfect* data, we review the spectral algorithm introduced in [6] for the seriation problem, and describe our Matlab implementation. The algorithm uses the entries of the Fiedler vector of the Laplacian matrix associated to the problem for finding all the possible reorderings of the nodes in the considered seriation problem, as it will be explained in the rest of the section. In particular, sorting the elements of the Fiedler vector reorders a similarity matrix in Robinson's form in the noiseless case, i.e. if the similarity matrix describing the seriation problem is in pre-P form. This technique, based on the use of the eigenvector associated to the second smallest eigenvalue of the Laplacian matrix, was introduced previously for the problem of reordering a sparse matrix to reduce its envelope size [12].

Given the set of units $U = \{u_1, u_2, \ldots, u_n\}$, we will write $i \preccurlyeq j$ if u_i precedes u_j in a chosen ordering. In [6], the authors consider a symmetric bivariate *correlation function* f reflecting the desire for units i and j to be close to each other in the sought sequence. The aim is to find all index permutation vectors $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_n)^T$ such that

$$\pi_i \preccurlyeq \pi_j \preccurlyeq \pi_k \quad \Longleftrightarrow \quad f(\pi_i, \pi_j) \geq f(\pi_i, \pi_k) \quad \text{and} \quad f(\pi_j, \pi_k) \geq f(\pi_i, \pi_k). \quad (2.4.1)$$

Algorithm 2 Extract all admissible permutations from a PQ-tree.

```
1:  function P = pqtreeperms(T)
2:  if T.type is a leaf
3:      P = T.value
4:  else if T.type is a Q-node
5:      w = T.value, lw: length of w
6:      R = pqtreeperms(w_1)
7:      for i = 2, ..., lw
8:          S = pqtreeperms(w_i)
9:          R: a matrix constructed using the Kronecker tensor product
10:     end for
11:     P = [R; reverse(R)]
12: else
13:     w = T.value, nw: length of w
14:     for i = 1, ..., nw
15:         W_i = pqtreeperms(w_i)
16:     end for
17:     PS:  nw! × nw matrix with all the possible permutations of the nw elements
18:     for k = 1, ..., number of the rows of PS
19:         p = PS(k, :)
20:         R = W(p_1)
21:         for i = 2, ..., nw
22:             S = W(p_i)
23:             R: a matrix constructed using the Kronecker tensor product
24:         end for
25:         sR : number of the rows of R
26:         if k = 1
27:             P zeros matrix of order N× columns number of R
28:             R composed by the first sR rows of P
29:             iperm = sR + 1
30:         else
31:             P(iperm : iperm + sR − 1, :) = R
32:             iperm = iperm + iR
33:         end if
```

It is natural to associate to such a correlation function a real symmetric matrix F, whose entries are defined by $f_{ij} = f(i,j)$. This matrix plays exactly the role of the similarity matrix S discussed in subsection 2.2.1, as the following theorem states.

Theorem 2.4.1. *A matrix F is an R-matrix if and only if* (2.4.1) *holds.*

Proof. Let us assume that the permutation $\boldsymbol{\pi}$ which realizes (2.4.1) has already been applied to the units. Then, since a permutation of the units corresponds to a simultaneous permutation

of the rows and columns of the matrix F, we obtain

$$i \leq j \leq k \quad \Longleftrightarrow \quad f_{ij} \geq f_{ik} \quad \text{and} \quad f_{jk} \geq f_{ik}.$$

The first inequality $f_{ij} \geq f_{ik}$ is exactly (2.2.3). Keeping in mind the symmetry of F and cyclically permuting the indices, we get from the second inequality

$$j \leq k \leq i \quad \Longleftrightarrow \quad f_{ij} \leq f_{ik},$$

which corresponds to (2.2.2). □

If a seriation data set is described by an adjacency (or abundance) matrix A, we set $F = AA^T$ to be the similarity matrix for the considered seriation problem. If F is pre-R, as mentioned before, then there exists a row/column permutation that takes it into R-form. Unfortunately, this property cannot be verified in advance, in general. It can be ascertained, e.g., after applying the spectral algorithm on which this section is focused. Indeed, the spectral algorithm represents an approach to recognize Robinson's similarities. Other different recognition algorithms, i.e., algorithm to decide whether or not a given matrix is pre-R and then return an ordering that leads to a Robinson's matrix, have been widely considered. Some of them are based on variants of the well known graph traversal algorithm Breadth-First Search (BFS) where nodes are explored by giving preference to those vertices whose neighbors have been visited first; see [162] and references inward.

The authors' approach in [6] (see also [66]) is to minimize the following *penalty* function

$$h(\mathbf{x}) = \frac{1}{2} \sum_{i,j=1}^{n} f_{ij}(x_i - x_j)^2, \quad \mathbf{x} \in \mathbb{R}^n,$$

whose value is small for a vector $\mathbf{x}$ such that each pair (i, j) of highly correlated units is associated to components x_i and x_j with close values. Once the minimizing vector $\mathbf{x}_{\min}$ is computed, it is sorted in either nonincreasing or nondecreasing value order, yielding $\mathbf{x}_{\boldsymbol{\pi}} = (x_{\pi_1}, \ldots, x_{\pi_n})^T$. The permutation of the units $\boldsymbol{\pi}$ realizes (2.4.1).

Note that h does not have a unique minimizer, since its value does not change if a constant is added to each of the components x_i of the vector $\mathbf{x}$. In order to ensure uniqueness and rule out the trivial solution, it is necessary to impose two suitable constraints on the components of the vector $\mathbf{x}$. The resulting minimization problem is:

$$\begin{aligned} &\text{minimize} \quad h(\mathbf{x}) = \frac{1}{2} \sum_{i,j=1}^{n} f_{ij}(x_i - x_j)^2 \\ &\text{subject to} \quad \sum_i x_i = 0 \quad \text{and} \quad \sum_i x_i^2 = 1. \end{aligned}$$

The solution to this problem can be used as a heuristic for sequencing and it may be obtained from the Fiedler vector of the Laplacian L of the correlation matrix F. Letting $D = \operatorname{diag}(d_i)$ be the degree matrix, $d_i = \sum_{j=1}^{n} f_{ij}$, it is immediate to observe that

$$h(\mathbf{x}) = \frac{1}{2} \sum_{i,j=1}^{n} f_{ij}(x_i^2 + x_j^2 - 2x_i x_j) = \mathbf{x}^T D\mathbf{x} - \mathbf{x}^T F\mathbf{x}.$$

This shows that the previous optimization problem can be rewritten as

$$\min_{\|\mathbf{x}\|=1,\ \mathbf{x}^T\mathbf{e}=0} \mathbf{x}^T L\mathbf{x} \tag{2.4.2}$$

where $L = D - F$. The constraints require $\mathbf{x}$ to be a unit vector orthogonal to $\mathbf{e}$. Since L is symmetric, all the eigenvectors except $\mathbf{e}$ can be chosen to satisfy the constraints. Consequently, a Fiedler vector is a solution to the constrained minimization problem.

The spectral algorithm by Atkins *et al.*, is based on the following results which delineate properties of the Fiedler vector of Robinson's similarity matrices and show how to reorder pre-R matrices by using the above heuristic.

Theorem 2.4.2. [Theorem 3.2 [6].] *If $F \in \mathbb{R}^{n\times n}$ is a Robinson similarity matrix then it has a monotone Fiedler vector $\mathbf{v}$, i.e. its entries are nonincreasing or nondecreasing.*

The following Theorem (Theorem 3.3 from [6]) implies, under suitable assumptions, that a reordering of the Fiedler vector takes a pre-R matrix to R-form.

Theorem 2.4.3. *Assume that $F \in \mathbb{R}^{n\times n}$ is a pre-R similarity matrix such that its Fiedler value is simple with associated eigenvector $\mathbf{v}$ with no repeated entries. Let π_1 and π_2 be the permutations induced by sorting the values of the Fiedler vector $\mathbf{v}$ in increasing and decreasing order respectively. Then the matrices F_1 and F_2, obtained by applying the found permutations to the rows and columns of F, are Robinson's matrices and no other permutations of F bring it to an R-form.*

Thus, the aforementioned theorems show that sorting monotonically the Fiedler vector of a pre-R similarity matrix reorders it as a Robinson matrix.

This confirms that the problem is well posed only when F is pre-R. Nevertheless, real data sets may be inconsistent or *imperfect*, in the sense that they do not necessarily lead to pre-R similarity matrices. In such cases, it may be useful to construct an approximate solution to the seriation problem, and sorting the entries of the Fiedler vector generates an ordering that tries to keep highly correlated elements close to each other. This is relevant because techniques based on Fiedler vectors are used for the solution of different sequencing problems [12, 84, 90, 106]. In particular, they are employed in complex network analysis, e.g., for community detection and partitioning of graphs [64, 67]. Nevertheless, the solution is not proved to be the "best" approximate to the seriation problem; we will discuss seriation in case of imperfect data, in Section 2.5.

The algorithm proposed in [6] is based upon the above idea, and uses a PQ-tree to store the permutations of the units that produce a solution to the seriation problem; our Matlab implementation is described in Algorithm 3. Summarizing, this spectral algorithm for recognizing Robinson matrices consists of determining the Laplacian matrix L corresponding to the given similarity matrix F associated to the considered seriation problem and then sorting the entries of the Fiedler vector of L either in increasing or decreasing order gives an ordering for the units/types depending on how F is constructed (see 2.2.1). Thus, if the similarity matrix $\tilde{F}$ obtained after the reordering is in Robinson's form, from Theorem 2.4.3 it follows, under suitable assumptions, that the starting similarity matrix F is pre-R whereas if $\tilde{F}$ is not an R-matrix then F is not pre-R.

The algorithm starts by translating all the entries of the correlation matrix so that the smallest one is 0 , i.e.,

$$\tilde{F} = F - \alpha \mathbf{e}\mathbf{e}^T, \qquad \alpha = \min_{i,j} f_{ij}; \tag{2.4.3}$$

see line 3 of Algorithm 3. This is justified by the fact that F and $\tilde{F}$ have the same Fiedler vectors and that if F is an irreducible R-matrix such a translation ensures that the Fiedler

Algorithm 3 Spectral sort algorithm.

```
1: function T = spectrsort(F, U)
2: n = row size of F
3: α = min_{i,j} f_{i,j}, if α ≠ 0, e = (1, ..., 1)^T, F = F − αee^T, end
4: call getconcomp to construct the connected components {F_1, ..., F_k} of F
5:       and the corresponding index sets U = {U_1, ..., U_k}
6: if k > 1
7:    for j = 1, ..., k
8:       v(j) = spectrsort(F_j, U_j)
9:    end for
10:   T = pnode(v)
11: else
12:   if n = 1
13:      T = lnode(U)
14:   else if n = 2
15:      T = pnode(U)
16:   else
17:      L = Laplacian matrix of F
18:      compute (part of) the eigenvalues and eigenvectors of L
19:      determine multiplicity n_F of the Fiedler value according to a tolerance τ
20:      if n_F = 1
21:         x = sorted Fiedler vector
22:         t number of distinct values in x according to a tolerance τ
23:         for j = 1, ..., t
24:            u_j indices of elements in x with value x_j
25:            if u_j has just one element
26:               v_j = lnode(u_j)
27:            else
28:               v(j) = spectrsort(F(u_j, u_j), U(u_j, u_j))
29:            end if
30:         end for
31:         T = qnode(v)
32:      else
33:         T = mnode(U)
34:      end if
35:   end if
36: end if
```

value is a simple eigenvalue of L. The operation of subtracting the smallest value from all the correlation values does not change whether or not the given similarity matrix is pre-R. In this way, the authors in [6] deal with the case when the Fiedler vector does not satisfy the hypothesis in Theorem 2.4.3 proving that if F is a a pre-R irreducible matrix having zero

smallest off-diagonal entry, then the Fiedler value is simple [6, Lemma 4.1 and Theorem 4.6]. Our software allows the user to disable this procedure (see Table 2.4 below) as he/she may decide to suitably preprocess the similarity matrix in order to reduce the computational load. Indeed, the translation procedure is repeated each time the algorithm calls itself recursively.

Algorithm 4 Determine the connected components of a graph.

```
1:  function U = getconcomp(F)
2:  preallocate the cell-array U, chlist =empty vector
3:  root ={node 1}, list = root, n =row size of F
4:  i = 0, flag = true (logical variable)
5:  while flag
6:      i = i + 1
7:      list = graphvisit(root, list)
8:      U{i} = list
9:      update chlist adding the nodes in list and sort the vector
10:     flag = true if the number of elements in chlist is different from n
11:             otherwise flag = false
12:     if flag
13:         choose the root for a new connected component
14:         if there are no connected components left
15:             exit
16:         end if
17:         list = root
18:     end if
19: end while
```

If the matrix F is reducible, i.e. its support graph is not connected, the irreducible blocks of F correspond to connected components and, then, the seriation problem can be decoupled dealing with each of them independently [6, Lemma 4.2]. Lines 4–5 of the algorithm detect irreducible blocks of the correlation matrix by using the function getconcomp.m, which also identifies the corresponding index sets. The function, described in Algorithm 4, constructs a *cell array* containing the vector of indices which identify each connected component of the graph corresponding to the input similarity matrix. It calls the function graphvisit.m, which visits a graph starting from a chosen node; see Algorithm 5. This function explores the graph defined by an adjacency matrix beginning from the node "root" given as input, and returns a list of the visited nodes. Note that these two functions, in order to reduce the stack consumption due to recursion, use a global variable (a variable that is visible and hence accessible throughout the program) to store the correlation matrix.

If more than one connected component is found, then the function calls itself on each component, and stores the returned lists of nodes as children of a P-node (lines 7–10). If the matrix is irreducible, the dimension n of the matrix is considered (lines 12–16). The cases $n = 1, 2$ are trivial. If $n > 2$, the Laplacian matrix L is computed, as well as the Fiedler value and vector (lines 17–18). Depending on the matrix being "small" or "large" different algorithms are used. For a small scale problem the full spectral decomposition of the Laplacian is computed by the `eig` function of Matlab. For a large scale problem, a small

Algorithm 5 Visit a graph starting from a node.

```
1:  function list = graphvisit(root, list)
2:  construct the list l of the indices of the nodes connected to the root
3:  initialize an empty list nlist
4:  find the elements of l which are not in list
5:  add the new elements to list and to nlist
6:  if nlist is not empty
7:      sort list
8:      for each node i in nlist
9:          list = graphvisit(nlist(i), list)
10:     end for
11: end if
```

subset of the eigenvalues and eigenvectors are evaluated using the `eigs` function, which is based on a Krylov space projection method. The `PQser` toolbox computes by default the eigenpairs corresponding to the three eigenvalues of smallest magnitude, since they are sufficient to understand if the Fiedler value is simple or multiple, but the default value can be modified. The choice between the two approaches is automatically performed and it may be influenced by the user; see Table 2.4 in Section 2.4.1.

The algorithm determines the multiplicity of the Fiedler value according to a given tolerance. If the Fiedler value is a simple eigenvalue of L, the algorithm sorts the elements of the current list according to the reordering of the Fiedler vector and stores them as the children of a Q-node. To deal with the case when the Fiedler vector has recurrent values, the problem can be decoupled by applying the spectral algorithm recursively to each submatrix of F identified by the indices corresponding to the set of repeated entries. Specifically, if two or more values of the Fiedler vector are repeated the function invokes itself recursively (line 28), in accordance with [6, Theorem 4.7]; on the contrary, the corresponding node becomes a leaf (line 26). In our implementation we introduce a tolerance τ to distinguish "equal" and "different" numbers: a and b are considered "equal" if $|a - b| < \tau$. The default value for τ is 10^{-8}.

In the case of a multiple Fiedler value, our algorithm constructs an "M-node" (line 33). This new type of node is introduced in order to flag this particular situation, which will be further discussed in Subsection 2.5.1.

Algorithm 3 produces a PQ-tree whether the similarity matrix F, corresponding to the considered seriation problem, is a pre-R matrix or not. If all the Fiedler values computed during the recursive calls of the algorithm are simple, then the starting correlation matrix is pre-R and any permutation encoded in the PQ-tree will take it to R-form. In the presence of a multiple Fiedler vector the problem is not well posed and an approximate solution is computed, as previously mentioned. We will describe this situation in the Section 2.5 concerning the seriation problem in case of imperfect data.

The number N of all the admissible permutations generated by the algorithm can be obtained by counting all the possible boundaries of the tree using the function pqtreeN-perm.m reported in Algorithm 1. In the case of a PQ-tree consisting of a single Q-node N is equal to 2, because only the left-to-right order of the children leaves and its reverse are possible. For a single P-node, the number of all the permutations is the factorial of

the number of the children. An M-node is temporarily treated as a P-node, although we experimentally observed that not all the permutations are admissible; this aspect is discussed in Subsection 2.5.1.

2.4.1 Implementation of spectral seriation

The functions related to the implementation of the spectral algorithm described above and included in the `PQser` toolbox are listed in Table 2.3. Besides the function `spectrsort`, which implements Algorithm 3, there is a version of the same method called `pspectrsort`, parallelized with respect to the irreducible blocks, which distributes the `for` loop at line 7 of the algorithm among the available processing units. In order to execute the function `pspectrsort`, the Parallel Computing Toolbox must be present in the current Matlab installation.

`spectrsort`	spectral sort for the seriation problem
`pspectrsort`	parallel version of `spectrsort`
`fiedvecs`	compute the Fiedler vectors and values of a Laplacian
`getconcomp`	determine the connected components of a graph
`graphvisit`	visit a graph starting from a node
`distinct`	sort and level the elements of a vector
`lapl`	construct the graph Laplacian of a matrix
`testmatr`	test matrices for PQser

Table 2.3: Functions in the `PQser` toolbox devoted to the solution of the seriation problem.

The function `testmatr` allows one to create some simple test problems. The remaining functions of Table 2.3 are not likely to be used in the common use of the toolbox. They are made available to the expert user, who may decide to call them directly or to modify their content.

`tau`	tolerance used to distinguish between "equal" and "different" values (`spectrsort` and `fiedvecs`, def. 10^{-8})
`translate`	apply translation (2.4.3) (`spectrsort`, def. true)
`lrg`	used to select small scale or large scale algorithm (`fiedvecs`, true if the input matrix is sparse)
`nlarge`	if matrix size is below this value, the small scale algorithm is used (`fiedvecs`, def. 1000)
`neig`	number of eigenpairs to be computed when the large scale algorithm is used (`fiedvecs`, def. 3)
`maxncomp`	maximum number of connected components (`getconcomp`, def. 100)
`bw`	half bandwidth of test matrix (`testmatr`, type 2 example, def. 2)
`spar`	construct a sparse test matrix (`testmatr`, type 2 example, def. true)

Table 2.4: Tuning parameters for the `PQser` toolbox; the functions affected are reported in parentheses, together with the default value of each parameter.

The toolbox has some tuning parameters, which are set to a default value. They can be modified by the user by passing to a function, as an optional argument, a variable of type `struct` with fields chosen among the ones listed in Table 2.4. For example:

```
opts.translate = 0;
T = spectrsort(F,opts);
```

applies Algorithm 3 to a similarity matrix F omitting the translation process described in (2.4.3).

To illustrate the use of the toolbox, we consider a similarity matrix R satisfying the Robinson criterion

$$R = \begin{bmatrix} 200 & 150 & 120 & 80 & 40 & 0 & 0 & 0 & 0 & 0 \\ 150 & 200 & 160 & 120 & 80 & 40 & 0 & 0 & 0 & 0 \\ 120 & 160 & 200 & 160 & 120 & 80 & 40 & 0 & 0 & 0 \\ 80 & 120 & 160 & 200 & 160 & 120 & 80 & 40 & 0 & 0 \\ 40 & 80 & 120 & 160 & 200 & 160 & 120 & 80 & 40 & 0 \\ 0 & 40 & 80 & 120 & 160 & 200 & 160 & 120 & 80 & 40 \\ 0 & 0 & 40 & 80 & 120 & 160 & 200 & 160 & 120 & 80 \\ 0 & 0 & 0 & 40 & 80 & 120 & 160 & 200 & 160 & 120 \\ 0 & 0 & 0 & 0 & 40 & 80 & 120 & 160 & 200 & 150 \\ 0 & 0 & 0 & 0 & 0 & 40 & 80 & 120 & 150 & 200 \end{bmatrix}$$

and the pre-R matrix obtained by applying to the rows and columns of R a random permutation

$$F = \begin{bmatrix} 200 & 0 & 0 & 150 & 120 & 0 & 160 & 40 & 0 & 80 \\ 0 & 200 & 150 & 0 & 0 & 120 & 0 & 80 & 160 & 40 \\ 0 & 150 & 200 & 0 & 0 & 80 & 0 & 40 & 120 & 0 \\ 150 & 0 & 0 & 200 & 80 & 0 & 120 & 0 & 0 & 40 \\ 120 & 0 & 0 & 80 & 200 & 80 & 160 & 120 & 40 & 160 \\ 0 & 120 & 80 & 0 & 80 & 200 & 40 & 160 & 160 & 120 \\ 160 & 0 & 0 & 120 & 160 & 40 & 200 & 80 & 0 & 120 \\ 40 & 80 & 40 & 0 & 120 & 160 & 80 & 200 & 120 & 160 \\ 0 & 160 & 120 & 0 & 40 & 160 & 0 & 120 & 200 & 80 \\ 80 & 40 & 0 & 40 & 160 & 120 & 120 & 160 & 80 & 200 \end{bmatrix}. \tag{2.4.4}$$

The PQ-tree T containing the solution of the reordering problem is constructed by calling the function `spectrsort`, which returns the resulting data structure:

```
T = spectrsort(F,opts)
T =
  struct with fields:
     type: 'Q'
    value: [1x10 struct]
```

Using the function `pqtreeplot`

```
pqtreeplot(T)
```

we obtain the representation of the PQ-tree displayed in Figure 2.2.

In this particular case, the PQ-tree T consists of just a Q-node as a root, so only two permutations of the leaves are allowed. They can be extracted from the tree using the function `pqtreeperms`, whose output is

```
perms_matrix = pqtreeperms(T)
perms_matrix =
     4     1     7     5    10     8     6     9     2     3
     3     2     9     6     8    10     5     7     1     4
```

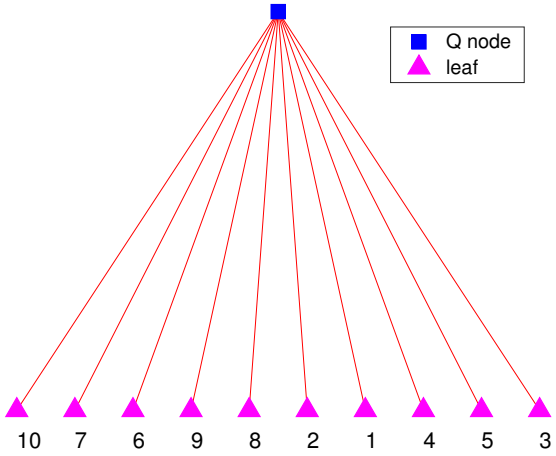


Figure 2.2: A PQ-tree corresponding to a pre-R matrix of dimension 10.

Sometimes, a PQ-tree may contain a very large number of permutations. In such cases, the function `pqtree1perm` extracts just one of the possible permutations, in order to apply it to the rows and columns of the matrix F:

```
seq = pqtree1perm(T);
AR = F(seq,seq);
```

Since F is pre-R, we clearly reconstruct the starting similarity matrix R.

2.4.2 Numerical experiments

In this section we illustrate the application of the `PQser` toolbox to some numerical examples. The experiments can be repeated by running the related Matlab scripts located in the `demo` sub-directory of the toolbox. We also compare the toolbox to other existing software.

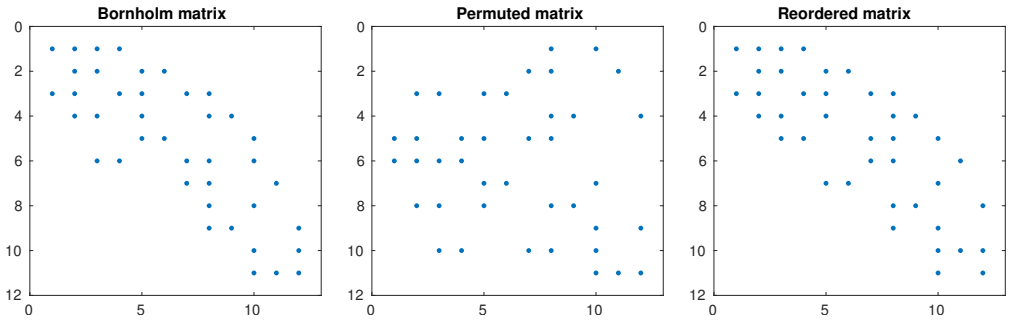


Figure 2.3: Processing of the Bornholm data set: the spy plot on the left shows the original matrix, a permuted version is reported in the central graph, and on the right, we display the matrix reordered by the spectral algorithm.

The first example is the numerical processing of the Bornholm data set, presented in Table 2.1. We randomly permute the rows of the adjacency matrix and apply the spectral algorithm to the similarity matrix on the rows/units, associated to the permuted adjacency

matrix. The resulting PQ-tree contains just a Q-node, so there is only one solution (actually, this is a proof that the matrix is pre-R) which we use to reorder the permuted matrix. The computational code is contained in the file `exper1.m`.

Figure 2.3 reports the spy plots which represent the nonzero entries of the initial matrix, its permuted version, and the final reordering. It is immediate to observe that the lower band of the reordered matrix is slightly narrower than that of the initial matrix, showing that the spectral algorithm was able to improve the results obtained empirically by archaeologists.

The second numerical experiment concerns the comparison between the sequential and parallel versions of the spectral algorithm in the solution of a large scale problem. The experiment was performed on a dual Xeon CPU E5-2620 system (12 cores), running the Debian GNU/Linux operating system and Matlab 9.2.

The function `testmatr` of the toolbox allows the user to create a block diagonal matrix formed by m banded blocks, whose size is chosen by using a second input parameter. The matrix is randomly permuted in order to hide its reducible structure.

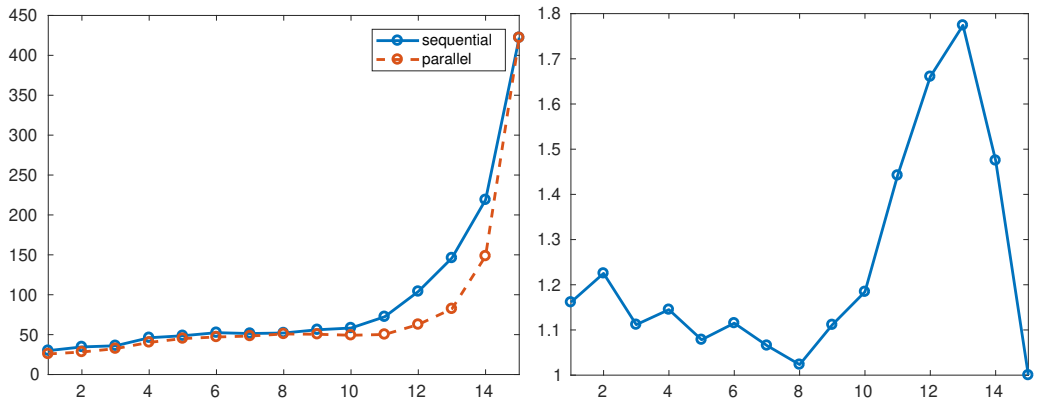


Figure 2.4: Comparison between the sequential and the parallel versions of Algorithm 3: on the left the execution time in seconds, on the right the parallel speedup, defined as the ratio between the sequential and the parallel timings. The test matrix is of dimension $2^{15} = 32768$, the size of each reducible block is 2^j, where j is the index reported in the horizontal axis.

We let the size of the problem be $n = 2^{15} = 32768$ and, for $j = 1, 2 \ldots, 15$, we generate a sequence of test adjacency matrices containing $n2^{-j}$ blocks, each of size 2^j.

We applied the function `spectrsort` that implements Algorithm 3 to the above problems, as well as its parallel version `pspectrsort`, and recorded the execution time; see the file `exper2.m`. The number of processing units available on our computer was 12.

The graph on the left of Figure 2.4 shows that there is a significant advantage from running the toolbox on a parallel computing system when the network associated to the problem is composed by a small number of large connected components. This is confirmed by the plot of the parallel speedup, that is, the ratio between the timings of the sequential and the parallel implementations, displayed in the graph on the right in the same figure.

As a third numerical example, we choose to compare the results of our implementation of the spectral algorithm with those provided by the `seriate` function (with `method="spectral"`), from the R package `seriation` [85] (`http://cran.r-project.org/web/packages/seriation/`).

We considered a well posed problem, represented by the pre-R similarity matrix (2.4.4). The `seriation` R package requires a slightly different input, that is, a *dissimilarity* matrix D rather than a similarity matrix F, but it is immediate to pass from one format to the other, by first computing

$$H = (h_{ij}) = \boldsymbol{\delta}\mathbf{e}^T + \mathbf{e}\boldsymbol{\delta}^T - 2F,$$

where $\boldsymbol{\delta} = \mathrm{diag}(F)$ and $\mathbf{e} = [1, \ldots, 1]^T$, and then setting $D = (d_{ij})$ with $d_{ij} = \sqrt{h_{ij}}$. The function `seriate` correctly returns the permutation

```
3     2     9     6     8     10     5     7     1     4,
```

i.e., one of the two symmetric permutations reported at the end of the previous paragraph.

We also applied `seriate` to the Bornholm data set of Table 2.1. In this case, the returned permutation leads to a matrix with a larger bandwidth than the reordered matrix displayed in Figure 2.3. The non-optimality of the solution is confirmed by the value of the function $\phi(L) = \mathbf{x}^T L\mathbf{x}$ (see (2.4.2)) minimized by the spectral algorithm. Indeed, $\phi(\tilde{L}) = 506$ and $\phi(L') = 655$, being $\tilde{L}$ and L' the Laplacians of the similarity matrices produced by `spectrsort` and `seriate`, respectively.

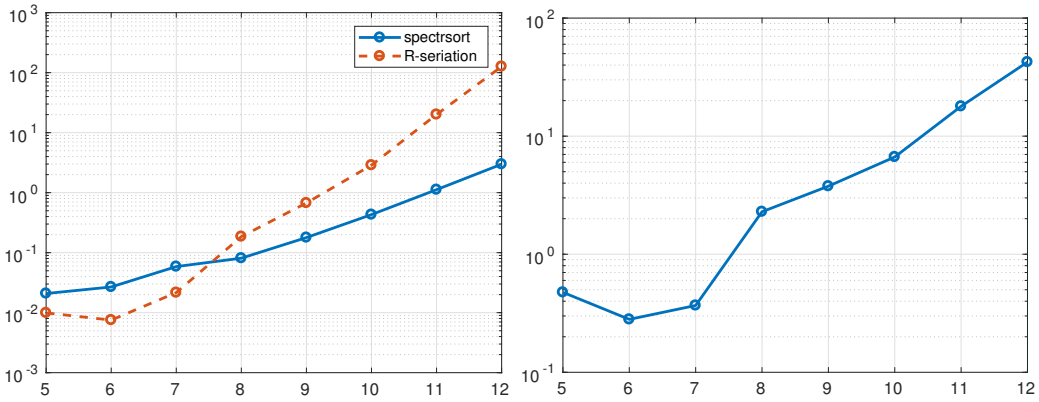


Figure 2.5: Comparison between the sequential version of Algorithm 3 and the function `seriate` from [85]: on the left the execution time in seconds, on the right the ratio between the timings of Algorithm 3 and of the function `seriate`. The size of the test matrix is 2^j, with $j = 5, \ldots, 12$. Each matrix is block diagonal, with 4 blocks of size 2^{j-2}.

The other comparison we considered in this section, investigates the execution time of our implementation of the spectral algorithm and the one included the R package. The test matrix is the same one used in the experiment depicted in Figure 2.4. In this case n, the size of the problem, ranges from 2^5 to 2^{12}; each of the test matrices is block diagonal, with 4 blocks each of size $n/4$. The graph on the left of Figure 2.5 shows the computing time for the two implementations. While the R package appears to be slightly faster for small scale problems, the sequential version of `spectrsort` is increasingly faster as the size of the problem gets large. The same trend can be observed in the graph on the right of the same figure, which depicts the ratio between the two timings. This may be due to the fact that Matlab is more likely to fully exploit the parallel processing capabilities of multi-core/multi-processor architectures.

To conclude, we consider another important application of the reordering defined by the Fiedler vector of the Laplacian, namely, the *reduction of the bandwidth* for a sparse matrix.

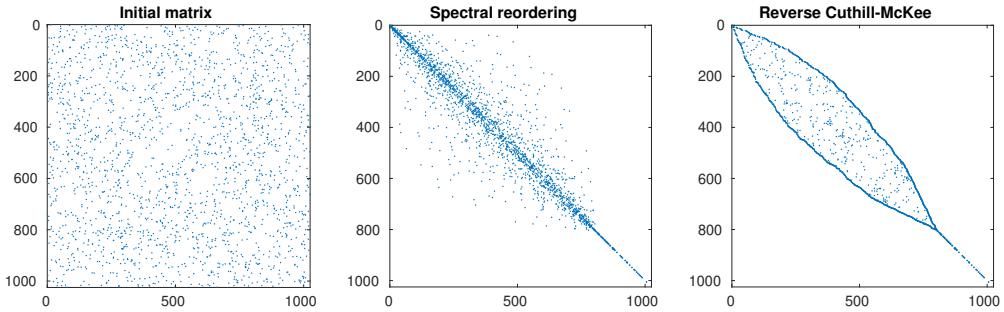


Figure 2.6: Bandwidth reduction of a sparse matrix of size 1024: the three spy plots display the initial matrix, the reordered matrix resulting from the spectral algorithm, and the one produced by the `symrcm` function of Matlab.

The spectral algorithm can be indeed used for computing an envelope-reducing ordering of sparse, symmetric matrices; see [12].

In our test, we generated with the Matlab function `sprandsym`, a sparse random symmetric matrix of size $n = 1024$, having approximately 0.2% nonzero elements. Notice that in this case the spectral algorithm must be applied to a matrix whose elements are taken in absolute value, so we applied the sequential version of Algorithm 3 to the test matrix with entries in absolute values. We then reorder it by using Algorithm 3 applying one of the obtained permutations to its rows and columns. The computation is described in the script `exper3.m` contained in the `demo` sub-directory.

The resulting matrix is depicted by displaying its nonzero pattern in Figure 2.6, where it is compared to the reverse Cuthill-McKee ordering, as implemented in the `symrcm` function of Matlab. The Cuthill-McKee algorithm is a variant of the standard breadth-first search algorithm used in graph algorithms. It is used to permute a sparse matrix that has a symmetric sparsity pattern into a band matrix form with a small bandwidth [43]. The reverse version renumbers the ordering obtained with the Cuthill-McKee but with the resulting index numbers reversed [77]. This modification, in general, yelds to superior performance efficiency than the original Cuthill-McKee ordering, although the bandwidth remains unchanged [125].

The spectral algorithm appears to be less effective than `symrcm`, leading to a reordered matrix with a wider band. This is due to the fact that `spectrsort` aims at placing the largest entries close to the diagonal, and this does not necessarily produce the maximal bandwidth reduction. Experiments with sparser matrices showed that quite often the two methods produce similar results.

2.5 Seriation in the presence of *imperfect* data

In this section we consider the case when the seriation data coming from a given problem, are *imperfect* and consequently the seriation problem is not well posed. Perfect data, analyzed in the previous section, leading to pre-P data matrices or pre-R similarity matrices, appear truly rarely in real situations. As a matter of fact, archeological or experimental data always contain errors, and then, in most applications, the data matrix A is typically noisy hence the pre-R assumption on the similarity matrix no longer holds. For that reason, the

similarity between the objects to be reordered can be only approximatively measured.

For imperfect data, the seriation problem was shown to be NP-complete in [78]. The spectral algorithm described in Section 2.4 is still applicable when the data deviate from being perfect within certain limits. Indeed, the spectral solution is stable when the magnitude of the noise remains within the spectral gap [73]. Even though the Fiedler vector can still be used as a heuristic to find an approximate solution to the seriation problem, there is no guarantee that it is optimal. Hence, devising efficient algorithms for dealing with imperfect data, permuting rows and/or columns of a non pre-P data matrix or non pre-R similarity matrix into the arrangement "closest" in some metric, to a P-form or an R-form respectively, is deemed of great importance theory- and application-wise.

Fogel et al. in [74], explicitly wrote seriation as an optimization problem by proving the equivalence between the seriation and the combinatorial 2-SUM problems. 2-SUM, see e.g. [78] and the references therein, is a quadratic minimization problem over permutations. They also proposed an approach based on convex relaxations for the 2-SUM problem, in order to solve matrix ordering and improve the robustness of seriation solutions in a noisy setting. Their results show in particular that 2-SUM is polynomially solvable for matrices coming from serial data. The 2-SUM problem is related to the NP-hard quadratic assignment problem (QAP) which is a combinatorial optimization problem introduced by Koopmans and Beckman [114] as a mathematical model for the facilities location of economic activities. Given n facilities and n locations, QAP involves a *flow* matrix A and a *distance* matrix B. Hence, in QAP(A, B) the entry A_{ij} of A represents the flow of activity between the facilities i and j, while the entry B_{ij} of the distance matrix, is the distance between the locations i and j. Laurent and Seminaroti, in [121], modelled the seriation problem as an instance of QAP(A, B), showing that if both matrices A and B are pre-R, one can find an explicit solution to QAP by using a *Robinsonian recognition algorithm* to find Robinson orderings of the two involved matrices. They also introduced a heuristic to solve the seriation problem, based on a generalization of the similarity-first-search (SFS) algorithm [120], by finding a "close" Robinson approximation of the original non-Robinson similarity matrix. In a couple of works [134, 135], Meidanis and collaborators proposed an extension of the PQ tree structure, the so-called *PQR tree*, which can point out possible obstructions when the C1P does not hold. These trees generalize the PQ tree of Booth and Lueker; indeed, a PQ tree is a PQR tree without R nodes whose presence indicates that the instance does not have the consecutive ones property. As an additional advantage, PQR trees can help in some cases to determine why the C1P is not valid pointing out specific sub-collections responsible for its failure [173].

Since in some of our experiments, when testing our implementation of the spectral algorithm, we observed the presence of a non simple Fiedler value of the Laplacian matrix associated to the considered problem, in the following paragraph we analyse this situation which highlights the presence of imperfect data.

2.5.1 The case of a multiple Fiedler value

As a particular case of the presence of imperfect seriation data, we discuss the situation where the Fiedler value is a multiple root of the characteristic polynomial of the Laplacian L. When this happens, the eigenspace corresponding to the smallest nonzero eigenvalue of L has dimension larger than one, so there is no uniqueness in the choice of the Fiedler vector, which can be any vector in the eigenspace.

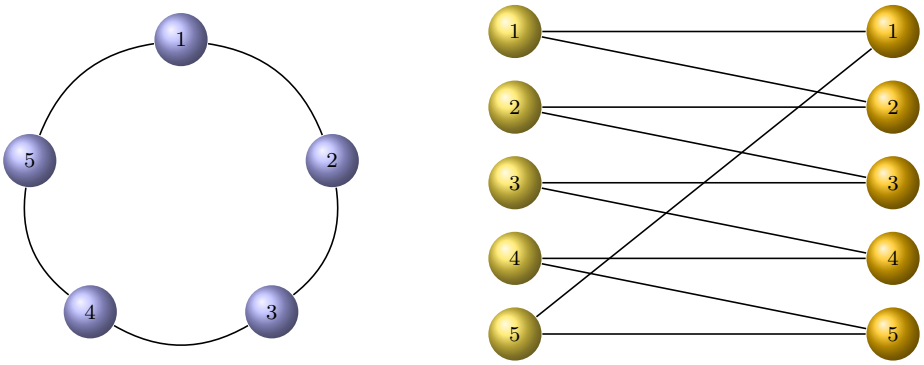


Figure 2.7: On the left: the cycle graph C_5. On the right: the bipartite graph related with the cycle graph C_5.

In [41], we conjecture that sorting the entries of one of the Fiedler vectors does not necessarily lead to all possible index permutations, i.e., the factorial of the number n of units. We experimentally observed that there may be some constraints that limit the number of permutations deriving from the Fiedler vector, and this number appears to be related to the structure of the eigenspace, and not simply to the multiplicity of the Fiedler value. We will illustrate this issue by numerical experiments.

Here we present a first simple example to justify our conjecture represented by the *cycle* or circular graph which is a graph, that we will indicate as C_n, whose n vertices are connected in a closed chain. The number of edges in C_n equals the number of nodes and every vertex has degree 2 since every node has exactly two edges incident to it. Hence a cycle is a *regular* graph, i.e., a graph in which each vertex has the same degree. The adjacency matrix A associated with C_n is given by

$$A = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 & 1 \\ 1 & 0 & 1 & 0 & \dots & 0 \\ 0 & 1 & \ddots & \ddots & \dots & \vdots \\ \vdots & \dots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & \dots & \ddots & \ddots & 1 \\ 1 & 0 & \dots & 0 & 1 & 0 \end{bmatrix}, \tag{2.5.1}$$

while the $n \times n$ signless incidence matrix

$$E = \begin{bmatrix} 1 & 1 & 0 & \dots & \dots & 0 \\ 0 & 1 & 1 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \dots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & \dots & 0 & 1 & 1 \\ 1 & 0 & \dots & \dots & 0 & 1 \end{bmatrix}, \tag{2.5.2}$$

can be interpreted as a data matrix of the considered seriation problem, that is, an element equal to 1 in position (i, j) indicates that unit i contains type j objects. This means that E is the upper-right part of the adjacency matrix of the related bipartite graph whose nodes

sets represent the archaeological units and types of the findings, as explained in Section 2.2; see Figure 2.7 for the representation of the cycle graph with $n = 5$ nodes and the related bipartite graph.

The similarity matrix $F = EE^T$ and its Laplacian L, that coincides with the Laplacian of A, are respectively

$$F = EE^T = \begin{bmatrix} 2 & 1 & 0 & \dots & 0 & 1 \\ 1 & 2 & 1 & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \ddots & \dots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & \ddots & \ddots & \ddots & 1 \\ 1 & 0 & \dots & 0 & 1 & 2 \end{bmatrix}, \qquad L = \begin{bmatrix} 2 & -1 & 0 & \dots & 0 & -1 \\ -1 & 2 & 1 & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \ddots & \dots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & \ddots & \ddots & \ddots & -1 \\ -1 & 0 & \dots & 0 & -1 & 2 \end{bmatrix}.$$

The matrix L, as well as F, is *circulant*, that is, it is fully specified by its first column, while the other columns are cyclic permutations of the first one with an offset equal to the column index [48]. A basic property of circulant matrices is that their spectrum is analytically known. Considering the following discrete Fourier transform of the first column of the Laplacian matrix L associated to the problem

$$\widehat{L}(\zeta) = 2 - \zeta^{-1} - \zeta^{-(n-1)}, \tag{2.5.3}$$

the spectrum of L is given by

$$\sigma(L) = \{\widehat{L}(1), \widehat{L}(\omega), \dots, \widehat{L}(\omega^{n-1})\} \tag{2.5.4}$$

where $\omega = \mathrm{e}^{\frac{2\pi \mathrm{i}}{n}}$ is the minimal phase nth root of unity, with i the imaginary unit.

The next results states the behavior of the eigenvalues of the Laplacian matrix in the special case of a circular graph C_n.

Theorem 2.5.1. *Let A be the adjacency matrix of a circular connected graph, with at least $n \geq 3$ vertices. Then the eigenvalues of the Laplacian matrix $L = D - A$ are coupled, that is*

$$\widehat{L}(\omega^k) = \widehat{L}(\omega^{n-k}), \qquad k = 1, \dots, n-1.$$

In particular, if n is odd, the smallest eigenvalues $\lambda_1 = 0$ is the only simple eigenvalue. Otherwise, the smallest eigenvalue $\lambda_1 = 0$ and the largest one λ_n are the only simple eigenvalues.

Proof. First, we recover a well known result in graph theory which states that the smallest eigenvalue of the Laplacian is $\lambda_1 = 0$. From (2.5.3) and (2.5.4), fixed $k = 0$, we obtain

$$\widehat{L}(1) = 2 - 1 - 1 = 0.$$

Next, let us consider $k = 1, \dots, n-1$. From (2.5.3) and (2.5.4) we obtain

$$\begin{aligned} \widehat{L}(\omega^k) &= 2 - e^{-\frac{2\pi i}{n}k} - e^{-\frac{2\pi i}{n}(n-1)k} \\ &= 2 - e^{-\frac{2\pi i}{n}k} - e^{\frac{2\pi i}{n}k} \\ &= 2 - \theta_k - \overline{\theta_k}, \qquad \text{where} \quad \theta_k = e^{-\frac{2\pi i}{n}k}. \end{aligned}$$

From this and the property $\omega^k = \overline{\omega^{n-k}}$, the thesis follows. □

Corollary 2.5.2. *Let A be the adjacency matrix of a graph satisfying the hypothesis of Theorem 2.5.1. Then the Fiedler value has multiplicity* 2.

Let us consider the cycle graph with $n = 5$ vertices and edges and the seriation problem described by the bipartite graph associated with C_5, depicted on the right of Figure 2.7. The nodes on the left represent the units, e.g., the excavation sites; the nodes on the right are the types, which may be seen as the archaeological findings. The relationships between units and types are represented by edges connecting the nodes.

This seriation problem is clearly unsolvable because each unit is related to surrounding units by a connection to a common type and the two extremal units are related to each other in the same way. At the same time, not all the units permutations are admissible. For example, one may argue that the permutation $\boldsymbol{\pi}_1 = (3, 4, 5, 1, 2)^T$ should be considered partially feasible, as it breaks only one of the constraints contained in the bipartite graph, while the ordering $\boldsymbol{\pi}_2 = (1, 4, 2, 5, 3)^T$ has nothing to do with the problem considered.

In the considered example, the Laplacian matrix L associated with the problem is given by

$$L = \begin{bmatrix} 2 & -1 & 0 & 0 & -1 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ -1 & 0 & 0 & -1 & 2 \end{bmatrix}$$

and hence, its eigenvalues are

$$\{\widehat{L}(1), \widehat{L}(\omega), \widehat{L}(\omega^2), \widehat{L}(\omega^3), \widehat{L}(\omega^4)\},$$

where $\widehat{L}(\zeta) = 2 - \zeta^{-1} - \zeta^{-4}$ is the discrete Fourier transform of the first column of L and $\omega = \mathrm{e}^{\frac{2\pi \mathbf{i}}{5}}$ is the minimal phase 5th root of unity. And a simple computation shows that

$$\widehat{L}(1) = 0, \quad \widehat{L}(\omega) = \widehat{L}(\omega^4) = 2 - 2\cos\frac{2\pi}{5}, \quad \widehat{L}(\omega^2) = \widehat{L}(\omega^3) = 2 - 2\cos\frac{4\pi}{5},$$

so that the Fiedler value $\widehat{L}(\omega)$ has multiplicity 2.

To explore this situation, we performed the following numerical experiment. We considered 10000 random linear combinations of an orthonormal basis for the eigenspace corresponding to the Fiedler value. This produces a set of random vectors belonging to a plane immersed in $\mathbb{R}^5$, which can all be considered as legitimate "Fiedler vectors".

Each vector is sorted, and the corresponding permutations of indexes are stored in the columns of a matrix. In the end, all the repeated permutations are removed. We obtain the following 10 permutations (displayed as columns), i.e., much less than the $5! = 120$ possible permutations,

```
perms =

     3     5     5     2     2     1     4     4     3     1
     2     1     4     1     3     5     3     5     4     2
     4     4     1     3     1     2     5     3     2     5
     1     2     3     5     4     4     2     1     5     3
     5     3     2     4     5     3     1     2     1     4
```

The obtained permutations reduce to 5 if we remove the permutations which are the reverse of another one. This confirms our conjecture: when a Fiedler value is a multiple eigenvalue, some constraints are imposed on the number of the admissible permutations of the units.

It is relevant to notice that the results produced by the above procedure do not contain the cyclic permutations of the units of the graph depicted in Figure 2.7. We can justify this outcome by comparing the initial matrix $F = AA^T$ to the matrix $\tilde{F}$, which results by reordering F according to the output of the experiment

$$F = \begin{bmatrix} 2 & 1 & 0 & 0 & 1 \\ 1 & 2 & 1 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 1 & 2 & 1 \\ 1 & 0 & 0 & 1 & 2 \end{bmatrix}, \qquad \tilde{F} = \begin{bmatrix} 2 & 1 & 1 & 0 & 0 \\ 1 & 2 & 0 & 1 & 0 \\ 1 & 0 & 2 & 0 & 1 \\ 0 & 1 & 0 & 2 & 1 \\ 0 & 0 & 1 & 1 & 2 \end{bmatrix}. \tag{2.5.5}$$

Indeed, the spectral algorithm aims at moving the nonzero components close to the main diagonal and this contrasts with the presence of non-zeros in the elements f_{51} and f_{15} of the matrix F, while $\tilde{F}$ exhibits a smaller bandwidth than F. We also run the `seriate` function from the R package `seriation` [85] on this ill posed problem, whose similarity matrix is F. The R package returns the same matrix $\tilde{F}$ constructed with our approach. We remark that `spectrsort` is not currently able to handle this "noisy" situation but, unlike `seriate`, it issues a warning about the presence of a multiple Fiedler value, signalling that the similarity matrix is not pre-R and hence the problem is not well posed.

Even in the application of the spectral algorithm to the reduction of the bandwidth of sparse, symmetric matrices, the presence of a multiple Fiedler value constitutes a problem. Indeed, for example, we were not able to correctly process the Matlab 60×60 test matrix `bucky` (representing the connectivity graph of the Buckminster Fuller geodesic dome), because the associated Laplacian possesses a Fiedler value with multiplicity 3.

Since we did not find any reference to the problem of a multiple Fiedler value in the literature, we are currently studying it, hence the rest of this paragraph is still a work in progress. This is the reason why the `PQser` toolbox conventionally associates an *M-node* to the presence of a multiple Fiedler value. In the software, the new type of node is temporarily treated as a P-node. This leaves the possibility to implement the correct treatment of this particular case of multiple Fiedler value, once the problem has been understood.

In order to investigate the particular situation when the Fiedler value is multiple we are currently developing two methods: a *graphic method* and a Montecarlo method based on repeated random samplings of linear combinations of an orthonormal basis for the eigenspace corresponding to the multiple Fiedler value. The graphic method works, up until now, only when the Fiedler value is double. Once the methods will be completed, we will study also the case of a Fiedler value with multiplicity larger than 2.

As an example of the admissible permutations obtained from the reordering of the nodes in a networks using the eigenvectors associated with a double Fiedler value, we considered a modification of the *star* graph. The star graph on n nodes, denoted here with $\mathcal{S}_n$, is a connected graph with n vertices and $n-1$ edges, where one vertex (the *central* one) has degree $n-1$ and the others $n-1$ vertices have degree 1. A star graph is a special case of a *complete* (i.e., every pair of distinct vertices is connected by an edge) bipartite graph in which one set has one vertex and the other set contains $n-1$ nodes.

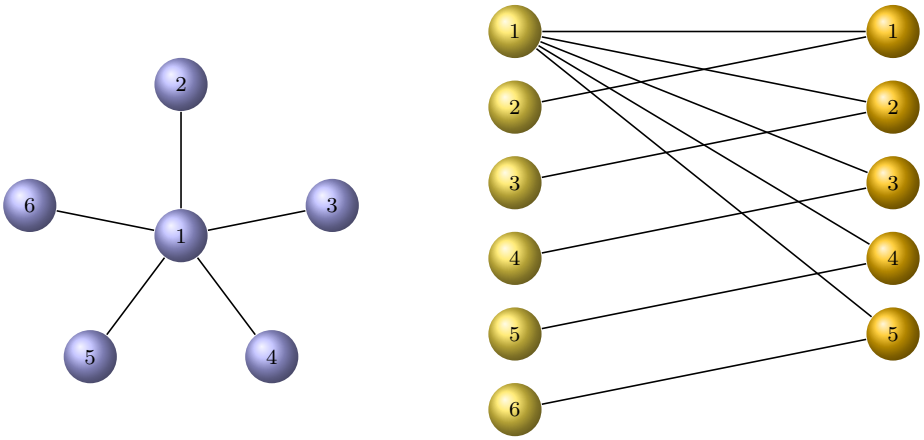


Figure 2.8: On the left: the star graph $\mathcal{S}_6$. On the right: the bipartite graph related with the star graph $\mathcal{S}_6$.

Without loss of generality we can assume that the first node is the center of the star so that the adjacency matrix is of the form

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 & \dots & 1 \\ 1 & 0 & 0 & 0 & \dots & 0 \\ 1 & 0 & 0 & 0 & \dots & 0 \\ 1 & 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & 0 & 0 & \dots & 0 \end{bmatrix}. \tag{2.5.6}$$

Following the same approach used previously in the description of the cycle graph, the upper-right part of the adjacency matrix of the bipartite graph related to the star graph depicted in Figure 2.8 (for $n = 6$ nodes)

$$E = \begin{bmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & 0 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & 0 & \dots & 0 \\ 0 & 0 & 1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \dots & \ddots & 0 \\ 0 & \dots & \dots & \dots & 0 & 1 \end{bmatrix} \in \mathbb{R}^{n \times (n-1)},$$

can be interpreted as data matrix of a considered seriation problem. The similarity and Laplacian matrices are respectively

$$F = \begin{bmatrix} n-1 & 1 & 1 & 1 & \dots & 1 \\ 1 & 1 & 0 & 0 & \dots & 0 \\ 1 & 0 & 1 & 0 & \dots & 0 \\ 1 & 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & 0 & 0 & \dots & 1 \end{bmatrix}, \quad L = \begin{bmatrix} n-1 & -1 & -1 & -1 & \dots & -1 \\ -1 & 1 & 0 & 0 & \dots & 0 \\ -1 & 0 & 1 & 0 & \dots & 0 \\ -1 & 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ -1 & 0 & 0 & 0 & \dots & 1 \end{bmatrix}. \tag{2.5.7}$$

The Laplacian matrix, as well as the similarity one, is an *arrowhead* matrix that is a real symmetric matrix which is zero except for its main diagonal, one row and one column. We can consider arrowhead matrices of the form

$$\begin{bmatrix} \alpha & \mathbf{z}^T \\ \mathbf{z} & B \end{bmatrix} \tag{2.5.8}$$

where α is a scalar, $\mathbf{z} = [z_1, \ldots, z_{n-1}]^T$ is a vector of length $n-1$ and B is a diagonal matrix of order $n-1$ with elements $b_1, \ldots, b_{n-1}$. From the Cauchy's interlacing theorem 1.1.4 for the eigenvalues of Hermitian matrices follows that its sorted eigenvalues interlace the sorted elements b_i of the diagonal matrix B. If $b_1 \geq b_2 \geq \cdots \geq b_{n-1}$ and if the eigenvalues λ_i for $i = 1, \ldots, n$ are sorted accordingly, then the following inequality holds

$$\lambda_1 \geq b_1 \geq \lambda_2 \geq b_2 \geq \cdots \geq \lambda_{n-1} \geq b_{n-1} \geq \lambda_n.$$

If $b_i = b_j$ for some $i \neq j$, the above inequality implies that b_i is an eigenvalue of the arrowhead matrix (2.5.8) considered.

Considering the Laplacian matrix (2.5.7) we can observe that it is an arrowhead matrix of the form (2.5.8) with $\alpha = n-1$, $\mathbf{z} = [-1, \ldots, -1]^T$ and $B = I_{n-1}$.

The following theorem explains the behaviour of the eigenvalues of the Laplacian matrix in the case of a star graph $\mathcal{S}_n$. We report the following results for completeness, even though we are interested in a particular graph obtainable from the star network, by adding edges.

Theorem 2.5.3. *Let A the adjacency matrix of a star graph $\mathcal{S}_n$. Then the spectrum of the Laplacian matrix $L = D - A$ consist of the three eigenvalues 0, 1 and n where the second one has multiplicity $n-2$.*

Proof. Let the Laplacian matrix (2.5.7) be given and consider a well known result which states that the smallest eigenvalue of the Laplacian is $\lambda_1 = 0$. From the Cauchy interlacing theorem applied to the matrices L and I follows that the sorted eigenvalues λ_i $(i = 1, \ldots, n)$ of L interlace the sorted element of the diagonal matrix I that is

$$\lambda_1 \leq 1 \leq \lambda_2 \leq 1 \leq \cdots \leq \lambda_{n-1} \leq 1 \leq \lambda_n.$$

As we saw before, it follows that 1 is an eigenvalue of L with multiplicity $n-2$. Moreover n belongs to the spectrum of L, in fact $\mathbf{v} = [-(n-1), 1, \ldots, 1]^T$ is a vector of length n for which

$$L\mathbf{v} = n\mathbf{v}.$$

□

Corollary 2.5.4. *Let A be an adjacency matrix of a graph satisfying the hypothesis of Theorem 2.5.3. Then the Fiedler value has multiplicity $n-2$.*

Corollary 2.5.5. *The $n-2$ Fiedler vectors of a star graph have null component in the position corresponding to the central node index.*

Proof. Without loss of generality we can assume that the first node is the central one of degree $n-1$. The coefficient matrix of the linear system associated with the Fiedler value is of the form

$$L - I = \begin{bmatrix} n-2 & -\mathbf{z}^T \\ -\mathbf{z} & 0 \end{bmatrix}.$$

The first equation of the system implies that the sum of components of the Fiedler vectors is 0, while the remaining $n-1$ show that their first component is always 0. □

Since we are interested in the case when the Fiedler value has multiplicity 2, let us consider a modified version of S_n obtained by adding $n-4$ edges of the form $\{u_i u_{i+1} | 2 \leq i \leq n-3\}$, to the adjacency matrix (2.5.6). The adjacency matrix of the *modified* star graph, denoted here by $\widehat{\mathcal{S}}_n$, having n nodes and $2n-5$ edges is of the form

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 & \cdots & \cdots & 1 \\ 1 & 0 & 1 & 0 & \cdots & \cdots & 0 \\ 1 & 1 & 0 & \ddots & \cdots & \cdots & \vdots \\ 1 & 0 & \ddots & 0 & 1 & 0 & \vdots \\ \vdots & \vdots & \vdots & 1 & \ddots & 0 & \vdots \\ \vdots & \vdots & \vdots & \cdots & 0 & \ddots & 0 \\ 1 & 0 & 0 & 0 & \cdots & 0 & 0 \end{bmatrix}. \tag{2.5.9}$$

The data matrix is given by the signless incidence matrix

$$E = \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 & 0 & \cdots & 0 \\ 1 & 0 & 0 & 0 & \cdots & 0 & \ddots & 0 & 0 \\ 0 & 1 & 0 & 0 & \cdots & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 & 0 & \ddots & 1 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots & 0 & 0 & 1 \\ \vdots & \vdots & \vdots & \ldots & \ddots & 0 & \vdots & 0 & 0 \\ 0 & \cdots & \cdots & \cdots & 0 & 1 & 0 & \cdots & 0 \end{bmatrix} \in \mathbb{R}^{n \times (2n-5)}, \tag{2.5.10}$$

that is again the block of the adjacency matrix of the bipartite graph, depicted in Figure 2.9, which describes the connections between units and types. From the data matrix we can compute the similarity matrix on the rows/units

$$F = EE^T = \begin{bmatrix} n-1 & 1 & \cdots & \cdots & \cdots & \cdots & \cdots & 1 \\ 1 & 2 & 1 & 0 & \cdots & \cdots & \cdots & 0 \\ \vdots & 1 & 3 & 1 & \ddots & \ddots & \ddots & 0 \\ \vdots & 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & 1 & 3 & 1 & 0 & \vdots \\ \vdots & \vdots & \ddots & \ddots & 1 & 2 & 1 & \vdots \\ \vdots & \vdots & \ddots & \ddots & \ddots & 0 & 1 & 0 \\ 1 & 0 & \cdots & \cdots & \cdots & \cdots & 0 & 1 \end{bmatrix},$$

while the Laplacian matrix of A, that coincides with the Laplacian of S, is given by

$$L = \begin{bmatrix} n-1 & -1 & \cdots & \cdots & \cdots & \cdots & \cdots & -1 \\ -1 & 2 & -1 & 0 & \cdots & \cdots & \cdots & 0 \\ \vdots & -1 & 3 & -1 & \ddots & \ddots & \ddots & 0 \\ \vdots & 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & -1 & 3 & -1 & 0 & \vdots \\ \vdots & \vdots & \ddots & \ddots & -1 & 2 & -1 & \vdots \\ \vdots & \vdots & \ddots & \ddots & \ddots & 0 & 1 & 0 \\ -1 & 0 & \cdots & \cdots & \cdots & \cdots & 0 & 1 \end{bmatrix}. \tag{2.5.11}$$

The following theorem explains the behaviour of the Fiedler value of the Laplacian matrix in the case of the *modified* star graph $\widehat{\mathcal{S}}_n$.

Theorem 2.5.6. *Let A be the adjacency matrix of a modified star graph $\widehat{\mathcal{S}}_n$. Then the Fiedler value is equal to 1 and has multiplicity 2.*

Proof. Without loss of generality we can assume that the first node is the central one of degree $n-1$. The added edges are of the form $\{u_i u_{i+1} | 2 \leq i \leq n-3\}$. We prove first that 1 is the Fiedler value. The Householder matrix associated with the eigenvector $v = [1, \ldots, 1]^T \in \mathbb{R}^n$ is given by

$$Q = \begin{bmatrix} \frac{1}{\sqrt{n}} & \cdots & \cdots & \cdots & \frac{1}{\sqrt{n}} \\ \vdots & \frac{1}{n}\left(\frac{\sqrt{n}}{1-\sqrt{n}}\right)+1 & \frac{1}{n}\left(\frac{\sqrt{n}}{1-\sqrt{n}}\right) & \cdots & \frac{1}{n}\left(\frac{\sqrt{n}}{1-\sqrt{n}}\right) \\ \vdots & \frac{1}{n}\left(\frac{\sqrt{n}}{1-\sqrt{n}}\right) & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \frac{1}{\sqrt{n}} & \frac{1}{n}\left(\frac{\sqrt{n}}{1-\sqrt{n}}\right) & \cdots & \cdots & \frac{1}{n}\left(\frac{\sqrt{n}}{1-\sqrt{n}}\right)+1 \end{bmatrix}.$$

The matrix $D_L = QLQ^T$, which is the deflated matrix with respect to the zero eigenvalue, has the form

$$D_L = \begin{bmatrix} 0 & \mathbf{0}^T \\ \mathbf{0} & \widetilde{L} \end{bmatrix}$$

where $\mathbf{0}$ is the null vector of length $n-1$ and

$$\widetilde{L} = \begin{bmatrix} 3 & 0 & 1 & \cdots & \cdots & \cdots & 1 \\ 0 & 4 & 0 & \ddots & \ddots & \ddots & 1 \\ 1 & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & 0 & 4 & 0 & 1 & \vdots \\ \vdots & \ddots & \ddots & 0 & 3 & 0 & \vdots \\ \vdots & \ddots & \ddots & \ddots & 1 & 2 & 1 \\ 1 & \cdots & \cdots & \cdots & \cdots & 1 & 2 \end{bmatrix} \in \mathbb{R}^{(n-1)\times(n-1)},$$

has the same eigenvalues of the matrix L. It is sufficient to show that the shifted matrix $\widetilde{L} - I_{n-1}$ is positive semi-definite. But this is clear noting the fact that

$$\widetilde{L} - I_{n-1} = \widehat{L} + \mathbf{e}\mathbf{e}^T, \qquad \mathbf{e} = [1, \cdots, 1]^T \in R^{(n-1)}$$

where $\widehat{L} = L(2:n, 2:n)$ and $\mathbf{e}\mathbf{e}^T$ are positive semi-definite matrices of order $n-1$.

The eigenspace associated with the eigenvalue 1 has dimension 2, in fact

$$\mathbf{v}_1 = [0, 1, \ldots, 1, -(n-2)]^T \quad \text{and} \quad \mathbf{v}_2 = [0, 1, \ldots, 1, -(n-3)]^T$$

are two linear independent vectors of length n for which

$$L\mathbf{v}_i = \mathbf{v}_i, \qquad i = 1, 2.$$

□

In the case of the derivation of the star graph obtained adding $n-4$ edges to its adjacency matrix, an orthogonal basis for the eigenspace $\mathcal{F}$ corresponding to the Fiedler value, of multiplicity 2, is given by

$$\mathbf{q_1} = [0, \underbrace{-1, \ldots, -1}_{n-3}, n-3, 0]^T, \quad \mathbf{q_2} = [0, \underbrace{-1, \ldots, -1}_{n-2}, n-2]^T. \tag{2.5.12}$$

For each $\mathbf{x} \in \mathcal{F}$, there is $\tilde{\mathbf{y}} \in \mathbb{R}^2$ such that $\mathbf{x} = Q_2\tilde{\mathbf{y}}$, where $Q_2 = [\mathbf{q}_1, \mathbf{q}_2]$. After immersing $\tilde{\mathbf{y}}$ in $\mathbb{R}^n$ by zero-padding, and completing the base Q_2, we can write $\mathbf{x} = Q\mathbf{y}$, with $\mathbf{y} \in \mathbb{R}^n$ given by

$$\mathbf{y} = \begin{bmatrix} \tilde{\mathbf{y}} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

Looking at $\mathbf{y}$ it is clear that any permutation of its last $n-2$ components is equivalent, and that only the first 2 entries are relevant in sorting.

Then, every $\mathbf{x} \in \mathcal{F}$ can be expressed as

$$\mathbf{x} = [0, \underbrace{-\alpha-\beta, \ldots, -\alpha-\beta}_{n-3}, (n-3)\alpha-\beta, (n-2)\beta]^T, \tag{2.5.13}$$

for $\alpha, \beta \in \mathbb{R}$. Since every vector in the eigenspace generated by the Fiedler vectors $\mathbf{q_1}$ and $\mathbf{q_2}$ contains $n-3$ coincident entries, the indexes associated with these components cannot be ordered uniquely, because they all have the same value. This means that in the associated PQ-tree they correspond to a P-node, originating $(n-3)!$ permutations. Specifically, the admissible permutations are related to the possible reorderings of the entries of $\mathbf{x}$ and these sortings depend on the values of the coefficients α and β in $\mathbf{x}$. Then, for sorting the entries of $\mathbf{x}$, we have to consider the following cases

1. $\alpha, \beta > 0$, with $\alpha < \beta$;
2. $\alpha, \beta > 0$, with $\beta < \alpha$;

3. $\alpha < 0 < \beta$;
4. $\beta < 0 < \alpha$;
5. $\alpha, \beta < 0$, with $\beta < \alpha$;
6. $\alpha, \beta < 0$, with $\alpha < \beta$.

In order to obtain only the admissible permutations ruling out the reverse ones, it can be observed that we can consider only the first three cases, since the cases 4,5,6 are coupled with 3,1,2 respectively, in the sense that they produce permutations which are the reverse of the ones obtained from the first three cases. Therefore, focusing only on the first three cases, we can compute the number of the admissible permutations. In case 1, when $\alpha, \beta > 0$ with $\alpha < \beta$, there are two subcases depending on the sign of the entry $(n-3)\alpha - \beta$ in $\mathbf{x}$

- when $0 < \alpha < \beta < (n-3)\alpha$, the ordering we find for the entries of $\mathbf{x}$ is $-\alpha - \beta < 0 < (n-3)\alpha - \beta < (n-2)\beta$ and then in this subcase there are $(n-3)!$ possible permutations given by the presence of a "macro" node of length $n-3$;
- when $0 < \alpha < (n-3)\alpha < \beta$ the ordering of the entries is $-\alpha - \beta < (n-3)\alpha - \beta < 0 < (n-2)\beta$ and, as in the previous subcase, there are again $(n-3)!$ admissible permutations.

When $\alpha, \beta > 0$ with $\beta < \alpha$ (case 2), the only possible reordering for the entries of $\mathbf{x}$ is $-\alpha - \beta < 0 < (n-2)\beta < (n-3)\alpha - \beta$ and then the admissible permutations are $(n-3)!$. In case 3, when $\alpha < 0 < \beta$, there are instead three possible subcases each one producing $(n-3)!$ admissible permutations due to the presence of the $(n-3)$ "macro" node given by the entry $-\alpha - \beta$

- when $\alpha < -(n-1)\beta < 0 < \beta$ the ordering of the entries in $\mathbf{x}$ is $(n-3)\alpha - \beta < 0 < (n-2)\beta < -\alpha - \beta$;
- when $-(n-1)\beta < \alpha < -\beta$ the sorting for the entries is $(n-3)\alpha - \beta < 0 < -\alpha - \beta < (n-2)\beta$;
- the last subcase is when $-\beta < \alpha$ and the only possible ordering for the entries is $(n-3)\alpha - \beta < -\alpha - \beta < 0 < (n-2)\beta$.

Hence, the admissible permutations for the nodes in the case of the graph $\widehat{S}_n$ are $3!(n-3)!$. This number is smaller than $n!$, that represents all the possible permutations for n indexes. Thus, again we can observe that also a double Fiedler value does not permit all the possible permutations, but imposes a constraint. The allowed permutations we found using the graphic method briefly explained above coincide with the ones obtained by using a Montecarlo method relying on repeated random sampling of linear combinations of $\mathbf{q_1}$ and $\mathbf{q_2}$ in (2.5.12). It can also be observed that in these admissible permutations node 1, that corresponds to the value 0 in $\mathbf{x}$, will never be in the first position in the resulting reorderings. In fact, considering the previous 6 cases for the values of α and β, at least one entry of $\mathbf{x}$ is negative.

If we consider the modified star graph $\mathcal{S}_6$ with $n = 6$ nodes, represented in Figure 2.9, the Fiedler vectors are

$$\mathbf{q}_1 = [0, -1, -1, -1, 3, 0]^T, \quad \mathbf{q_2} = [0, -1, -1, -1, -1, 4]^T$$

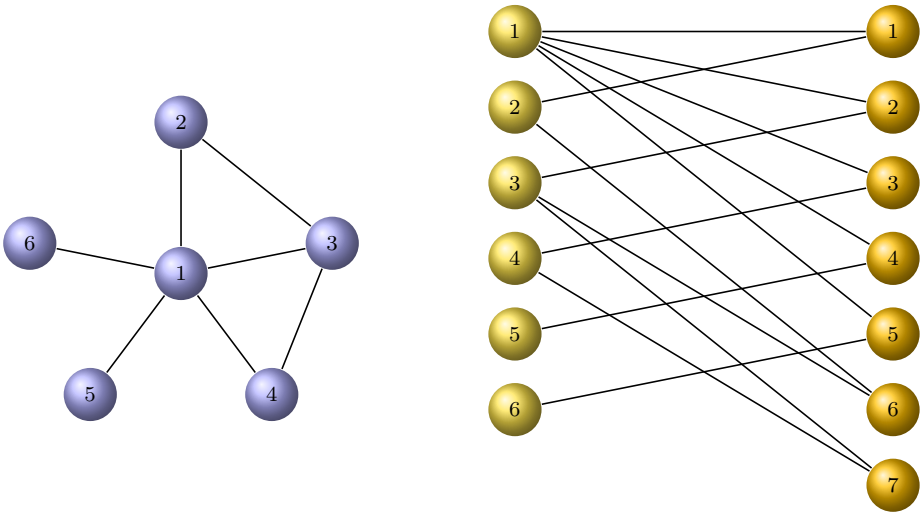


Figure 2.9: On the left: the star graph $\widehat{\mathcal{S}}_6$. On the right: the bipartite graph related with the star graph $\widehat{\mathcal{S}}_6$.

and any vector $\mathbf{x}$ in the eigenspace associated with the Fiedler value is of the form

$$\mathbf{x} = [0, -\alpha - \beta, -\alpha - \beta, -\alpha - \beta, 3\alpha - \beta, 4\beta]^T$$

for α and $\beta \in \mathbb{R}$. Following the above discussion on the reorderings of the components of $\mathbf{x}$ for each of the first 3 cases on the values of the coefficients α and $\beta \in \mathbb{R}$, we found $(n-3)! = 6$ admissible orderings and hence the allowed permutations of the nodes in the considered graph illustrating a seriation problem. Concluding, the admissible permutations, which we can found using both the graphic and the Montecarlo methods we are developing, are 36 (i.e. $3!(n-3)!$) and not $n! = 720$.

2.6 Conclusions and future work

In this Chapter we presented a new Matlab toolbox principally aimed to the solution of the seriation problem, but which can be applied to other related problems where one seeks to reconstruct a linear ordering based on unsorted and possibly noisy, pairwise similarity information.

Our software is based on a spectral algorithm introduced first in [12] for computing a reordering for envelope size reduction of sparse, symmetric matrices and then applied to the seriation and the consecutive ones problems by Atkins *et al.* [6]. The toolbox contains an implementation of PQ-trees as well as some tools for their manipulation, including an interactive visualization tool. The implemented spectral algorithm includes the possibility to choose between a small scale and a large scale algorithm for the computation of the Fiedler vector of the Laplacian associated to the considered seriation problem, and the software is compatible with Octave. Further, a parallel version of the spectral method is provided.

We also point out the importance of the presence of multiple Fiedler values, a problem which has not been considered before in the literature, and which has a significant influence on the computation of an approximate solution to the seriation problem in presence of imperfect seriation data.

The use of the toolbox has been illustrated by practical examples, and its performance is investigated through a set of numerical experiments, both of small and large scale.

Future work involves the study of seriation in a noisy setting, i.e. when the data matrix is not pre-P or the similarity matrix is not in Robinson's form. Some details have been given in the end of the previous Section regarding the noisy setting represented by the situation when the Laplacian of the similarity matrix associated with the considered seriation problem, has multiple Fiedler value.

Another approach we are considering, which is still a work in progress, aims at studying the seriation problem in the presence of imperfect data. This new line of research is based on investigating the asymptotic behaviour of a block matrix representing a bipartite graph associated to a given seriation problem, constructed from matrices obtained by properly normalizing the rows and columns of the given $(0, 1)$ seriation data matrix. Our purpose is to develop a spectral algorithm capable of dealing with imperfect seriation data, by using the same approach, which will be explained in Chapter 3, adopted for approximating a given network by a bipartite graph.

Chapter 3

A spectral method for "bipartizing" a network and detecting a large anti-community

In Chapter 1 we recalled some fundamental definitions of *graph* and *complex network theory*; see Section 1.3. All the systems, real or synthetic, consisting of entities that interact pairwise can be described in terms of networks. A network consists of nodes, which represent the entities of the system. Pairs of nodes are joined by links or edges describing a particular kind of interconnection between those items. Networks, as already mentioned in the first chapter, are represented by graphs $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{W}\}$, which are determined by a set of vertices (nodes) $\mathcal{V} = \{v_i\}_{i=1}^{n}$, a set of edges $\mathcal{E} = \{e_k\}_{k=1}^{m}$, and a set of positive weights $\mathcal{W} = \{w_k\}_{k=1}^{m}$. Here $e_k = (i_k, j_k)$ represents an edge from vertex v_{i_k} to vertex v_{j_k}. The weight w_k is associated with the edge e_k. Also in this chapter, we will consider connected undirected graphs without self-loops and multiple edges and, in particular, we will focus again on bipartite graphs. More specifically, we are interested in the *approximation of bipartite graphs*.

3.1 Bipartite graphs and bipartivity measures

A bipartite network involves objects that can be split into two disjoint groups with connections occurring only across, but not within, the two groups. Specifically, a network $\mathcal{G}$ is said to be bipartite if the set of vertices $\mathcal{V}$ that make up the graph can be partitioned into two disjoint nonempty subsets $\mathcal{V}_1$ and $\mathcal{V}_2$ (with $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2$), such that any edge starting at a vertex in $\mathcal{V}_1$ points to a vertex in $\mathcal{V}_2$, and vice versa. This, in particular, excludes the presence of self-loops in a bipartite graph.

Bipartite graphs model the interactions between two different types of objects. Many systems can be naturally modelled as bipartite networks, such as reaction or biochemical networks that model chemical reactions described by vertices representing chemical substances separated by nodes describing the chemical reactions. Another example is given by the "two-mode" networks, that appear frequently in sociology and economics, in which two disjoint sets of nodes are related by links representing the relationship between the elements of both classes; see [64]. As an example we can mention the citation networks in which a set of nodes gives the authors who cite (or are cited by) papers, which are represented as the

other set of nodes. A survey on mathematical properties and applications of bipartite graphs in the areas of algebra, combinatorics, chemistry, communication networks and computer science, was written by Asratian et al. [5].

There are various characterizations of bipartite graphs; the most widely used, obtained by König [113] is expressed by the following Theorem, a proof of which can be found in [54].

Theorem 3.1.1. *A graph G is bipartite if and only if it has no cycle of odd length.*

In [65] the authors provided another characterisation of bipartite networks given by the result below.

Theorem 3.1.2. *Let A be the adjacency matrix of a graph G. G is bipartite if and only if* $\operatorname{trace}\sinh(A) = 0$.

Bipartivity is an important topological property in graph theory. It has been studied also as the 2-coloring problem [20]. Indeed, bipartite networks are also known as *two-colourable graphs*, as we need only two colours, for example red and blue, to colour their nodes so that red nodes are connected only to blue nodes, and vice versa. Hence, determining if a graph can be colored with 2 colors is equivalent to determining whether or not the graph is bipartite, and thus testing if a network is bipartite or not is computable in linear time using breadth-first or depth-first search algorithms.

However, in many real-world situations the "perfect" separation into two classes, leading to exact bipartite networks, is not always possible and consequently many networks in real applications are only almost bipartite. Hence, it is therefore interesting to determine a bipartite approximation of a non-bipartite graph, or measure the distance of a non-bipartite graph from being bipartite.

In the literature there have been some attempts to characterise and quantify how much bipartivity a non-bipartite graph has. The pioneer work of Holme et al. proposed the first and more intuitive way of defining and measuring the bipartivity of a network [95]. This measure is obtained by accounting for the fraction of links that "destroy" the bipartivity in the network, whose deletion makes the graph perfectly bipartite. Such kind of links e_k, in the physics literature, are called "frustrated", due to the relation of this problem with the spin frustration in spin glasses; see [64]. In a bipartite network, all edges are "unfrustrated" links. Let m_{fr} be the number of links connecting nodes in the same subset and m the total number of links in the network, then the bipartivity of a network can be measured using the index

$$b = 1 - \frac{m_{fr}}{m}. \tag{3.1.1}$$

The aim consists in finding the best partition of the nodes in the network into two almost disjoint subsets in such a way that the smallest value of (3.1.1) for all possible partitions corresponds to the bipartivity of the considered network. Despite the simplicity of the method, the computation of this index is an NP-complete problem since in non-bipartite networks, computing the minimum number of edges whose deletion makes the graph bipartite is an NP-hard optimization problem; see [188]. Some approximate algorithms for computing this index have been reported in the literature. Holme and collaborators have themselves proposed one of such computational approximations for calculating the bipartivity of a network.

In 2005 Estrada and Rodríguez-Velázquez applied spectral graph theory to develop another characterisation of graph bipartivity [68]. This bipartivity measure is based on the *Estrada index* and on the concept of *closed walks*. The Estrada index of a graph G with adjacency matrix A, is define as

$$E(G) = \sum_{j=1}^{n} (\exp(A))_{jj} = \operatorname{trace}(\exp(A)),$$

and, by the properties of the trace of a matrix and the matrix exponential, it can be expressed as sum of the contributions deriving from odd and even closed walks, whose number is counted by the hyperbolic sine and hyperbolic cosine matrix functions respectively

$$E(G) = \operatorname{trace}(\sinh(A)) + \operatorname{trace}(\cosh(A)).$$

From the fact that a bipartite graph does not contain any odd-length cycle (see Theorem 3.1.1) and since every closed walk of odd length involves at least one odd cycle, it follows that a bipartite network is characterised by the absence of odd closed walks. Thus, G is bipartite if and only if

$$\operatorname{trace}(\exp(A)) = \operatorname{trace}(\cosh(A)).$$

Consequently, the measure of the network bipartivity is obtained by taking the proportion of even closed walks to the total number of closed walks

$$b = \frac{\operatorname{trace}(\cosh(A))}{\operatorname{trace}(\exp(A))} = \frac{\sum_{j=1}^{n} \cosh(\lambda_j)}{\sum_{j=1}^{n} \exp(\lambda_j)}, \tag{3.1.2}$$

where λ_j, for $j = 1, \ldots, n$, are the eigenvalues of A. It is evident that $b \leq 1$, with equality if and only if the network is bipartite.

The computation of the bipartivity index (3.1.2) is computationally complicated since it is defined in terms of the exponential and the hyperbolic cosine matrix functions. Hence, Estrada and Gómez-Gardeñes proposed a new approach for quantifying the bipartivity of a network. This measure, called *spectral bipartivity index* and obtained by considering only the exponential matrix function, is given by

$$b_s = \frac{\operatorname{trace}(\exp(-A))}{\operatorname{trace}(\exp(A))}. \tag{3.1.3}$$

We say that a splitting of the set of vertices $\mathcal{V}$ of a weighted undirected graph $\mathcal{G}$ into two disjoint nonempty subsets $\mathcal{V}_1$ and $\mathcal{V}_2$ (with $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2$), is a best bipartization of $\mathcal{G}$ if the sum of the weights w_k associated with edges $e_k = (i, j)$ that point from vertices v_i in $\mathcal{V}_\ell$ ($\ell = 1, 2$) to vertices v_j in the same set $\mathcal{V}_\ell$ is minimal. We remark that the above definition is analogous to the definition of a measure of bipartivity given by the spectral bipartivity index (3.1.3). This index also can be applied to the weighted graphs considered in the following.

The problem of discovering approximately bipartite structures in graphs and networks has been considered by various authors. Most popular approaches are based on the eigen-decomposition of the Laplacian matrix and in particular of the *signless Laplacian matrix*. Let A be the adjacency matrix of a graph G and D the diagonal degree matrix; the matrix $Q = D + A$ is the so-called signless Laplacian matrix. Even though the Laplacian matrix is studied extensively, the signless Laplacian matrix appears very rarely in the literature;

see [44] and references therein for a survey on this positive semi-definite matrix and its spectrum. The smallest eigenvalue of the signless Laplacian matrix, denoted here as $k(G)$, is related to the bipartivity of a network as the following result from [44] highlights.

Theorem 3.1.3. [Cvetković et al, 2007.] *The least eigenvalue of the signless Laplacian of a connected graph is equal to 0 if and only if the graph is bipartite. In this case 0 is a simple eigenvalue.*

The eigenvalue $k(G)$ has been studied in [51] as a measure of non-bipartiteness of a graph. Other spectral approaches consider the adjacency matrix associated with the graph. In the case of a symmetric bipartite adjacency matrix, the signs of the entries of an eigenvector associated with the smallest eigenvalue can be used to partition the graph, i.e., nodes that correspond to positive entries belong to one set, and nodes that correspond to negative entries belong to the other set; see [160]. In case the smallest eigenvalue is multiple, the splitting of the nodes may vary according to the considered vector in the associated eigenspace. In [172], the authors developed a spectral approach that can be used to identify hidden bipartite substructures. In particular, they presented a method for discovering approximately bipartite subnetworks in directed graphs. Some other attempts aiming at finding an approximate bipartite structure, have been applied to protein-protein interaction (PPI) networks, which model physical contacts between proteins in a cell and where the nodes are given by proteins while the edges denote that two proteins have been observed to interact physically. In [142], an algorithm was developed for finding bipartite substructure in PPI networks. The algorithm exploits the presence of $\pm$ pairs in the spectrum of the adjacency matrix of a bipartite graph (see Proposition 3.2.1 below) in order to identify approximate bipartite structures within PPI undirected networks and was shown to be robust in the presence of noise.

In this chapter, after recalling some properties of bipartite graphs and their spectral structure, we will discuss a numerical method developed for determining a "good" bipartization $(\mathcal{V}_1, \mathcal{V}_2)$, i.e., a bipartization for which the sum of the weights w_k associated with the edges $e_k = (i, j)$ that point from a vertex v_i in $\mathcal{V}_1$ to a vertex v_j in $\mathcal{V}_2$, or vice versa, is fairly small. Hence, we will describe how a given network can be approximated by a bipartite one by solving a sequence of fairly simple optimization problems. The spectral method also produces a node permutation which makes the possible bipartite nature of the initial adjacency matrix evident, and identifies the two sets of nodes. The algorithm has to be considered approximate, or "heuristic", in the sense that it does not necessarily produce the best possible bipartization. As it will be made clear in the following, the same bipartization method may be used also to detect the presence of a large anti-community in a given network and for its identification; see Section 3.4.

The contents of this chapter are based on our work [42].

3.2 The spectral structure of a bipartite graph and its approximation

This section discusses some properties of the adjacency matrix for an undirected bipartite graph focusing on the spectral structure of a bipartite graph. Some inequalities that are useful for the design of our bipartization method also will be introduced. The discussion in

the first part of the chapter assumes that the vertices are suitably *ordered* and subsequently, we will describe how to achieve such an ordering.

Assume for the moment that the undirected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{W}\}$ is bipartite, i.e., its vertex set $\mathcal{V}$ can be split into two disjoint nonempty subsets $\mathcal{V}_1$ and $\mathcal{V}_2$ with n_1 and n_2 nodes, respectively, such that there are no edges between the nodes in $\mathcal{V}_1$ and between the nodes in $\mathcal{V}_2$. We may assume that $n_1 \geq n_2$, otherwise we interchange the sets $\mathcal{V}_1$ and $\mathcal{V}_2$.

Let assume that the vertices in the set $\mathcal{V}$ are *ordered* so that the first n_1 of them belong to the set $\mathcal{V}_1$ and the remaining n_2 nodes belong to $\mathcal{V}_2$. Then the adjacency matrix for the graph $\mathcal{G}$ has the characteristic block structure

$$A_B = \begin{bmatrix} O_{n_1} & C \\ C^T & O_{n_2} \end{bmatrix}, \tag{3.2.1}$$

where O_k denotes the $k \times k$ zero matrix, and $C = [c_{i,j}] \in \mathbb{R}^{n_1 \times n_2}$ with $c_{i,j} > 0$ if the node v_i in $\mathcal{V}_1$ is connected to the node v_{n_1+j} in $\mathcal{V}_2$; otherwise $c_{i,j} = 0$.

An algebraic property of the adjacency matrix of a bipartite graph is given by the following well-known result, that is here adapted to our notation; see, e.g., [10, Theorem 3.14].

Proposition 3.2.1. *Let $\mathcal{G}$ be an unweighted graph with n nodes. Then $\mathcal{G}$ is bipartite and the adjacency matrix can be partitioned as in (3.2.1) if and only if the spectrum of the adjacency matrix is symmetric with respect to the origin, i.e.,*

$$\sigma(A_B) = \{\lambda_1, \ldots, \lambda_{n_2}, \underbrace{0, \ldots, 0}_{n_1 - n_2}, -\lambda_{n_2}, \ldots, -\lambda_1\}, \tag{3.2.2}$$

for some integers $n_1 \geq n_2$ and non-negative numbers $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_{n_2}$. The claim holds true also for weighted graphs, as long as the weights are positive.

Proof. For the sake of clarity, we give a quick sketch of the proof. The necessary condition is straightforward. The sufficient condition can be proved by noting that, for $k = 0, 1, \ldots$, $\mathrm{trace}(A_B^{2k+1}) = 0$ if the spectrum is symmetric. Then, the positivity of the weights implies that $(A_B^{2k+1})_{i,i} = 0$, that is, the graph is bipartite since it does not contain odd cycles. □

Remark 3.2.1. Under the assumption of Proposition 3.2.1, it is immediate to verify that if λ is a nonzero eigenvalue of A_B and $\mathbf{q} = \left[\begin{smallmatrix} \mathbf{x} \\ \mathbf{y} \end{smallmatrix}\right]$, with $\mathbf{x} \in \mathbb{R}^{n_1}$ and $\mathbf{y} \in \mathbb{R}^{n_2}$, is an associated eigenvector, then $(-\lambda, \left[\begin{smallmatrix} \mathbf{x} \\ -\mathbf{y} \end{smallmatrix}\right])$ is an eigenpair, too. This implies that λ is a singular value of the block C in (3.2.1), while $\mathbf{x}$ and $\mathbf{y}$ are its left and right singular vectors, respectively, if scaled to be of unit length.

Let $n = n_1 + n_2$ with $n_1 \geq n_2 \geq 1$. Then, the above observation gives us the possibility to describe the spectral structure of A_B in terms of the singular value decomposition of the block C which describes the connections in the graph; see also [83, Section 8.6.1]. Let $C = X\widetilde{D}Y^T$ be a singular value decomposition of C, where $\widetilde{D} \in \mathbb{R}^{n_1 \times n_2}$ has $D = \mathrm{diag}(\lambda_1, \ldots, \lambda_{n_2})$ as its upper block, and $X = [X_1, X_2] \in \mathbb{R}^{n_1 \times n_1}$ and $Y \in \mathbb{R}^{n_2 \times n_2}$ are orthogonal matrices with $X_1 \in \mathbb{R}^{n_1 \times n_2}$. Introduce the diagonal matrix

$$\mathcal{D} = \mathrm{diag}(D, O_{n_1 - n_2}, -D),$$

and the orthogonal matrix

$$Q = \begin{bmatrix} U_1 & U_2 & U_1 \\ V & O_{n_2, n_1 - n_2} & -V \end{bmatrix}, \tag{3.2.3}$$

where $U_1 = \frac{1}{\sqrt{2}}X_1$, $U_2 = X_2$, and $V = \frac{1}{\sqrt{2}}Y$, with $U_1^T U_1 = V^T V = \frac{1}{2}I_{n_2}$ and $U_2^T U_2 = \frac{1}{2}I_{n_1-n_2}$. Then, the spectral factorization

$$A_B = Q\mathcal{D}Q^T, \tag{3.2.4}$$

takes the form

$$\begin{bmatrix} U_1 & U_2 & U_1 \\ V & O_{n_2,n_1-n_2} & -V \end{bmatrix} \operatorname{diag}(D, O_{n_1-n_2}, -D) \begin{bmatrix} U_1 & U_2 & U_1 \\ V & O_{n_2,n_1-n_2} & -V \end{bmatrix}^T. \tag{3.2.5}$$

In the special case when the two nodes subsets have same cardinality $n_1 = n_2$, the submatrices of (3.2.3) with $n_1 - n_2$ columns disappear, and hence the spectral factorization (3.2.5) simplifies to

$$A_B = \begin{bmatrix} U_1 & U_1 \\ V & -V \end{bmatrix} \begin{bmatrix} D & 0 \\ 0 & -D \end{bmatrix} \begin{bmatrix} U_1 & U_1 \\ V & -V \end{bmatrix}^T,$$

with $U_1 U_1^T = VV^T = \frac{1}{2}I_{n_1}$.

3.2.1 Approximating the spectral structure of a bipartite graph

Let A be an adjacency matrix of an undirected graph. We would like to approximate the given graph by a bipartite one, and therefore seek to approximate A by a matrix of the form A_B taking into account the spectral decomposition, having the form (3.2.5), of the adjacency matrix of a bipartite graph. We do this in several steps and first we show some inequalities that are applicable to diagonal eigenvalue matrices and are useful for delineating how the eigenvalues of A_B can be approximated, given those of the starting adjacency matrix A.

Proposition 3.2.2. *Let $\alpha_1 \geq \alpha_2 \geq \cdots \geq \alpha_\ell$ be a nonincreasing real sequence and let $\beta_1, \beta_2, \ldots, \beta_\ell$ be another real sequence. The distance between these sequences measured in the least squares sense,*

$$\left(\sum_{i=1}^{\ell}(\alpha_i - \beta_i)^2\right)^{1/2}, \tag{3.2.6}$$

is minimal if and only if the β_i are in nonincreasing order, i.e., if $\beta_1 \geq \beta_2 \geq \cdots \geq \beta_\ell$.

Proof. Assume that both sequences are in nonincreasing order and that the distance can be reduced by changing the order of the β_i. Consider the pairs (α_1, β_1) and (α_2, β_2). Then

$$(\alpha_1 - \beta_2)^2 + (\alpha_2 - \beta_1)^2 \leq (\alpha_1 - \beta_1)^2 + (\alpha_2 - \beta_2)^2$$

is equivalent to

$$\alpha_2(\beta_1 - \beta_2) \geq \alpha_1(\beta_1 - \beta_2).$$

Assume $\beta_1 > \beta_2$. Then $\alpha_2 \geq \alpha_1$, which is a contradiction unless $\alpha_1 = \alpha_2$. If the β_j are ordered arbitrarily, then we can reorder these coefficients pairwise until they form a nonincreasing sequence. Each pairwise swap reduces (3.2.6). □

In our application of Proposition 3.2.2, we let $\alpha_1 \geq \alpha_2 \geq \cdots \geq \alpha_n$ be the eigenvalues of a given adjacency matrix A of order $n = n_1 + n_2$. The graph associated with this matrix might not be bipartite. We would like the sequence of eigenvalues of the matrix $A_B \in \mathbb{R}^{n\times n}$, given by (3.2.1), to be close to the sequence $\alpha_1, \alpha_2, \ldots, \alpha_n$ and appear in $\pm$ pairs. By Proposition 3.2.2, we know that the eigenvalues $\beta_1, \beta_2, \ldots, \beta_n$ of A_B should be in nonincreasing order, and by Proposition 3.2.1 they vanish or appear in $\pm$ pairs. We know from (3.2.5) that at least $n_1 - n_2$ eigenvalues of A_B should be zero and via the following Proposition we explain how to define the eigenvalues β_j of A_B that approximate the α_i of the given adjacency matrix A.

Proposition 3.2.3. *Let $\{\alpha_j\}_{j=1}^n$, with $n = n_1 + n_2$ and $n_1 \geq n_2$, be a real nonincreasing sequence. Then the sequence $\{\beta_j\}_{j=1}^n$ with elements*

$$\beta_j = \begin{cases} \frac{1}{2}(\alpha_j - \alpha_{n-j+1}), & j = 1, 2, \ldots, n_2, \\ 0, & j = n_2 + 1, \ldots, n_1, \\ -\beta_{n-j+1}, & j = n_1 + 1, \ldots, n, \end{cases} \tag{3.2.7}$$

is the closest sequence to $\{\alpha_j\}_{j=1}^n$ in the least squares sense consisting of at least $n_1 - n_2$ zeros and nonvanishing entries appearing in $\pm$ pairs.

Proof. The sequence $\{\beta_j\}_{j=1}^n$ consists of $n_1 - n_2$ zero values and n_2 $\pm$ pairs. Indeed, we have

$$\beta_j - \beta_{j+1} = \begin{cases} \frac{1}{2}(\alpha_j - \alpha_{j+1}) + \frac{1}{2}(\alpha_{n-j} - \alpha_{n-j+1}), & 1 \leq j \leq n_2 - 1, \\ \frac{1}{2}(\alpha_{n_2} - \alpha_{n_1+1}), & j = n_2, \\ 0, & n_2 + 1 \leq j \leq n_1 - 1, \\ \beta_{n_2}, & j = n_1, \\ \beta_{n-j} - \beta_{n-j+1}, & n_1 + 1 \leq j \leq n - 1, \end{cases}$$

and it follows that the sequence is nonincreasing. It remains to establish that the β_j defined by (3.2.7) are the best possible. Consider the minimization problems

$$\begin{cases} \min_\beta \Big((\alpha_j - \beta)^2 + (\alpha_{n-j+1} + \beta)^2 \Big), & 1 \leq j \leq n_2, \\ \min_\beta (\beta^2), & n_2 + 1 \leq j \leq n_1. \end{cases} \tag{3.2.8}$$

The solution sequence $\{\beta_j\}_{j=1}^n$ is given by (3.2.7). Thus, the β_j form a nonincreasing sequence consisting of $n_1 - n_2$ zero values and n_2 $\pm$ pairs. It is the closest such sequence to the sequence $\{\alpha_j\}_{j=1}^n$ in the sense that it solves the minimization problems (3.2.8). □

We would like to determine an approximation of the matrix A by a matrix of the form (3.2.1), where we allow row and column permutations of the latter matrix. Define the spectral factorization

$$A_B = W_B \Lambda_B W_B^T, \quad \Lambda_B = \operatorname{diag}(\lambda_1^{(B)}, \lambda_2^{(B)}, \ldots, \lambda_n^{(B)}),$$

where W_B is an orthogonal matrix and the eigenvalues are ordered according to

$$\lambda_1^{(B)} \geq \lambda_2^{(B)} \geq \cdots \geq \lambda_n^{(B)}.$$

We remark that only the first n_1 eigenvalues are ordered as in (3.2.4). In the following we will explain how to approximate the eigenvectors of A_B given an orthogonal eigenvector matrix of A.

Let us initially assume that the nonzero eigenvalues are distinct. If the eigenvectors are made unique, e.g., by making their first component positive, a comparison with (3.2.5) shows that

$$W_B = \begin{bmatrix} U_1 & U_2 & U_1 Z \\ V & O & -VZ \end{bmatrix}, \qquad \Lambda_B = \begin{bmatrix} D & O & O \\ O & O_{n_1-n_2} & O \\ O & O & -ZDZ \end{bmatrix}, \tag{3.2.9}$$

where Z is the *flip* matrix

$$Z = \begin{bmatrix} O & & 1 \\ & \iddots & \\ 1 & & O \end{bmatrix} \in \mathbb{R}^{n_2 \times n_2}.$$

In the presence of multiple nonzero eigenvalues, the corresponding eigenvectors are not uniquely determined, so the spectral factorization (3.2.9) is only one of several possible distinct factorizations.

Let

$$A = W \Lambda W^T, \quad \Lambda = \operatorname{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n), \tag{3.2.10}$$

be a spectral factorization of the adiacency matrix A of a given undirected graph, with an orthogonal eigenvector matrix W and the eigenvalues ordered according to

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n. \tag{3.2.11}$$

Let us partition the eigenvector matrix W conformally with the eigenvector matrix W_B of A_B, i.e.,

$$W = \begin{bmatrix} W_{11} & W_{12} & W_{13} \\ W_{21} & W_{22} & W_{23} \end{bmatrix}.$$

We would like to to approximate the eigenvector matrix W of A by the eigenvector matrix W_B of A_B. This suggests that we solve the minimization problem

$$\min_{\substack{U_1^T U_1 = V^T V = \frac{1}{2} I_{n_2} \\ U_2^T U_2 = \frac{1}{2} I_{n_1 - n_2}}} \left\| \begin{bmatrix} U_1 & U_2 & U_1 \\ V & O & -V \end{bmatrix} - \begin{bmatrix} W_{11} & W_{12} & W_{13} Z \\ W_{21} & W_{22} & W_{23} Z \end{bmatrix} \right\|_F, \tag{3.2.12}$$

where $\|\cdot\|_F$ denotes the Frobenius norm. This problem can be split into the three independent problems

$$\min_{U_1^T U_1 = \frac{1}{2} I_{n_2}} \{\|U_1 - W_{11}\|_F^2 + \|U_1 - W_{13} Z\|_F^2\}, \tag{3.2.13}$$

$$\min_{V^T V = \frac{1}{2} I_{n_2}} \{\|V - W_{21}\|_F^2 + \|V + W_{23} Z\|_F^2\}, \tag{3.2.14}$$

$$\min_{U_2^T U_2 = \frac{1}{2} I_{n_1 - n_2}} \{\|U_2 - W_{12}\|_F^2\}. \tag{3.2.15}$$

Problem (3.2.13) can be written as

$$\min_{X_1^T X_1 = I_{n_2}} \left\{ \left\| X_1 - \sqrt{2} W_{11} \right\|_F^2 + \left\| X_1 - \sqrt{2} W_{13} Z \right\|_F^2 \right\}. \tag{3.2.16}$$

The following result shows how we can easily solve this minimization problem in the Frobenius norm.

Proposition 3.2.4. *The solution of problem* (3.2.16) *can be determined by computing the singular value decomposition of* $W_{11} + W_{13}Z$ *and setting all singular values to one.*

Proof. Consider the problem

$$\min_{X^T X = I} \|X - W\|_F^2.$$

It can be written as

$$\min_{X^T X = I} \{\operatorname{trace}(X^T X) - 2\operatorname{trace}(X^T W) + \operatorname{trace}(W^T W)\}.$$

The first and last terms are independent of X. Therefore we obtain the equivalent linear minimization problem

$$\min_{X^T X = I} \{-\operatorname{trace}(X^T W)\}.$$

Similarly, the linear problem associated to the minimization problem (3.2.16) is given by

$$\min_{X_1^T X_1 = I_{n_2}} \{-\operatorname{trace}(X_1^T (W_{11} + W_{13} Z))\}. \tag{3.2.17}$$

Hence, the problem (3.2.16) is equivalent to determining the closest orthogonal matrix in the Frobenius norm to the matrix $W_{11} + W_{13}Z$. The solution is given by computing the singular value decomposition $P\Sigma Q^T$ of $W_{11} + W_{13}Z$ and setting $X_1 = PQ^T$; see [91, Theorem 4.1] for a proof of the latter statement. □

The other two minimization problems (3.2.14) and (3.2.15) are solved similarly. In this way we can obtain the eigenvector matrix in the spectral factorization (3.2.5) of A_B.

Remark 3.2.2. We note that if $P\Sigma Q^T$ denotes the singular value decomposition of $W_{11} + W_{13}Z$, then we can express its polar decomposition by

$$W_{11} + W_{13} Z = (PQ^T)(Q\Sigma Q^T).$$

Since the first factor PQ^T is the minimizer of (3.2.17), the deviation of $Q\Sigma Q^T$ from the identity matrix measures the quality of the approximation.

Remark 3.2.3. If some of the nonzero eigenvalues of A in (3.2.10) are multiple, the corresponding columns of W_{11}, W_{21}, W_{13}, and W_{23}, are not uniquely determined. Anyway, when approximating $W_{11} + W_{13}Z$ by X_1, and $W_{21} - W_{23}Z$ by Y, those columns contain linear combinations of the previous ones, and so they belong to the same space. Then, the approximations X_1 and Y will make factorization (3.2.9) valid.

3.3 A spectral bipartization method

In this Section we give an outline of a *spectral bipartization method*, based on the results presented above. Our technique exploits the spectral structure (3.2.5) of a bipartite graph to determine a node permutation that separates the two sets $\mathcal{V}_1$ and $\mathcal{V}_2$, and to construct a *bipartite approximation* to a connected undirected graph $\mathcal{G}$, having a perturbed bipartite structure. The bipartization algorithm is exact whenever the input is the adjacency matrix of a bipartite graph, however it has to be considered "heuristic", as we were not able to prove a complete convergence result for it, apart from the spectrum approximation theorems in Section 3.2.

Let A be the adjacency matrix of a connected undirected graph $\mathcal{G}$, and assume that its spectral factorization

$$A = W\mathcal{D}W^T, \quad \mathcal{D} = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n),$$

is available, where W is an orthogonal matrix and the eigenvalues are ordered by increasing absolute value. There are three problems to deal with. First of all we have to estimate the cardinality of the sets $\mathcal{V}_1$ and $\mathcal{V}_2$, then to suitably order the nodes in $\mathcal{G}$, and the last step consists in approximating the starting adjacency matrix by a matrix of the form (3.2.1). This three stages in our procedure are briefly examined as follows.

1. The first step of our algorithm consists of finding the cardinalities of the two disjoint node sets $\mathcal{V}_1$ and $\mathcal{V}_2$, i.e. the integers n_1 and n_2, unless they are known in advance. We do this by identifying the number of eigenvalues that are approximately zero. In principle, this could be done by detecting how many eigenvalues have absolute value larger than a fixed tolerance, but this process is extremely sensitive to the choice of the tolerance. In our numerical experimentation, we found it to be more reliable to detect the largest gap between "small" and "large" eigenvalues.

   To do this, we compute the ratios

   $$\rho_i = \frac{|\lambda_{i+1}|}{|\lambda_i|}, \quad i = 1, 2, \ldots, n-1. \tag{3.3.1}$$

   Then, for suitably chosen constants R and τ, we consider the index set

   $$\mathcal{J} = \{i \in \{1, 2, \ldots, n-1\} : \ \rho_i > R \text{ and } |\lambda_{i+1}| > \tau\}. \tag{3.3.2}$$

   In our experiments, we set $R = 10^2$ and $\tau = 10^{-8}$. Therefore, an index i is in $\mathcal{J}$ if there is a significant gap between λ_i and λ_{i+1} (i.e. if $\rho_i > R$), and λ_{i+1} is numerically nonzero (i.e. if $|\lambda_{i+1}| > \tau$). If the set $\mathcal{J}$ is empty, then we are not able to identify a partition of the nodes, and consequently we consider the cardinality of the sets $\mathcal{V}_1$ and $\mathcal{V}_2$ to be the same. On the contrary, we let k be the index defined by

   $$\rho_k = \max_{i \in \mathcal{J}} \rho_i,$$

   and set

   $$n_2 = \left\lceil \frac{n-k}{2} \right\rfloor, \quad n_1 = n - n_2,$$

   where $\lceil x \rfloor$ denotes the closest integer to the real number x.

The above approach is clearly not completely robust. Indeed, it can be easily tricked by constructing particular numerical examples, for example by letting C in (3.2.1) have singular values that decay to zero exponentially, or by introducing large gaps in the spectrum of the adjacency matrix. Nevertheless, we found the procedure quite accurate on networks stemming from real-world applications; see, e.g., Figures 3.6 and 3.8 in Section 3.5.

In order to avoid overflow, it may be preferable to use the reciprocal ratios ρ_i^{-1}. This is not required in our Matlab implementation, given the features of the programming language.

2. The subsequent step is to find the sets $\mathcal{V}_1$ and $\mathcal{V}_2$, and reorder the nodes. Assume that $\mathcal{G}$ is bipartite, but that its adjacency matrix A corresponds to a random ordering of the nodes, so that

$$A = \Pi A_B \Pi^T,$$

for a permutation matrix Π and a matrix A_B of the form (3.2.1). Obviously, the structure (3.2.1) is lost and in this case the spectral factorization (3.2.4) becomes

$$A = (\Pi Q)\mathcal{D}(\Pi Q)^T,$$

i.e., the rows of the eigenvector matrix are permuted. In order to recover the structure of the eigenvectors, let us partition the eigenvector matrix as

$$W := \Pi Q = \begin{bmatrix} W_1 & W_2 & W_3 \end{bmatrix}, \tag{3.3.3}$$

with $W_1, W_3 \in \mathbb{R}^{n\times n_2}$ and $W_2 \in \mathbb{R}^{n\times(n_1-n_2)}$.

Suppose first that $n_1 > n_2 \geq 1$ and consider the matrix block W_2. For (3.2.9) to be valid, the last n_2 rows of W_2 must vanish. Sorting in descending order the 1-norms of its rows concentrates the smallest entries in the lower block of W_2. Applying the corresponding permutation σ to the rows of W brings this matrix to the form (3.2.9) and the adjacency matrix to the form (3.2.1), with the block C possibly permuted.

When $n_1 = n_2$ the block W_2 is empty, so we consider the matrix $W_1 - W_3 Z$. As its first n_1 rows should be exactly zero, we sort the 1-norms of its rows in ascending order, and apply the obtained permutation σ to the rows of W.

After the reordering, the first n_1 nodes are in the set $\mathcal{V}_1$, and the remaining n_2 are contained in the set $\mathcal{V}_2$. We note that applying the permutation σ to the rows and columns of the initial adjacency matrix A highlights the presence in the graph of an approximate bipartite structure.

3. The last step of our procedure consists in finally obtaining an approximation of the adjacency matrix (3.2.1) of a bipartite graph from the computed spectral factorization of the given adjacency matrix A. To do this we first approximate the eigenvector matrix W_B by solving the minimization problem (3.2.12), and then approximate the eigenvalues in the spectral factorization (3.2.10) of the starting adjacency matrix A of a given undirected graph, by scalars that appear in $\pm$ pairs by using Proposition 3.2.3. Specifically, we let the α_j in the proposition be the eigenvalues (3.2.11) of the adjacency matrix A The β_j defined in the proposition are the eigenvalues of the matrix D in (3.2.5), in the same order.

Algorithm 6 Spectral bipartization algorithm.

Require adjacency matrix A of size n, the user may optionally provide the cardinalities n_1 and n_2 of $\mathcal{V}_1$ and $\mathcal{V}_2$
Ensure permutation σ which reorders the nodes, adjacency matrix A_B of the approximate bipartite graph
compute the spectral factorization $A = W\mathcal{D}W^T$, with $\lambda_1 \geq \cdots \geq \lambda_n$
// Step 1 of the algorithm
if n_1, n_2 are not provided
    sort the eigenvalues by increasing absolute value
    compute ρ_i, $i = 1, \ldots, n-1$ by (3.3.1)
    construct set $\mathcal{J}$ by (3.3.2)
    if $\mathcal{J} = \emptyset$
        $n_1 = \lceil n/2 \rceil$, $n_2 = n - n_1$
    else
        $k = \arg\max_{i \in \mathcal{J}} \rho_i$
        $n_2 = \lceil (n-k)/2 \rceil$, $n_1 = n - n_2$
    end if
end if
// Step 2 of the algorithm
partition $W = \begin{bmatrix} W_1 & W_2 & W_3 \end{bmatrix}$ as in (3.3.3)
if $n_1 > n_2$
    find the permutation σ which sorts the 1-norms of the rows of W_2 decreasingly
else
    find the permutation σ which sorts the 1-norms of the rows of $W_1 - W_3 Z$ increasingly
end if
apply the permutation σ to the rows of W
// Step 3 of the algorithm
approximate the eigenvectors of A by minimizing (3.2.12)
approximate the eigenvalues of A by $\pm$ pairs β_i, by Proposition 3.2.3
set $D = \mathrm{diag}(\beta_1, \ldots, \beta_n)$
construct the adjacency matrix $A_B = WDW^T$ of the bipartite graph

The above procedure, outlined in Algorithm 6, determines the eigenvectors and eigenvalues of a matrix A_B with the block structure

$$A_B = \begin{bmatrix} O & C \\ C^T & O \end{bmatrix}, \tag{3.3.4}$$

where the matrix C has real entries. The matrix A_B may have a different number of nonzero entries than A. In fact, not all nonzero entries may be positive. We can handle this issue in the following ways:

- allow A_B to be an adjacency matrix for a weighted graph with both positive and negative weights;
- allow A_B to be an adjacency matrix for a weighted graph with positive weights. We

achieve this purpose by replacing the matrix C in (3.3.4) by the closest matrix, C_+, in the Frobenius norm with nonnegative entries. The matrix C_+ is obtained from C by setting all negative entries to zero;

- require A_B to represent an unweighted graph. The closest such matrix in the Frobenius norm to the matrix (3.3.4), is obtained by setting every entry of C to the closest member of the set $\{0, 1\}$.

The last approach is the one adopted in the numerical experiments presented in Section 3.5.

Algorithm 6 can be applied only to small to medium sized problems, i.e., when it is possible to compute a full spectral factorization of the given adjacency matrix A. For larger problems, one may reduce the complexity of the computation by renouncing the third step of the algorithm. Indeed, when $n_1 - n_2$ is not too large, a partial spectral factorization may lead to constructing a basis for the null space of A, that is, to obtaining the matrix W_2. This would allow one to generate the permutation σ that takes the adjacency matrix to an almost bipartite form, identifying the two sets $\mathcal{V}_1$ and $\mathcal{V}_2$.

3.4 Anti-communities

In this Section we consider an application of our bipartization method, that is the *detection of a large anti-community*.

In many real-world networks, nodes frequently congregate together forming densely connected groups which are poorly connected with other parts of the network or isolated from them. This clusters are known as *communities*. For instance, in a social network where edges represent friendship, nodes belong to groups formed by friends and inside these groups it may be possible to find a relatively high density of connections but in many cases these clusters are poorly connected to others groups in society. Communities may also form because of similarities among the vertices. For instance, groups of proteins having similar functions in a PPI network can be more highly connected among themselves than with proteins which have different roles. An *anti-community* is, conversely, a node set that is loosely connected internally, but has many external connections with the rest of the network [67]. Community and anti-community detection in networks is a relevant problem with applications in various fields, including physics, computer science, natural and social sciences. Several methods have been developed to identify this kind of structures in networks and some of them allow communities to overlap with each other; see [35, 122, 149, 156, 187]. In [69] a spectral method is used to detect simultaneously communities and anti-communities. The identification of communities and their counterparts is predominant in the investigation of the so-called meso-scale structures which represent middle-scale properties in networks and differ from micro-scale and macro-scale structures which involve local and global information respectively. Although considerable efforts have gone into the study of community and anti-community structures, the detection of the so-called *core-periphery* structures, attracts a continuing interest also in the mathematical community. In [159] the authors presented a method for computing different possible cores and for identifying multiple cores. The most popular notion of core-periphery structures was developed by Borgatti and Everett [22] who proposed also a quantitative method to depict them. By identifying a network's core-periphery structure, one attempts to find which nodes are part of a densely connected core and which are instead part of a sparsely connected periphery.

Then, in contrast to communities, the nodes in a core-perifery are also reasonably well-connected to those in the periphery. Hence, for our purposes, the identification of a single large anti-community can be understood as that of a core-periphery structure in the given network.

Let us consider a symmetric matrix A of size $n = n_1 + n_2$ with a zero leading square block of size n_1. Then, A may be considered the adjacency matrix of a network with an anti-community of n_1 nodes which is a nodes set poorly connected internally, but with many external connections. Then, the matrix A has the block form

$$A = \begin{bmatrix} O_{n_1} & C \\ C^T & B \end{bmatrix}, \tag{3.4.1}$$

with C of size $n_1 \times n_2$ and B a square matrix of order n_2.

In the following, we denote by $\mathcal{N}(C)$ the null space of C, by $\mathcal{R}(C)$ its range, and by $B_{|\mathcal{N}(C)}$ the restriction of the submatrix B to $\mathcal{N}(C)$.

The succeeding result allows us to delineate the structure of the spectral factorization of the adjacency matrix of a graph with a large anti-community.

Theorem 3.4.1. *Let A be as in (3.4.1) and let $\mathbf{x} = \left[\begin{smallmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{smallmatrix}\right]$ be partitioned consistently with A. Then the equation*

$$A\mathbf{x} = \mathbf{0} \tag{3.4.2}$$

has $\nu = \dim \mathcal{N}(C^T)$ linearly independent solutions with $\mathbf{x}_2 = 0$. Moreover, if

$$d = \dim \left(\mathcal{R}(B_{|\mathcal{N}(C)}) \cap \mathcal{R}(C^T) \right) \geq 1,$$

then there are also d linearly independent solutions to $A\mathbf{x} = \mathbf{0}$ with $\mathbf{x}_2 \neq 0$, so that $\dim \mathcal{N}(A) = d + \nu$.

Proof. Let $k = \operatorname{rank}(C)$ and consider the case $n_1 > n_2 = k$. Let us search for vectors $\mathbf{x}$ such that $A\mathbf{x} = \mathbf{0}$. Then we have

$$A \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} = \begin{bmatrix} O_{n_1} & C \\ C^T & B \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} = \begin{bmatrix} C\mathbf{x}_2 \\ C^T\mathbf{x}_1 + B\mathbf{x}_2 \end{bmatrix}. \tag{3.4.3}$$

Since C is of full rank and $n_1 > n_2$, it follows from $C\mathbf{x}_2 = 0$ that $\mathbf{x}_2 = 0$ and, hence, $C^T\mathbf{x}_1 = 0$. The latter implies that $\mathbf{x}_1$ is in the null space of C^T, which has dimension $n_1 - n_2$. Thus, the matrix A admits the following linearly independent eigenvectors corresponding to the eigenvalue $\lambda = 0$,

$$\mathbf{x}^{(i)} = \begin{bmatrix} \mathbf{u}_{n_2+i} \\ \mathbf{0} \end{bmatrix}, \quad i = 1, 2, \ldots, n_1 - n_2,$$

where $\mathbf{u}_i$, $i = 1, 2, \ldots, n_1$, are the left singular vectors of C. Hence, $\lambda = 0$ has multiplicity $n_1 - n_2 = \dim \mathcal{N}(C^T)$.

Let us now assume that $k = n_1 < n_2$. Then A may or may not have zero eigenvalues. Indeed, for A to have a vanishing eigenvalue, the vector $\mathbf{x}_2 \in \mathbb{R}^{n_2}$ that appears in (3.4.3) has to belong to the null space of C, which has dimension $n_2 - n_1$. Then, there will be zero eigenvalues if and only if the system

$$C^T\mathbf{y} = -B\mathbf{x}_2$$

has a solution.

If instead $k = n_1 = n_2$, i.e., if C is nonsingular, then $\lambda = 0$ implies that both $\mathbf{x}_1 = 0$ and $\mathbf{x}_2 = 0$. Hence, $\mathbf{x} = \left[\begin{smallmatrix}\mathbf{x}_1\\ \mathbf{x}_2\end{smallmatrix}\right] = 0$, and all the eigenvalues of A are different from zero.

We finally turn to the case when the submatrix C is rank deficient, that is, $k < \min\{n_1, n_2\}$. The right-hand side of (3.4.3) is equivalent to

$$\mathbf{x}_2 \in \mathcal{N}(C), \quad C^T\mathbf{x}_1 = -B\mathbf{x}_2.$$

Let $\mathbf{x}$ be a nontrivial solution of (3.4.2). When $\mathbf{x}_2 = \mathbf{0}$, there has to be a vector $\mathbf{x}_1 \neq \mathbf{0}$ with $C^T\mathbf{x}_1 = \mathbf{0}$. Since in this case the null space of C^T has dimension $n_1 - k$, there are $n_1 - k$ linearly independent solutions of (3.4.2) with $\mathbf{x}_2 = \mathbf{0}$.

The existence of a solution $\mathbf{x}$ of (3.4.2) with a nonzero subvector $\mathbf{x}_2$ is equivalent to

$$\dim\left(\mathcal{R}(B_{|\mathcal{N}(C)}) \cap \mathcal{R}(C^T)\right) \geq 1.$$

This condition does not hold for most matrix pairs (B, C). □

Remark 3.4.2. We note that if $B = 0$, then the equation $A\mathbf{x} = \mathbf{0}$ has exactly

$$\dim\mathcal{N}(C) + \dim\mathcal{N}(C^T) = n - 2\,\mathrm{rank}(C)$$

linearly independent solutions.

Theorem 3.4.1 proves that if a network has a large anti-community ($n_1 > n_2$) then the spectral decomposition of its adjacency matrix $A = WDW^T$ has the form

$$W = \begin{bmatrix} E & U_2 & F \\ G & O_{n_2,n_1-n_2} & H \end{bmatrix}, \quad D = \begin{bmatrix} D_1 & O & O \\ O & O_{n_1-n_2} & O \\ O & O & D_2 \end{bmatrix}.$$

The structures of W and D are very similar to those of W_B and Λ_B in (3.2.9), respectively. For this reason, the bipartization algorithm described in Section 3.3, is able to detect the presence of a large anti-community and to order the nodes so that the adjacency matrix takes the form (3.4.1). In case a group of nodes is only approximately an anti-community, the algorithm produces an adjacency matrix that approximates (3.4.1).

To summarize, when $n_1 > n_2$, if a network is either bipartite or contains a large anti-community, its adjacency matrix has zero eigenvalues. The converse is not true in general, however if A has a multiple zero eigenvalue, then we can recognize the presence of one of the two above cases by observing the structure of the eigenvector matrix.

3.5 Computed examples

In the following numerical experiments, we fix the integers n_1 and n_2, and construct a random matrix A having the block form (3.2.1), with a sparse block C with density ξ. The matrix is first perturbed, by replacing its (1,1) and (2,2) zero blocks by sparse matrices of appropriate size and density η, and then "scrambled", by applying the same random permutation to its rows and columns.

We apply the algorithm of Section 3.2 to the matrix A either by supplying the cardinality of the two sets $\mathcal{V}_1$ and $\mathcal{V}_2$ (this approach is referred to as `specbip-`n_1), or letting the

method estimate n_1 and n_2 from the data; we refer to the latter approach as `specbip`. Since the block (1,2) of the matrix returned by the method is generally permuted with respect to the initial test matrix, the rows and columns are reordered according to the original sequence of the nodes. The final reordering allows us to make a comparison between the resulting matrix A_B and the test matrix A.

The results obtained with our spectral algorithm are then compared to the ones obtained by red-black ordering using the MatlabBGL library [80], a Matlab package implementing graph algorithms. A matrix has a red-black ordering if the corresponding graph is bipartite. To find a bipartite ordering, this software uses a breadth first search algorithm, starting from an arbitrary vertex. The partition of the nodes is determined by forming a group containing all the vertices having even distance from the node selected as root, and another group with the vertices at odd distance from the root. This procedure is designed for bipartite networks, not to produce an approximation when the bipartization is not exact.

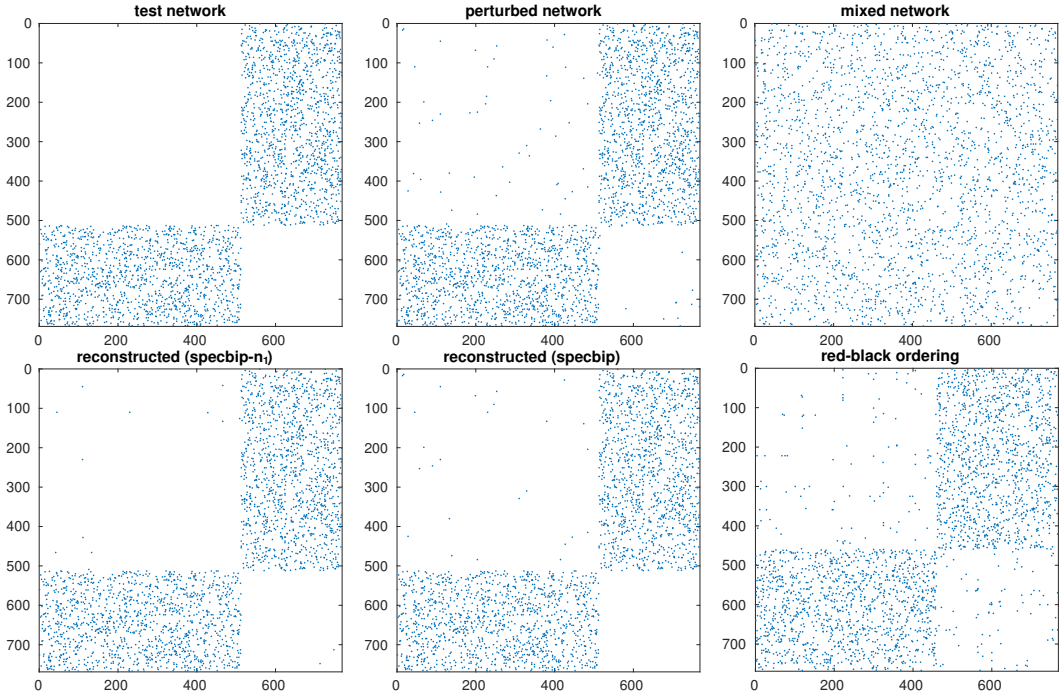


Figure 3.1: $(n_1, n_2) = (512, 256)$, $(\tilde{n}_1, \tilde{n}_2) = (492, 276)$, $\xi = 10^{-2}$, $\eta = 10^{-4}$.

Figure 3.1 displays the results for a test matrix with $(n_1, n_2) = (512, 256)$, sparsity of the block C $\xi = 10^{-2}$, and perturbation $\eta = 10^{-4}$ for the zero blocks (1,1) and (2,2) of the starting matrix. In particular, it reports in the upper row a spy plot of the original test matrix, the perturbed version, with random arcs in the (1,1) and (2,2) blocks, and the permuted matrix that is fed to the bipartization methods. The bottom row shows the reconstructed networks. The `specbip`-n_1 approach, which receives the information about the cardinality of the node sets, produces the matrix closest to the original. The general algorithm estimates the cardinalities $(\tilde{n}_1, \tilde{n}_2) = (492, 276)$, according to the number of "small" eigenvalues; see Figure 3.2, where the absolute values of the eigenvalues are displayed in nondecreasing order. This version of the spectral bipartization algorithm produces a slightly less accurate approximation than the previous one, but is anyway much better than the matrix produced

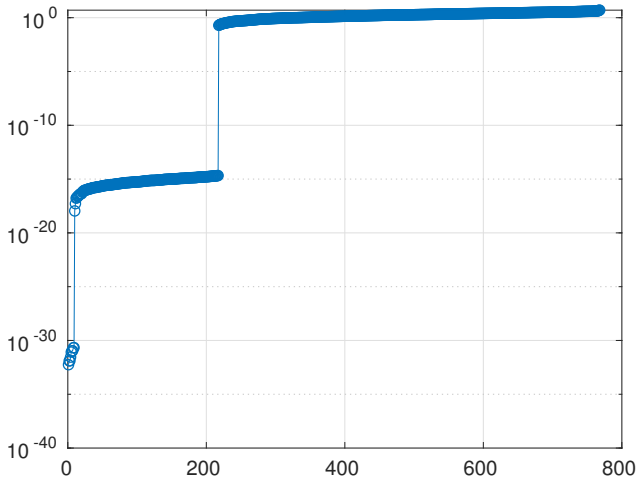


Figure 3.2: $(n_1, n_2) = (512, 256)$, $(\tilde{n}_1, \tilde{n}_2) = (492, 276)$, $\xi = 10^{-2}$, $\eta = 10^{-4}$.

by using the red/black ordering.

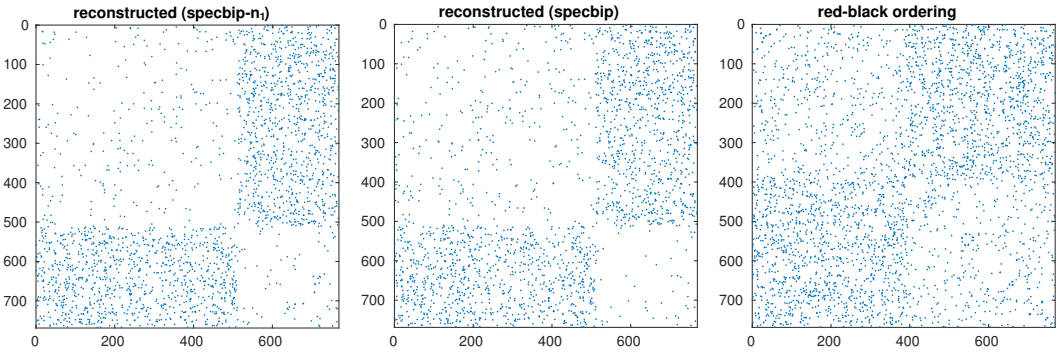


Figure 3.3: $(n_1, n_2) = (512, 256)$, $(\tilde{n}_1, \tilde{n}_2) = (396, 372)$, $\xi = 10^{-2}$, $\eta = 10^{-3}$.

In Figure 3.3 we show the results for a test matrix similar to the previous one, but with a larger perturbation $\eta = 10^{-3}$ on its zero blocks. The estimation of (n_1, n_2) is inaccurate in this case, but the approximation produced by the `specbib` methods is quite close to the unperturbed matrix, while the red/black ordering matrix is far from it.

Now, let

$$E = A - A_B = \begin{bmatrix} E_{11} & E_{12} \\ E_{21} & E_{22} \end{bmatrix},$$

where E_{11} and E_{22} are square matrices of size n_1 and n_2, respectively, and let $|M|$ denote the number of nonzero elements of M. To evaluate the quality of the results obtained by using our algorithms, we consider the following three indices

$$\mathcal{I}_B = 1 - b_s, \qquad \mathcal{E}_B = \frac{|E_{11}|}{n_1^2} + \frac{|E_{22}|}{n_2^2}, \qquad \mathcal{E}_A = \frac{|E_{12}|}{|C|}.$$

The first two indices measure the distance of the resulting matrix A_B from the adjacency matrix of a bipartite graph; see (3.1.3) for the definition of the spectral bipartivity index

b_s. The third index measures the approximation error with respect to the starting bipartite network (3.2.1). To better evaluate the error in the bipartition, we introduce the fourth index $\mathcal{E}_N = \widetilde{\mathcal{E}}_N/n_1$, where $\widetilde{\mathcal{E}}_N$ is the number of nodes from the set $\mathcal{V}_1$ that were incorrectly ascribed to the set $\mathcal{V}_2$.

Tables 3.1, 3.2, and 3.3 report the average values of the above four quality indices over 10 realizations of the random test networks. Three different pairs (n_1, n_2) are considered and each table refers to different densities ξ and η; T stands for the execution time in seconds.

Table 3.1: Results for $\xi = 10^{-2}$, $\eta = 10^{-4}$.

(256,128)	`specbip`-n_1	`specbip`	red-black
$\mathcal{I}_B$	1.22e-16	1.89e-16	2.33e-03
$\mathcal{E}_B$	5.46e-04	6.74e-04	3.72e-03
$\mathcal{E}_A$	2.80e-01	2.79e-01	-
$\mathcal{E}_N$	1.45e-01	1.58e-01	2.76e-01
T	4.94e-02	5.05e-02	3.15e-04
(512,256)	`specbip`-n_1	`specbip`	red-black
$\mathcal{I}_B$	1.11e-17	1.11e-17	2.98e-03
$\mathcal{E}_B$	1.13e-04	1.50e-04	3.39e-03
$\mathcal{E}_A$	4.84e-02	6.27e-02	-
$\mathcal{E}_N$	3.36e-02	5.96e-02	2.97e-01
T	2.77e-01	2.95e-01	4.94e-04
(1024,512)	`specbip`-n_1	`specbip`	red-black
$\mathcal{I}_B$	7.77e-17	0.00e+00	4.17e-02
$\mathcal{E}_B$	9.92e-05	2.11e-04	4.75e-03
$\mathcal{E}_A$	1.06e-01	1.80e-01	-
$\mathcal{E}_N$	3.62e-02	1.15e-01	2.75e-01
T	1.92e+00	1.94e+00	8.67e-04

A comparison of the tables shows that the spectral bipartization algorithm is more accurate than the red-black ordering method. At the same time, it is much slower than the MatlabBGL function, as in our experiments we compute the whole spectrum of the given adjacency matrix, without exploiting its sparsity. To be competitive with existing methods for large-scale problems, the spectral method should be modified in order to perform its task by suitable iterative methods, in order to take advantage of the structure of the adjacency matrix.

From the tables, it can also be observed that knowing in advance the cardinality of the two sets $\mathcal{V}_1$ and $\mathcal{V}_2$ leads in some cases to a substantial improvement in the quality of the results.

To further investigate the behavior of the bipartition error, we construct a matrix A of the form (3.2.1), letting $n_1 = 512$ and $n_2 = 256$, with a sparse random block C having density $\xi = 10^{-2}$. After randomly permuting the rows and columns, we apply our algorithms to this matrix, as well as to those perturbed by replacing the (1,1) and (2,2) blocks by a sparse matrix with density $\eta = 10^{-6}, 10^{-5}, \ldots, 10^{-3}$. The graph on the left of Figure 3.4 shows the value of the bipartization error $\mathcal{E}_N$ obtained when our two algorithms and the red-black ordering method are applied to an unweighted graph; the graph on the right corresponds to a weighted graph. All values are averaged over 10 realizations of the random test matrices. Both graphs show that the bipartization determined by our approaches is closer to the

Table 3.2: Results for $\xi = 10^{-2}$, $\eta = 10^{-5}$.

(256,128)	specbip-n_1	specbip	red-black
$\mathcal{I}_B$	1.11e-17	7.77e-17	1.68e-06
$\mathcal{E}_B$	6.68e-04	8.79e-04	3.36e-03
$\mathcal{E}_A$	2.70e-01	2.68e-01	-
$\mathcal{E}_N$	1.23e-01	1.49e-01	2.58e-01
T	4.39e-02	4.70e-02	1.01e-03
(512,256)	specbip-n_1	specbip	red-black
$\mathcal{I}_B$	0.00e+00	2.22e-17	1.40e-04
$\mathcal{E}_B$	3.05e-05	1.91e-05	8.81e-04
$\mathcal{E}_A$	3.88e-02	2.38e-02	-
$\mathcal{E}_N$	1.87e-02	1.93e-02	3.16e-01
T	2.72e-01	2.77e-01	5.12e-04
(1024,512)	specbip-n_1	specbip	red-black
$\mathcal{I}_B$	0.00e+00	0.00e+00	4.04e-03
$\mathcal{E}_B$	1.91e-07	1.03e-05	1.07e-03
$\mathcal{E}_A$	1.73e-04	9.49e-03	-
$\mathcal{E}_N$	9.77e-05	9.47e-03	3.25e-01
T	1.91e+00	1.89e+00	9.52e-04

Table 3.3: Results for $\xi = 10^{-1}$, $\eta = 10^{-4}$

(256,128)	specbip-n_1	specbip	red-black
$\mathcal{I}_B$	0.00e+00	0.00e+00	7.71e-02
$\mathcal{E}_B$	0.00e+00	5.83e-04	1.35e-02
$\mathcal{E}_A$	2.43e-02	4.24e-02	-
$\mathcal{E}_N$	0.00e+00	2.58e-02	3.18e-01
T	5.56e-02	6.05e-02	3.07e-03
(512,256)	specbip-n_1	specbip	red-black
$\mathcal{I}_B$	0.00e+00	0.00e+00	1.44e-01
$\mathcal{E}_B$	0.00e+00	8.19e-04	8.01e-03
$\mathcal{E}_A$	8.02e-03	5.19e-02	-
$\mathcal{E}_N$	0.00e+00	4.47e-02	3.31e-01
T	2.77e-01	2.76e-01	1.08e-03
(1024,512)	specbip-n_1	specbip	red-black
$\mathcal{I}_B$	0.00e+00	0.00e+00	2.60e-01
$\mathcal{E}_B$	0.00e+00	1.04e-03	6.54e-03
$\mathcal{E}_A$	2.33e-03	9.04e-02	-
$\mathcal{E}_N$	0.00e+00	8.71e-02	3.28e-01
T	2.02e+00	2.07e+00	3.99e-03

correct one, with respect to red-black ordering. We can also observe that specbip-n_1, that receives the information about the cardinalities rather than estimating them by detecting the largest gap between "small" and "large" eigenvalues, produces slightly better results. The performance of all algorithms degrades as the perturbation becomes less sparse.

In Figure 3.5, we display the value of $\mathcal{E}_N$ for the same examples, for a fixed $\eta = 10^{-2}$,

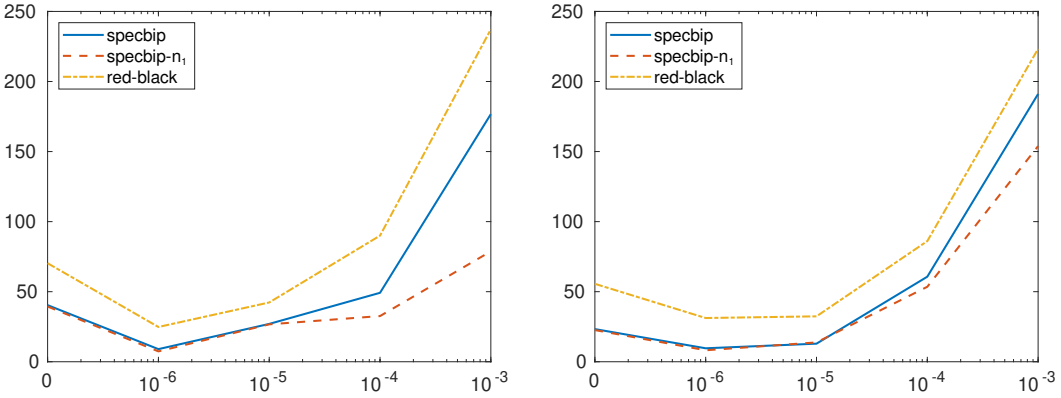


Figure 3.4: Bipartition error $\widetilde{\mathcal{E}}_N$ for $(n_1, n_2) = (512, 256)$; on the left unweighted random graphs, on the right weighted random graphs, both with $\xi = 10^{-2}$, as a function of $\eta = 0, 10^{-6}, 10^{-5}, \ldots, 10^{-3}$.

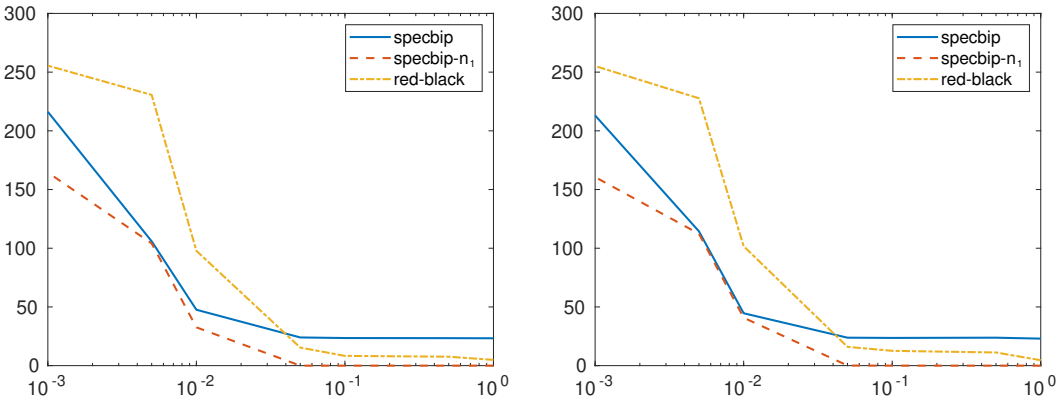


Figure 3.5: Bipartition error $\widetilde{\mathcal{E}}_N$ for $(n_1, n_2) = (512, 256)$; on the left unweighted random graphs, on the right weighted random graphs, both with $\eta = 10^{-2}$, as a function of $\xi = 10^{-3}, 10^{-2}, 10^{-1}, 1$.

and letting the density ξ of the block C take values in $[10^{-3}, 1]$. The red-black ordering method is more accurate than the `specbip` algorithm for very sparse networks, while providing the correct cardinality of the set $\mathcal{V}_1$ to `specbip`-n_1 produces the best results.

3.5.1 The NDyeast network

In this paragraph, we illustrate the performance of the spectral bipartization algorithm when applied to the detection of anti-communities by analyzing a case study. The *NDyeast* network is an undirected graph which describes the protein interaction network for yeast, each edge representing an interaction between two proteins [102]. The data set is available at [148]. In the following we analyze this network, testing the presence of a bipartization or of a large anti-community.

The *NDyeast* network has 2114 nodes and 2277 edges. In particular, there are 74 self-loops (nodes connected only to themselves) and 268 "isolated" nodes, i.e. vertices

disconnected from the network. We consider then the adjacency matrix resulting by removing both the self-loops and the isolated nodes. This reduced adjacency matrix has size $n = 1846$, and 149 connected components. They were identified by the getconcomp function from the PQser Matlab toolbox [41], described in Algorithm 4.

In the case of a reducible adjacency matrix, the spectral bipartization algorithm should treat each single connected component one at a time. Since most of the components in the *NDyeast* network are very small, the majority of them having just 2 or 3 nodes, we consider the only component with more than 10 nodes, which has 1458 nodes. We process the reduced adjacency matrix A with our bipartization method.

The algorithm determines $n_0 = 564$ zero eigenvalues (see Figure 3.6) and identifies two sets of nodes with cardinalities $n_1 = 1011$ and $n_2 = 447$.

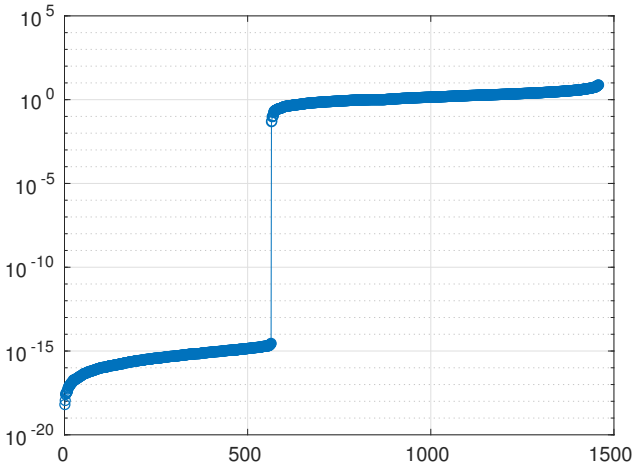


Figure 3.6: Spectrum of the reduced adjacency matrix for the *NDyeast* network.

The starting adjacency matrix is displayed in the top-left spy plot of Figure 3.7. The top-right plot shows the same matrix after the ordering produced by the spectral bipartization algorithm is applied to its rows and columns. This graph clearly displays that there is a large group of nodes in the *NDyeast* network that do not communicate much among themselves, that is, an anti-community. In the same graph we show the bipartization detected by the algorithm by means of red lines.

Our algorithm can also be applied by supplying the values of (n_1, n_2), rather than estimating them from the number of zero eigenvalues. If we do this by setting $\tilde{n}_1 = 800$ and $\tilde{n}_2 = 658$, we obtain the bottom left graph in the same figure. It shows that in the group of the first 800 proteins, only four of them directly interact.

The bottom-right graph of Figure 3.7 displays the result of the red-black ordering method, which does not supply any useful information.

We remark that a data set similar to *NDyeast* (but different) is available at [148]. It is called simply *yeast* and represents the protein-protein interaction network in budding yeast. The *yeast* network consists of 2361 nodes, 7182 edges such that 536 of them are self-loops) and it refers to the paper [30]. By processing this data set with our spectral algorithm, we obtain results very similar to the ones displayed in Figure 3.7.

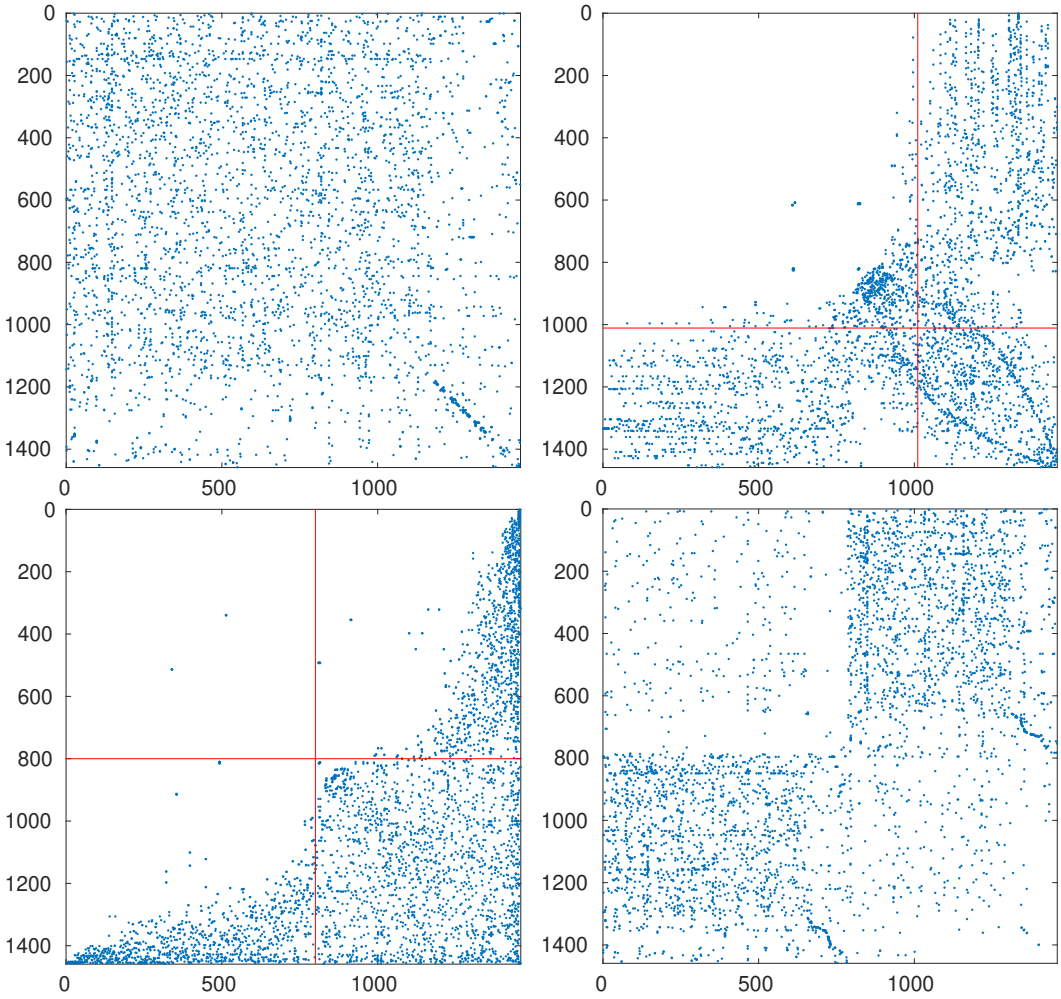


Figure 3.7: Analysis of the *NDyeast* network: top-left, the starting adjacency matrix; top-right, the node reordering produced by the spectral algorithm; bottom-left, the reordering induced by the choice $(\tilde{n}_1, \tilde{n}_2) = (800, 658)$; bottom-right, the output of the red-black ordering method.

3.5.2 The geom network

We also applied the spectral bipartization algorithm to a weighted graph, namely, the *geom* network This dataset It is extracted from the Computational Geometry Database *geombib* by B. Jones (version 2002). Nodes represent authors and the value of the entry (i, j) of the adjacency matrix is the number of papers coauthored by authors i and j. The data set is available at [148].

The *geom* network has 7343 nodes and 11898 edges. After removing 1185 isolated nodes, the network presents 875 connected components identified again with the `getconcomp` function, the largest of which has 3621 nodes. We applied the bipartization method to the reduced adjacency matrix associated with this largest connected component.

The eigenvalues are displayed in Figure 3.8; 533 of them are detected as being numerically zero, and the cardinalities of the two node sets are then $n_1 = 2077$ and $n_2 = 1544$.

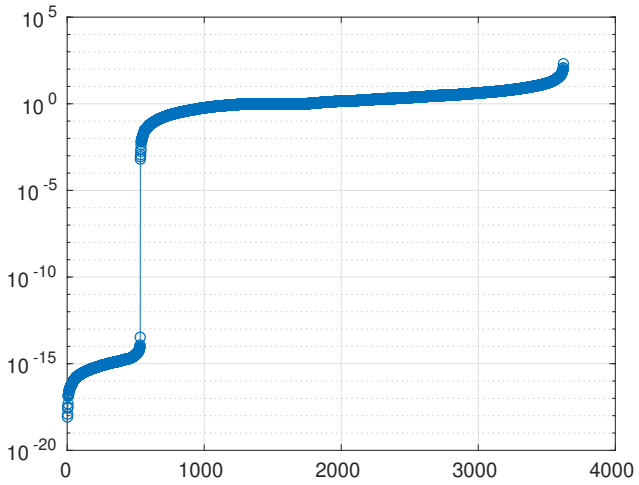


Figure 3.8: Spectrum of the reduced adjacency matrix for the *geom* network.

The left graph of Figure 3.9 reports the spy plot of the original adjacency matrix; the graph on the right shows the matrix reordered by the spectral bipartization algorithm. The graph highlights the presence of an anti-community of about 1000 authors, who did not collaborate with each other when writing papers.

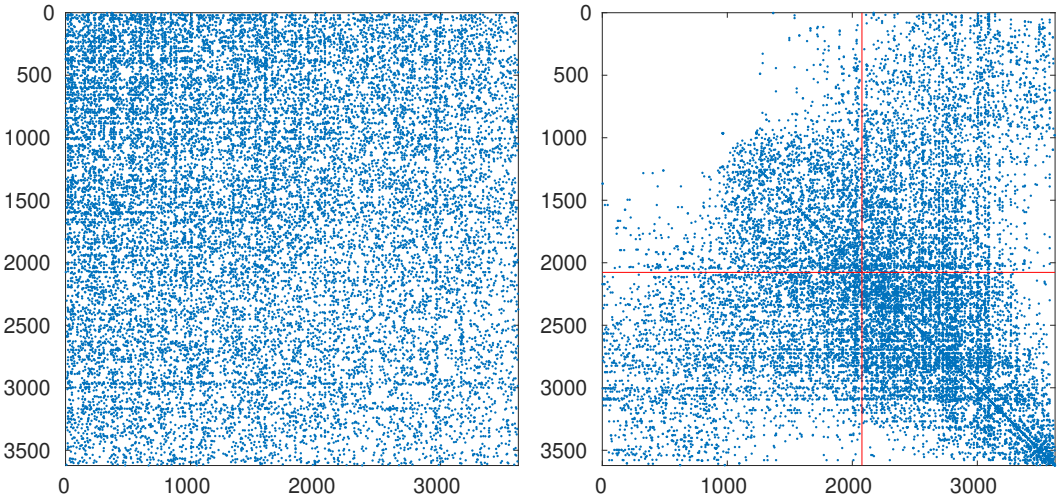


Figure 3.9: Analysis of the *geom* network: on the left, the starting adjacency matrix; on the right, the node reordering produced by the spectral algorithm.

3.6 Conclusions and future work

This chapter describes how an approximate bipartization of a given graph can be determined by solving a sequence of optimization problems. Some computed examples have been given to illustrate the performance of the presented spectral method. This technique can also be applied for detecting the presence of a large anti-community in a network and identifying

it. Two case studies have been numerically analyzed for finding a large anti-community in both unweighted and weighted networks. Future work is related to generalizing the spectral method for finding approximate multi-partite structures. Indeed, bipartivity can also be generalized to k-partivity. A graph is said to be *k-partite* if its vertices can be partitioned into k non-empty, disjoint subsets $V_1, V_2, \ldots, V_k$ such that there is no edge between nodes belonging to the same set.

Another possible future development involves the use of the spectral method described in this chapter for solving the seriation problem in the presence of imperfect data by studying the asymptotic behaviour of a stochastic block matrix as mentioned at the end of Chapter 2.

Chapter 4

Photometric stereo under unknown lighting

The aim of this Chapter is to give an overview of a particular photographic technique, called *Photometric Stereo*, for reconstructing the 3D shape of an object and of its application in archaeology.

Recovering the shape of an object by using one or more 2D digital images as input, is a classical fundamental problem in Computer Vision and in Applied Mathematics in general. 3D recovery is frequently adopted in different application fields such as, for example, topography for mapping the surfaces of the Earth and other planets [23], medicine for medical images [89] and for the reconstruction of computed tomography [186], and archaeology as will be described in the following.

4.1 Shape-from-Shading, Photometric Stereo and application to archaeology

In Computer Vision all the methods for recovering the shape of an object belong to the class of the so-called *Shape-from-X* techniques. The problems in this set of approaches differ from one to another for the kind of input data, even though their common aim is that of obtaining a 3D reconstruction of surfaces. Shape-from-X techniques include Shape-from-Contour [177], Shape-from-Fractal Geometry [112], Shape-from-Motion [175], Shape–from–Polarization [7], Shape-from-Shading [96], Shape-from-Stereo [93] and Shape-from-Texture [79].

Shape-from-Shading (SfS) exploits shading information when one source illuminates an object and hence, it deals with the recovery of the 3D shape of the observed object from a gradual variation of shading in the given image; see [97, 111, 167]. Since only one picture is available, the classical Shape-from-Shading problem is not well posed. The impossibility of avoiding any ambiguity in recovering the 3D shape of an object from a single image, arises from the difficulty met in distinguishing concave from convex surfaces. A way to solve this problem and remove the uncertainty due to the presence of a single digital picture, is to use more than one image. Woodham showed in [185] that two is the minimum number of pictures to ensure well-posedness of the SfS problem introducing a novel technique, the one analysed in this Chapter, called *Photometric Stereo* (PS).

Photometric Stereo is a SfS approach that uses multiple images of an object, taken

under different lighting directions. Specifically, it exploits shading information when several sources illuminate the observed object and it requires that the object is observed from a fixed point of view but under different lighting conditions. Photometric Stereo is opposite to another possible approach to the SfS problem, namely the *stereo vision* photographic technique. Indeed, stereo vision, sometimes generalized to multiple views or *multi-view* [60], assumes the availability of different pictures of an object obtained varying the point of view but taken under the same illumination. The pictures are typically acquired by a set of fixed cameras, or extracted from the frames of a movie shot by a movable camera. Photometric stereo (PS), on the other hand, uses a fixed camera and a movable light to acquire a set of images that embed shape and color (albedo) information of the framed object [37].

Practical PS has some severe limitations. The ideal PS requires the position of the direction of the light sources and the illumination intensity used for acquiring the pictures, to be accurately known [13]. Any deviation from this requirement often results in a reconstruction affected by distortion. Various attempts have been made to estimate the lights position directly from the data; see, e.g., [14, 34] where a new approach, based on the decomposition of the effective lights into linear combination of special functions (spherical harmonics), is proposed. Releasing the constraint on the knowledge of the precise positioning of the light sources, makes the acquisition process much simpler. Since the first Shape-from-Shading technique was developed by Horn in the early 1970s [98], many different approaches and numerical algorithms have emerged. See [190] for a survey on different SfS techniques and a comparison between SfS algorithms. For an updated review on numerical methods for solving the SfS problem see [59]; see [1, 181] for surveys on PS techniques.

3D restoration has a wide application in archaeology; digital archiving of 3D objects, representing archaeological discoveries, is fundamental in cultural heritage conservation. The main application in our research is *rock art documentation*, in particular the decorations in the "Domus de Janas" (Sardinian for "House of the Fairies" or "House of Witches"), a particular kind of pre-Nuragic chamber tombs, typical of Sardinia, dated from the late Neolithic and early Bronze Age. In an archaeological setting, a 3D restoration technique easy to practically implement, like photometric stereo, is crucial because the findings are frequently located in narrow positions and the current protocols exclude any physical contact for creating replicas and, hence, extensive acquisition systems are not suitable. The difficulty to access specific sites, often associated to a large number of items to be documented, makes it impractical to use other 3D reconstruction techniques like the most popular one represented by *laser range scanning*. Laser scanning has several drawbacks since it is characterized by long acquisition time and large instrumentation cost. Moreover, the surface should be sufficiently small to to fit into the detector and the need for the laser scanner to be in close proximity to the finding, makes this technique not practical for recovering the shape of sculptures or artifacts that have to be preserved. Because of the fact that PS overcomes the disadvantages of laser scanning, this technique represents an efficient way to generate accurate 3D scans of objects. Indeed, PS is not only a less invasive approach but also an economical one. Therefore, the cheap instrumentation (only a camera, a tripod and a hand-positioned lighting source) would allow for 3D recording in the most difficult sites and also for parallel operation on different findings by a team of researchers. The research group in Cagliari, have been devoted various papers to the application of PS to the archaeological rock art documentation; see [52, 127, 180].

After introducing the notation adopted in this Chapter, we first review a model for the

solution of the photometric stereo problem assuming that the light source position is a known quantity. This is a realistic assumption in controlled conditions such as, for example, in scientific laboratories. In uncontrolled settings this information is not always accessible. Hence, in the remaining of the Chapter, we explore the photometric stereo problem under unknown lighting conditions. The considered method was developed by Hayakawa [88]. It represents the first work for dealing with the PS problem without an a priori knowledge of the light source direction and it gave rise to other commonly named *uncalibrated* photometric stereo approaches. Hayakawa showed that the lights position can be estimated directly from the data, when at least 6 images of the observed object are available, under suitable conditions. We will recall a theorem that gives sufficient conditions for the existence of the solution and illustrate some numerical results for proving the effectiveness of the method. Finally, we will describe possible future developments.

4.2 Notation and classical assumptions

When light interacts with matter on a macroscopic level, it can be absorbed, transmitted, or reflected. Here we will take into account only the reflection phenomenon by considering the fraction of light that is reflected.

A 3D reconstruction method based on the photometric stereo approach can recover the shape of an object by first estimating the surface normal vectors and the surface reflectance of the observed object from three or more pictures taken under different lighting conditions. The images are acquired by using a fixed camera, a movable light source and, in order to obtain an effective reconstruction, also the object has to be considered not in motion. By using the extracted photometric data, the 3D surface is obtained by integrating the normal vectors.

Let us consider an object, whose 3D shape has to be recovered, placed at the origin of a reference system in $\mathbb{R}^3$. The object is observed from a fixed camera placed along the $z-$axis so that the latter coincides with the *optical axis*, and it is directed from the object to the camera. The light sources need to be placed sufficiently far from the object so that the surface is uniformly illuminated and the lighting rays are parallel. Hence, the point of view is assumed to be at infinite distance from the observed object, and different pictures of the object itself are taken, each one corresponding to a different light direction. We assume that each image has resolution $(r+2)\times(s+2)$, and we let A to be the length of the horizontal side of a picture. Assuming the pixels to be square, we let the length of the vertical side be $B=(s+1)h$, with $h=A/(r+1)$. Hence, each picture defines a domain $\Omega=[-A/2,A/2]\times[-B/2,B/2]$, and induces the discretization

$$x_i=-A/2+ih,\quad i=0,\dots,r+1,\quad y_j=-B/2+jh,\quad j=0,\dots,s+1. \tag{4.2.1}$$

We assume that the surface of the object is represented by a function of the type $z=u(x,y)$, with $(x,y)\in\Omega$. Consequently,

$$\nabla u(x,y)=\begin{bmatrix}\dfrac{\partial u(x,y)}{\partial x}\\[2ex] \dfrac{\partial u(x,y)}{\partial y}\end{bmatrix}=\begin{bmatrix}u_x\\ u_y\end{bmatrix},\qquad \mathbf{n}(x,y)=\frac{(-u_x,-u_y,1)^T}{\sqrt{1+\|\nabla u\|^2}}, \tag{4.2.2}$$

denote the gradient of u and the normal vector to the surface of the object, respectively.

As it is customary, when it is useful to vectorize images we order the pixels lexicographically, i.e., stacking into a single vector the columns of the matrix containing the image. The pixel of coordinates (i, j) takes the index $k = (i-1)s + j$, where $k = 1, \dots, p$, and p is the number of pixels in the image. We will set either $p = (r+2)(s+2)$ or $p = rs$ depending on whether the boundary pixels are considered in the discretization or not. For each point in the discretization of the domain Ω, we write indifferently

$$\begin{aligned} u(x_i, y_j) &= u_{i,j} = u_k, \\ u_x(x_i, y_j) &= (u_x)_{i,j} = (u_x)_k, \\ u_y(x_i, y_j) &= (u_y)_{i,j} = (u_y)_k, \\ \mathbf{n}(x_i, y_j) &= \mathbf{n}_{i,j} = \mathbf{n}_k, \end{aligned}$$

using the two-index notation to refer to the values on the grid, and the one-index notation to identify the values after the vectorization.

We assume that q pictures are available, each one taken with light source placed at infinite distance from the origin along the direction

$$\boldsymbol{\ell}_t = \begin{pmatrix} \ell_{1t} \\ \ell_{2t} \\ \ell_{3t} \end{pmatrix}, \qquad t = 1, \dots, q.$$

Each vector $\boldsymbol{\ell}_t$ stems from the object to the light source and its Euclidean norm is proportional to the light intensity. This introduces an undetermined proportionality constant in the problem. The image vectors are denoted by $\mathbf{m}_1, \mathbf{m}_2, \dots, \mathbf{m}_q \in \mathbb{R}^p$.

The aim consists in reconstructing the 3D shape of the observed object. In particular, we consider the *Lambertian model* according to which the light incident on a surface is reflected equally in all directions [111]. A Lambertian surface appears equally bright from any direction, for any illumination direction. Therefore, its reflected intensity is independent of the viewing direction. Specifically, if the surface of the object is *Lambertian*, then the light intensity observed at each point is proportional to the angle between the normal to the observed surface at that point and the light direction. In particular, the surface obeys the Lambert's cosine law which can be stated as

$$\rho(x, y)\, \langle \mathbf{n}(x, y), \boldsymbol{\ell}_t \rangle = I_t(x, y), \qquad t = 1, \dots, q \tag{4.2.3}$$

where the scalar ρ_k represents the *albedo* of the surface point at coordinates (x, y), which keeps into account the partial light absorption of that portion of the surface, $I_t(x, y)$ is the light intensity at the same point in the tth image, and $\langle \cdot, \cdot \rangle$ is the usual inner product in $\mathbb{R}^3$. When the albedo is constant, the object is said to be a *Lambertian reflector*.

Note that real objects having a perfect Lambertian surface are rarely met in practice. Indeed, Lambertian reflection surfaces are typically characterised also by *specular reflection*. Specifically, there are reflection directions that do not obey the Lambert's law. In these situations it is used the so-called *Phong's model* which is more complete and realistic compared with the Lambertian one. This model is based on the linear combination of three components of the reflected light, namely, the diffuse (i.e. the Lambertian component), the specular and the ambient part.

The assumption on the surface, constrained to be Lambertian, simplifies the computation. Relaxations of the photometric stereo problem to non-Lambertian reflectance models can be

found in [40, 143]. See [52] for how to deal with technical photographic issues, related to cameras, lights and computer screens, concerning the application of PS to real cases.

Summing up, the classical assumptions for the considered model are the following:

- the surface is Lambertian;
- the light sources are placed at infinite distance from the object;
- no portion of the surface is shaded in all the pictures otherwise the shape reconstruction may have spherical distortion;
- the camera need to be sufficiently far from the object so that perspective deformations can be neglected.

When the light sources directions and intensities are known, the photometric stereo problem can be solved as a linear system as it will be explained in Section 4.3. One of the simplifying assumptions that are typically used (together with the Lambertian model, distant point light sources and ignoring shadows/inter-reflections) is the *orthographic* projection of the scene onto the image sensor. Although the majority of the photometric stereo approaches assumes an orthographic projection, the *perspective* projection has been shown to be a more realistic assumption. Papadhimitri and Favaro [150] showed how to incorporate a perspective camera model in a photometric reconstruction. In particular, they introduced a novel perspective in uncalibrated photometric stereo showing that one can uniquely reconstruct the normals of the object and the lights given only the input images and the camera calibration (focal length and image center).

The methods discussed in this Chapter, assume an *orthographic* camera model and deal with greyscale image. For a generalisation of existing greyscale photometric stereo (GPS) techniques to the use with colour images see [13].

4.3 Photometric stereo with known lights position

In this Section we assume that the light sources position is known, i.e. the light directions $\boldsymbol{\ell}_t$, $t = 1, \dots, q$, are given. In the following, we briefly recall the nonlinear differential Hamilton–Jacobi model which is based on a continuous formulation of the Photometric stereo problem and, subsequently we will consider, more in detail, the Poisson approach.

4.3.1 Hamilton–Jacobi formulation

The continuous formulation (4.2.3) of Lambert's law leads to a Hamilton–Jacobi differential model; see [132, 133] for a thorough study. Here we briefly recall its construction.

Let us set the normal field to the surface and the light directions

$$\mathbf{n}(x,y) = \frac{(-u_x, -u_y, 1)^T}{\sqrt{1 + \|\nabla u\|^2}}, \qquad \boldsymbol{\ell}_t = \begin{pmatrix} \widetilde{\boldsymbol{\ell}}_t \\ \ell_{3t} \end{pmatrix}, \qquad \widetilde{\boldsymbol{\ell}}_t \in \mathbb{R}^2, \quad t = 1, \dots, q.$$

Then, the Lambert's cosine law (4.2.3) becomes

$$\rho(x,y)\, \frac{\langle -\nabla u(x,y), \widetilde{\boldsymbol{\ell}}_t \rangle + \ell_{3t}}{\sqrt{1 + \|\nabla u(x,y)\|^2}} = I_t(x,y), \qquad t = 1, \dots, q,$$

or, equivalently,

$$\sqrt{1+\|\nabla u(x,y)\|^2}I_t(x,y)+\rho(x,y)\left(\langle\nabla u(x,y),\widetilde{\boldsymbol{\ell}}_t\rangle-\ell_{3t}\right)=0.$$

By imposing Dirichlet boundary conditions, we obtain a system of q first order nonlinear PDEs of Hamilton–Jacobi type

$$\begin{cases} H_t(x,y,\nabla u(x,y))=0, & t=1,\dots,q,\\ u(x,y)=g(x,y), & (x,y)\in\partial\Omega. \end{cases}$$

Following [133], we obtain for $t=1$

$$\sqrt{1+|\nabla u(x,y)|^2}=\rho(x,y)\frac{\langle-\nabla u(x,y),\widetilde{\boldsymbol{\ell}}_1\rangle+\ell_{31}}{I_1(x,y)}$$

and then we substitute this expression in the other equations corresponding to $t=2,\dots,q$, to obtain

$$\left(\langle-\nabla u(x,y),\widetilde{\boldsymbol{\ell}}_1\rangle-\ell_{31}\right)I_t(x,y)=\left(\langle-\nabla u(x,y),\widetilde{\boldsymbol{\ell}}_t\rangle-\ell_{3t}\right)I_1(x,y).$$

This shows that the minimal number of images for the problem to be well-posed is 2. Nevertheless, if $q=2$ the solution may not exist for particular light orientations. Considering $q>2$ leads to a least-squares approach, that may be effective to reduce the influence of noise in experimental data sets, without making data acquisition significantly harder. In any case, knowing accurately the lights position $\boldsymbol{\ell}_t$ is a strong requirement.

After that the solution $u(x,y)$ is computed, the albedo is given by

$$\rho(x,y)=\frac{I_t(x,y)}{\langle\mathbf{n}(x,y),\boldsymbol{\ell}_t\rangle},\qquad\text{for any } t=1,\dots,q.$$

Conditions for the existence of solutions are discussed in [115], and in [133] the problem is studied under more realistic assumptions; see also [170].

The main disadvantage of the Hamilton–Jacobi model is that the operator to be inverted depends upon the data. The matrix of the linear system obtained through the discretization may be singular or severely ill-conditioned in certain lighting conditions. However, this problem can be tackled by suitably choosing the position of the light sources.

4.3.2 Poisson formulation

Here we consider the Poisson approach for the 3D reconstruction of a Lambertian surface. This method follows a different strategy, composed by two main parts. The first step consists of immediately discretizing Lambert's law on a regular grid, in order to determine the normal field to the surface by solving a matrix equation. After the photometric data are extracted, then the components of the normal vectors are numerically differentiated to obtain an approximate discretization of the Laplace operator. Finally, the 3D profile of the observed object is recovered by solving a Poisson partial differential equation.

An advantage of this approach is that the computation can be decoupled into simpler problems, allowing for the solution of the unknown lighting case, as it will be shown in the next section. A drawback is that this procedure requires a larger number of images

than the Hamilton–Jacobi formulation, i.e., at least 3. This is not a substantial problem in applications, since usually dozens of images can be easily made available.

Let us apply discretization (4.2.1) to equation (4.2.3). We ignore for the moment the boundary pixels, where we will impose suitable boundary conditions, and rearrange the internal pixels by the lexicographical ordering. Denoting by ρ_k and $\mathbf{n}_k$, respectively, the value of the albedo at the kth pixel and the (normalized) vector normal to the surface at the same point, then the following relation holds at any point of each picture

$$\rho_k \mathbf{n}_k^T \boldsymbol{\ell}_t = m_{kt}, \qquad k = 1, \dots, p, \quad t = 1, \dots, q. \tag{4.3.1}$$

The scalars m_{kt} represent the radiation $I_t(x, y)$ reflected by the small area near the kth pixel when illuminated from the direction $\boldsymbol{\ell}_t$, that is, the components of the vectors $\mathbf{m}_1, \mathbf{m}_2, \dots, \mathbf{m}_q \in \mathbb{R}^p$, containing the vectorized images.

By defining the matrices

$$D = \operatorname{diag}(\rho_1, \dots, \rho_p) \in \mathbb{R}^{p \times p}, \qquad N = [\mathbf{n}_1, \dots, \mathbf{n}_p] \in \mathbb{R}^{3 \times p},$$

representing the albedo and the normal vectors field, respectively, and the matrices

$$L = [\boldsymbol{\ell}_1, \dots, \boldsymbol{\ell}_q] \in \mathbb{R}^{3 \times q}, \qquad M = [\mathbf{m}_1, \dots, \mathbf{m}_q] \in \mathbb{R}^{p \times q},$$

containing the light sources direction and the pictures, the equations (4.3.1) can be grouped into the equation

$$DN^T L = M \tag{4.3.2}$$

which represents the matrix formulation of the Lambert's law.

When the lights positions are known, i.e. the matrix L is given, we first compute

$$\widetilde{N}^T = ML^\dagger, \tag{4.3.3}$$

where $L^\dagger$ is the Moore-Penrose pseudoinverse of L [18]. Then, the matrices D and N, defining the albedo and the normal vectors, can be computed from the factorization $ND = \widetilde{N}$ by simply normalizing the columns of $\widetilde{N}$.

For the solution of (4.3.3) to be unique, it is necessary that $q \geq 3$, from which we see that the minimum number of images required to obtain the normal field by this approach is 3.

Once the field of the normal vectors to the surface is obtained, we consider the vectors

$$((u_x)_k, (u_y)_k, -1)^T = -\frac{\mathbf{n}_k}{(\mathbf{n}_k)_3},$$

obtained by normalizing to -1 the third component of the normals $\mathbf{n}_k$; see (4.2.2). We numerically differentiate the first two components of the above vectors to obtain an approximation on the grid (4.2.1) of the Laplacian $f(x, y) = u_{xx} + u_{yy}$. To do that, we employ the following formula, based on the second order centered finite differences approximation for the first derivative

$$f_{i,j} = f(x_i, y_j) \approx \frac{(u_x)_{i+1,j} - (u_x)_{i-1,j}}{2h} + \frac{(u_y)_{i,j+1} - (u_y)_{i,j-1}}{2h}. \tag{4.3.4}$$

Now, the 3D profile of the object, represented by the explicit function $z = u(x, y)$, can be recovered by solving the Poisson partial differential equation

$$\Delta u(x, y) = f(x, y), \tag{4.3.5}$$

where Δ denotes the Laplace operator.

We start by imposing homogeneous Dirichlet boundary conditions on the solution and discretizing the Poisson equation by a second order finite differences scheme. Consider the equation (4.3.5) on the rectangle $[-A/2, A/2] \times [-B/2, B/2]$, with boundary conditions

$$\begin{aligned} u(x, -B/2) &= \phi_1(x), \qquad u(x, B/2) = \phi_2(x), \qquad x \in [-A/2, A/2], \\ u(-A/2, y) &= \psi_1(y), \qquad u(A/2, y) = \psi_2(y), \qquad y \in [-B/2, B/2], \end{aligned}$$

which corresponds to assuming that the surface of the observed object is approximately flat in a neighborhood of the boundary. Let $u(x_i, y_j) = u_{i,j}$ and $f(x_i, y_j) = f_{i,j}$ at each point of the mesh (4.2.1). Then, the boundary values are denoted by $u_{i,0} = \phi_1(x_i)$, $u_{i,s+1} = \phi_2(x_i)$, $u_{0,j} = \psi_1(y_j)$, and $u_{r+1,j} = \psi_2(y_j)$.

Discretizing the Poisson equation by the well-known 5 points scheme with stepsize h, we obtain the linear system

$$u_{i-1,j} + u_{i,j-1} - 4u_{i,j} + u_{i,j+1} + u_{i+1,j} = \tilde{f}_{i,j}, \tag{4.3.6}$$

for $i = 1, \ldots, r$ and $j = 1, \ldots, s$, with $\tilde{f}_{i,j} = h^2 f_{i,j}$. We remark that approximating $f_{i,j}$ by (4.3.4) does not deteriorate the quality of the results, as both (4.3.4) and the 5 points scheme produce an approximation of order $O(h^2)$. This assertion has been verified numerically.

By aggregating the mesh points $u_{i,j}$ by columns, we obtain the following pentadiagonal system of size $p = rs$

$$\begin{cases} T\mathbf{u}_1 + I_s\mathbf{u}_2 & = \mathbf{b}_1, \\ I_s\mathbf{u}_{i-1} + T\mathbf{u}_i + I_s\mathbf{u}_{i+1} & = \mathbf{b}_i, \qquad i = 2, \ldots, r-1, \\ I_s\mathbf{u}_{r-1} + T\mathbf{u}_r & = \mathbf{b}_r, \end{cases} \tag{4.3.7}$$

where I_s denotes the identity matrix of size s, with

$$T = \begin{bmatrix} -4 & 1 & & \\ 1 & -4 & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & -4 \end{bmatrix} \in \mathbb{R}^{s \times s}, \qquad \mathbf{u}_i = \begin{bmatrix} u_{i,1} \\ u_{i,2} \\ \vdots \\ u_{i,s} \end{bmatrix} \in \mathbb{R}^s, \qquad i = 1, \ldots, r.$$

The right-hand side vectors are

$$\mathbf{b}_i = \begin{bmatrix} \tilde{f}_{i,1} \\ \tilde{f}_{i,2} \\ \vdots \\ \vdots \\ \tilde{f}_{i,s} \end{bmatrix} - \begin{bmatrix} u_{i,0} \\ 0 \\ \vdots \\ 0 \\ u_{i,s+1} \end{bmatrix} \in \mathbb{R}^s, \qquad i = 2, \ldots, r-1,$$

and

$$\mathbf{b}_1 = \begin{bmatrix} \tilde{f}_{1,1} \\ \tilde{f}_{1,2} \\ \vdots \\ \tilde{f}_{1,s} \end{bmatrix} - \begin{bmatrix} u_{0,1} + u_{1,0} \\ u_{0,2} \\ \vdots \\ u_{0,s} + u_{1,s+1} \end{bmatrix}, \qquad \mathbf{b}_r = \begin{bmatrix} \tilde{f}_{r,1} \\ \tilde{f}_{r,2} \\ \vdots \\ \tilde{f}_{r,s} \end{bmatrix} - \begin{bmatrix} u_{r+1,1} + u_{r,0} \\ u_{r+1,2} \\ \vdots \\ u_{r+1,s} + u_{r,s+1} \end{bmatrix}.$$

The system has the condensed representation

$$A\mathbf{u} = \mathbf{b}$$

where

$$A = \begin{bmatrix} T & I_s & & \\ I_s & T & \ddots & \\ & \ddots & \ddots & I_s \\ & & I_s & T \end{bmatrix} \in \mathbb{R}^{p\times p}, \qquad \mathbf{u} = \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \vdots \\ \vdots \\ \mathbf{u}_r \end{bmatrix} \in \mathbb{R}^{p}, \qquad \mathbf{b} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \mathbf{b}_{r-1} \\ \mathbf{b}_r \end{bmatrix} \in \mathbb{R}^{p}.$$

The resulting square linear system of size p (number of pixels) can be solved by any general direct or preconditioned iterative method suited for large sparse matrices [83], or, specifically, by a fast Poisson solver [31, 33].

In practical photometric stereo, one usually focuses on the case of homogeneous Dirichlet boundary conditions, which corresponds to assuming that the observed object stands on a flat background. In this case the above description simplifies, considering that $\phi_1(x) = \phi_2(x) = \psi_1(y) = \psi_2(y) = 0$.

When the value of the function $u(x, y)$ on the boundary is unknown, there may be information on the normal derivatives of the solution. This amounts to imposing the following Neumann boundary conditions:

$$-\frac{\partial u(x, -B/2)}{\partial y} = \mu_1(x), \qquad \frac{\partial u(x, B/2)}{\partial y} = \mu_2(x), \qquad x \in [-A/2, A/2], \tag{4.3.8}$$

$$-\frac{\partial u(-A/2, y)}{\partial x} = \nu_1(y), \qquad \frac{\partial u(A/2, y)}{\partial x} = \nu_2(y), \qquad y \in [-B/2, B/2]. \tag{4.3.9}$$

Neumann boundary conditions are not sufficient to make the problem well posed. In fact, the solution is determined up to an additive constant, so the slope of a point of the solution has to be fixed arbitrarily.

In this case, the solution at the boundary is to be determined too, and the number of unknowns $u_{i,j}$ increases from rs to $p = (r + 1)(s + 1)$. Equation (4.3.6) is still valid for all the internal points of the grid, that is, for $i = 1, \dots, r$ and $j = 1, \dots, s$, but it has to be coupled with the discretization of the boundary conditions (4.3.8) and (4.3.9). To do this, we employed a one-sided second order discretization, obtaining on the horizontal boundaries of the domain

$$3u_{i,0} - 4u_{i,1} + u_{i,2} = \tilde{\mu}_{1,i},$$
$$u_{i,s-1} - 4u_{i,s} + 3u_{i,s+1} = \tilde{\mu}_{2,i},$$

with $\tilde{\mu}_{1,i} = 2h\mu_1(x_i)$ and $\tilde{\mu}_{2,i} = 2h\mu_2(x_i)$, for $i = 1, \dots, r$. Similarly, on the vertical boundaries, we get

$$3u_{0,j} - 4u_{1,j} + u_{2,j} = \tilde{\nu}_{1,j},$$
$$u_{r-1,j} - 4u_{r,j} + 3u_{r+1,j} = \tilde{\nu}_{2,j},$$

with $\tilde{\nu}_{1,j} = 2h\nu_1(y_j)$ and $\tilde{\nu}_{2,j} = 2h\nu_2(y_j)$, for $j = 1, \dots, s$. On the four corner points, we perform a linear interpolation from the three neighbour nodes of the grid. The equation for the corner with coordinates (x_0, y_0) is

$$u_{0,0} - u_{0,1} - u_{1,0} + u_{1,1} = 0.$$

Similar equations correspond to the other three corners of the rectangular domain.

The resulting linear system

$$B\mathbf{u} = \mathbf{c} \tag{4.3.10}$$

is defined by

$$B = \begin{bmatrix} P & Q & S & & \\ S & R & S & & \\ & \ddots & \ddots & \ddots & \\ & & S & R & S \\ & & S & Q & P \end{bmatrix}, \qquad \mathbf{u} = \begin{bmatrix} \mathbf{u}_0 \\ \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_r \\ \mathbf{u}_{r+1} \end{bmatrix}, \qquad \mathbf{c} = \begin{bmatrix} \mathbf{c}_0 \\ \mathbf{c}_1 \\ \vdots \\ \mathbf{c}_r \\ \mathbf{c}_{r+1} \end{bmatrix},$$

with

$$P = \begin{bmatrix} 1 & -1 & & & \\ & 3 & & & \\ & & \ddots & & \\ & & & 3 & \\ & & & -1 & 1 \end{bmatrix}, \qquad Q = \begin{bmatrix} -1 & 1 & & & \\ & -4 & & & \\ & & \ddots & & \\ & & & -4 & \\ & & & 1 & -1 \end{bmatrix},$$

$S = \operatorname{diag}(0, 1, \ldots, 1, 0)$, and

$$R = \begin{bmatrix} 3 & -4 & 1 & & \\ 1 & -4 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -4 & 1 \\ & & 1 & -4 & 3 \end{bmatrix}.$$

The expression of the right-hand side is the following:

$$\mathbf{c}_0 = \begin{bmatrix} 0 \\ \tilde{\nu}_{1,1} \\ \vdots \\ \tilde{\nu}_{1,s} \\ 0 \end{bmatrix}, \qquad \mathbf{c}_{r+1} = \begin{bmatrix} 0 \\ \tilde{\nu}_{2,1} \\ \vdots \\ \tilde{\nu}_{2,s} \\ 0 \end{bmatrix}, \qquad \mathbf{c}_i = \begin{bmatrix} \tilde{\mu}_{1,i} \\ \tilde{f}_{i,1} \\ \vdots \\ \tilde{f}_{i,s} \\ \tilde{\mu}_{2,i} \end{bmatrix}, \qquad i = 1, \ldots, r.$$

Remark 4.3.1. To make both the matrix B nonsingular and the solution unique, we substitute in (4.3.10) the equation associated to a chosen internal point of the domain, say, (x_i, y_j), with the equation $u_{i,j} = \gamma$, where γ is the slope assigned to that point.

4.4 Photometric stereo under unknown lighting

The need for the accurate localization of the light sources with respect to the observed object is a strong limitation for the practical application of the method described in the previous section, and the problem with unknown lights position is more involved. The *uncalibrated* photometric stereo problem consists in estimating the surface normal field from a set of pictures, without knowing in advance the direction of the light sources.

The first work for solving the photometric stereo problem under unknown lighting, proposed by Hayakawa [88], uses a singular value decomposition (SVD) technique to factorize the data matrix, i.e. the matrix containing the pictures. After the Hayakawa's approach, many methods have been proposed for the uncalibrated photometric stereo problem. In [34], the authors suggest an approach based on the use of low-order spherical harmonics for Lambertian objects, while [14] proposes a method based on the decomposition of the light intensity into a linear combination of spherical harmonics for recovering the 3D shape of objects by performing a simple optimization in a low-dimensional space.

Generally, uncalibrated photometric stereo is sensitive to deviations from ideality such as the presence of specular reflection and/or shadow. In [138] the authors proposed a robust SVD method in order to overcome this problem.

In this Section, we describe our implementation of the SVD based approach by Hayakawa for solving the uncalibrated photometric stereo problem. The method discussed below, assumes a sufficiently distant point light source. This idealized setting is often violated, e.g. by ambient light that is hard to prevent in a real setup. The ideas introduced for punctiform lights can be easily generalised to more complicated illumination conditions. Yuille and Snow [189] showed how the SVD approach can be extended to include ambient background illumination.

The photometric stereo technique under unknown lighting conditions consists of computing the rank-3 factorization

$$\widetilde{N}^T L = M, \tag{4.4.1}$$

where $\widetilde{N} = ND$ (see (4.3.2)), without knowing in advance the lights location. This problems has not a unique solution. Nevertheless, there are some physical constraints which allow one to find a meaningful solution.

Lemma 4.4.1. *The matrices D, N, and L, containing the albedo, the normals to the observed surface, and the lights directions, are determined up to a unitary transformation, that is, (4.4.1) is satisfied by the matrix pair $(Q\widetilde{N}, QL)$, for any orthogonal matrix $Q \in \mathbb{R}^{3\times 3}$.*

Proof. Any matrix pair $(A^{-T}\widetilde{N}, AL)$, with $A \in \mathbb{R}^{3\times 3}$ nonsingular, satisfies (4.4.1). Since the normal vectors $\mathbf{n}_k$ are normalized, the norm of the kth column of $\widetilde{N}$ equals the albedo ρ_k, while $\|\boldsymbol{\ell}_t\|$ is proportional to the light intensity. This implies that the transformation matrix A has to be orthogonal. □

The above Lemma suggests that the original orientation of the observed object cannot be determined without further information. This fact should be expected, since only the relative position between the object and the camera can be deduced from a set of images. This indetermination imposes some care in the shape reconstruction, because there is the possibility of axes reflections in the computation of the solution, which would alter significantly the shape of the reconstructed object.

In what follows, it is not restrictive to assume $\|\boldsymbol{\ell}_t\| = 1$, $t = 1, \ldots, q$. First of all, we already noticed that there is an undetermined proportionality constant in the problem, depending upon the unit of measure adopted for light intensity. Moreover, in the typical experimental setting the pictures are taken using a flashlight at a fixed distance from the object, which produces a constant light intensity across the observations. In particular situations the light intensity may vary, for example when the size of the object requires the

use of the sun as a light source, taking pictures at different times of the day. In this case a light meter can be used to obtain an estimate of $\|\boldsymbol{\ell}_t\|$.

Let the "compact" singular value decomposition (SVD) [83] of the observations matrix be

$$M = U\Sigma V^T, \tag{4.4.2}$$

where $\Sigma = \operatorname{diag}(\sigma_1, \ldots, \sigma_q)$ is the diagonal matrix containing the singular values and $U \in \mathbb{R}^{p\times q}$, $V \in \mathbb{R}^{q\times q}$ are matrices whose orthonormal columns $\mathbf{u}_i$ and $\mathbf{v}_i$ are the left and right singular vectors, respectively. In our application $q \ll p$, since the number of pixels in an image is usually very large, while we would like to obtain a reconstruction using a set of observations as small as possible. As we observed in the previous section, it is only required that $q \geq 3$.

When q is small (10–20), the SVD factorization can be computed efficiently by standard numerical libraries; we used the `svd` function of Matlab [129] even for a quite large value of p. In particular situations, in order to reduce the computation time, one may compute a partial singular value decomposition by an iterative method; see, e.g., [8, 9].

Since experimental data are always affected by noise, factorization (4.4.2) may have numerical rank $r > 3$, because of error propagation. In this case, a *truncated SVD* must be adopted, by setting $\sigma_4 = \cdots = \sigma_q = 0$. We let $W = [\sigma_1\mathbf{u}_1, \sigma_2\mathbf{u}_2, \sigma_3\mathbf{u}_3]^T$ and $Z = [\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3]^T$, so that $W^TZ \simeq M$. This choice produces the best rank-3 approximation to the data matrix M with respect to both the Euclidean and the Frobenius norm [18]. The constructive proof of the following theorem shows how to obtain the sought matrices $\widetilde{N}$ and L from this initial factorization.

Theorem 4.4.1. *The normal vectors to the observed surface and the lights position can be uniquely determined from* (4.3.2)*, up to a unitary transformation, only if at least 6 images taken in different lighting conditions are available.*

Proof. Let us consider the initial rank-3 factorization $W^TZ = M$ described above, with $W = [\mathbf{w}_1, \ldots, \mathbf{w}_p]$ and $Z = [\mathbf{z}_1, \ldots, \mathbf{z}_q]$. Given the assumption on the norms of the vectors $\boldsymbol{\ell}_t$, we first determine a matrix B such that $\|B\mathbf{z}_t\| = 1$ for each $t = 1, \ldots, q$. This implies solving the system of equations

$$\operatorname{diag}(Z^TGZ) = \mathbf{1}, \tag{4.4.3}$$

where $\mathbf{1} = (1, \ldots, 1)^T \in \mathbb{R}^q$ and $G = B^TB$ is a symmetric positive definite 3×3 matrix. The matrix G depends upon 6 independent parameters, say, its elements g_{ij} with $i \leq j$. As each equation in (4.4.3) is of the type

$$\mathbf{z}_t^TG\mathbf{z}_t = \sum_{i,j=1}^{3} z_{it}z_{jt}g_{ij} = 1,$$

the system (4.4.3) can be rewritten as the linear system

$$H\mathbf{g} = \mathbf{1},$$

where $\mathbf{g} = (g_{11}, g_{22}, g_{33}, g_{12}, g_{13}, g_{23})^T$ and H is a $q \times 6$ matrix, whose rows are

$$\begin{bmatrix} z_{1t}^2 & z_{2t}^2 & z_{3t}^2 & 2z_{1t}z_{2t} & 2z_{1t}z_{3t} & 2z_{2t}z_{3t} \end{bmatrix}, \qquad t = 1, \ldots, q.$$

A necessary condition for the solution vector $\mathbf{g}$ to be unique is that $q \geq 6$. This completes the proof. □

Remark 4.4.2. The requirement on the rank of the matrix H poses some constraints on the lights disposition. For example, a very common experimental approach is to place the light sources roughly on a circle around the camera, i.e., at a fixed distance δ from the origin, on a plane parallel to the observation plane. This is equivalent to fixing angles $\theta_1, \dots, \theta_q \in [0, 2\pi)$ and setting

$$\boldsymbol{\ell}_t = \frac{(\cos\theta_t, \sin\theta_t, \delta)^T}{\sqrt{1+\delta^2}}, \qquad t = 1, \dots, q.$$

This lights placement is not acceptable, because in this case the third column of the matrix H would be a linear combination of the first two. So at least one light source should violate this scheme. Placing the light sources at random positions around the object is often the safest way to ensure that H is full-rank.

The above theorem shows that at least 6 images are needed to reconstruct a shape by photometric stereo under unknown lighting, using this approach. Anyway, only a necessary condition for the unique solvability is given. In fact, H can be rank-deficient even for $q \geq 6$.

As the matrix B is determined up to a unitary transformation, we represent it by its QR factorization $B = QR$. The "essential" factor R can be obtained by the Cholesky factorization $G = R^T R$ [83], while Q cannot be uniquely determined; see Lemma 4.4.1. We will discuss a reasonable choice for the matrix Q in the following section.

Finally, factorization (4.4.1) is solved by setting

$$\widetilde{N} = QR^{-T}W \quad \text{and} \quad L = QRZ. \tag{4.4.4}$$

By normalizing the columns of $\widetilde{N}$ one obtains the diagonal albedo matrix D and the matrix N, whose columns are the normal vectors. The integration of the normal vector field is then performed by the approaches described in Section 4.3.

4.5 Determining the right orientation of the surface

As we already observed, the matrices $\widetilde{N}$ and L can be determined only up to a unitary transformation Q. It is nevertheless important to suitably choose Q, at least for two reasons.

First of all, the indetermination in the factorization (4.4.1) may introduce axes reflections in the reference system centered at the object, with the result of capsizing the direction of a part of the normal vectors.

Secondly, the integration procedures in Section 4.3 assumes that the function describing the shape of the object is single-valued and explicit, that is, has the form $z = u(x, y)$. The matrix Q must introduce a rotation of the reference system which meets this assumption.

We propose a computational procedure that, when coupled to a good practice in taking the pictures, leads to determining an effective rotation matrix Q. The shooting procedure consists of ordering the pictures by letting the light source rotate counterclockwise around the object, in the half space containing the camera. We assume that the first picture is taken with the light at the right hand of the camera, but this is not restrictive. The angles between the light positions do not have to be evenly spaced, and the light must not move exactly on a circle; see Remark 4.4.2. What is important, is that the light positions are distributed counterclockwise all around the object, and that the pictures are ordered according to this sequence.

After determining the matrices $\widehat{N} = R^{-T}W$ and $\widehat{L} = R^{-T}Z$ by the procedure described in Section 4.4, we consider the three columns $\hat{\boldsymbol{\ell}}_t$ of $\widehat{L}$, with $t = 1, \lfloor \frac{q}{3} \rfloor, \lfloor \frac{2q}{3} \rfloor$, where $\lfloor x \rfloor$ denotes the integer part of x. Given the above shooting procedure, this vector triplet must have a right-handed (or positive) orientation. This can be checked by the sign of the determinant of the matrix formed by the vector triplet. If the determinant is negative, then the factorization procedure introduced an axis reflection. To restore the original orientation we change the sign of the third row of $\widehat{N}$ and $\widehat{L}$, which corresponds to inverting the direction of the z axis.

Now we turn to determining the rotation matrix Q. To approximately identify the direction of the camera as seen from the observed object, we set

$$\mathbf{v}_3 = \sum_{t=1}^{q} \hat{\boldsymbol{\ell}}_t.$$

The vector $\mathbf{v}_3$ is assumed to be the direction of the z axis. The direction $\mathbf{v}_1$ of the x axis is obtained by projecting $\hat{\boldsymbol{\ell}}_1$ on the plane orthogonal to $\mathbf{v}_3$, and the y axis by computing the cross product $\mathbf{v}_2 = \mathbf{v}_3 \wedge \mathbf{v}_1$. After normalizing the three vectors, the orthogonal matrix

$$Q = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3 \end{bmatrix}$$

determines the sought rotation, which is used for computing the matrices $\widetilde{N}$ and L in (4.4.4).

4.6 Numerical experiments

In this section we illustrate the performance of our implemented method for the solution of the photometric stereo problem. All the computation were performed using Matlab 9.5 on a Debian GNU/Linux system. The 3D meshes were generated by our Matlab software, and visualized by the MeshLab open source system for processing and editing 3D triangular meshes (`www.meshlab.net`).

The algorithm described in the previous sections was tested both with synthetic and experimental data sets, in order to investigate its performance not only when the assumptions on which the algorithm is based are met, but also in a real-world setting, where the assumptions are only approximately verified.

4.6.1 Synthetic data set

In order to assess the accuracy attainable by the described computational approach in the ideal situation when all the assumptions of the methods are satisfied, we resorted to a synthetic dataset. We fixed a disposition of $q = 7$ light sources, placing them around the object at angles $(0, \frac{\pi}{4}, \frac{3\pi}{4}, \pi, \frac{5\pi}{4}, \frac{3\pi}{2}, \frac{7\pi}{4})$, and generated a set of digital images by applying the direct model (4.2.3) to the surface represented by the function

$$u(x, y) = \frac{1}{2} \mathrm{e}^{x} \sin(\pi x) \sin(\pi y), \tag{4.6.1}$$

on the square domain $[-1, 1] \times [-1, 1]$. Each image is 101×101 pixels, and the albedo equals $\frac{1}{2}$ for $x^2 + y^2 < \frac{1}{4}$, and 1 otherwise. Figure 4.1 shows both the synthetic surface and the corresponding data set.

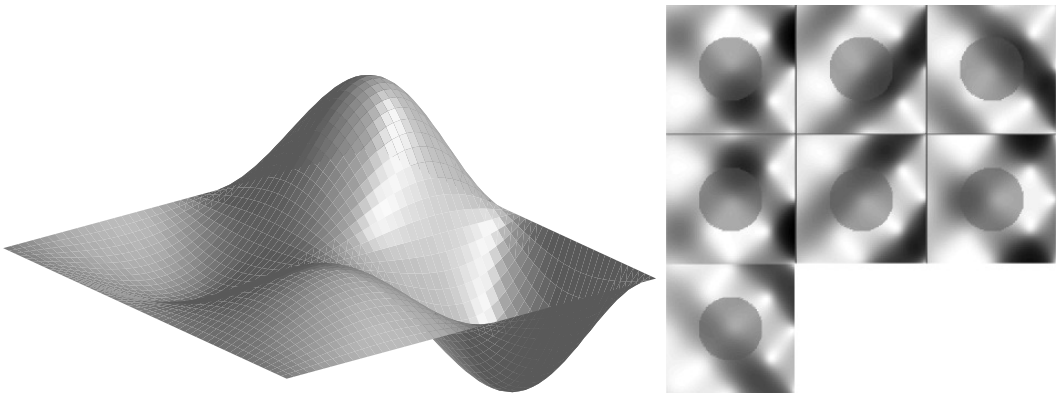


Figure 4.1: The graph on the left represents a synthetic surface used to investigate the performance of the algorithm; the data set used for the 3D reconstruction is composed by the 7 pictures on the right, each one corresponding to a different lighting condition.

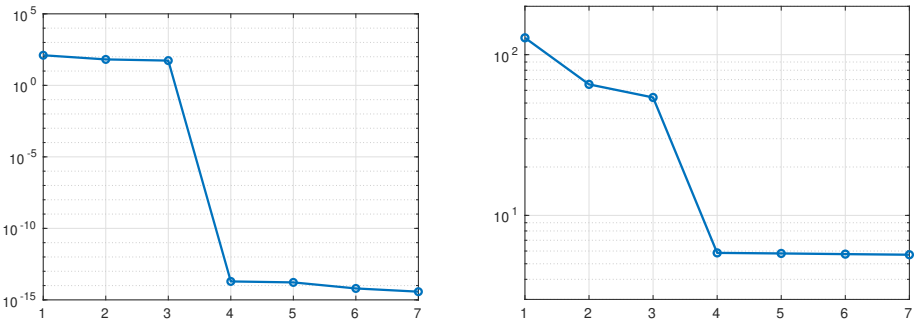


Figure 4.2: On the left, singular values of the data matrix M for the synthetic data set; on the right, singular values of the same matrix after 10% Gaussian noise is added.

The graph on the left of Figure 4.2 displays the singular values of the data matrix M from (4.3.2), showing that it clearly has numerical rank 3. Figure 4.3 shows the reconstruction of the light vectors and of the model surface. The light vectors are recovered up to machine precision, while the relative accuracy on the surface is $2.6 \cdot 10^{-4}$, in accord to the quite large step size $h = \frac{1}{50}$. The two errors are defined by

$$E_{\text{lights}} = \frac{\|L - \tilde{L}\|_F}{\|L\|_F}, \qquad E_{\text{surface}} = \frac{\|U - \tilde{U}\|_F}{\|U\|_F},$$

where $\|\cdot\|_F$ is the Frobenius norm, (L, U) denote the exact matrices containing the light vectors and the surface slopes, respectively, and $(\tilde{L}, \tilde{U})$ the reconstructed matrices.

We repeated the same test after introducing 10% Gaussian noise in the right-hand side M of (4.3.2). The presence of the noise is reflected in the singular values of M, depicted in the graph on the right of Figure 4.2. Though its rank is 7, it is evident that the matrix M can be well approximated by a rank 3 approximation. The computation is quite steady, as in this case $E_{\text{lights}} = 3.6 \cdot 10^{-3}$, while $E_{\text{surface}} = 1.5 \cdot 10^{-2}$.

The reconstruction in Figure 4.3 was obtained by imposing homogeneous Dirichlet boundary conditions, which are exactly verified by model function (4.6.1). The error

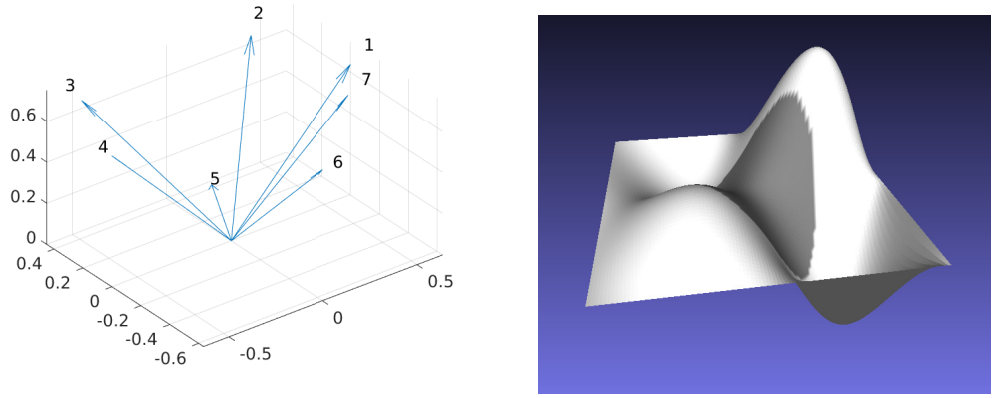


Figure 4.3: On the left, reconstruction of the light vectors, on the right, the recovered surface.

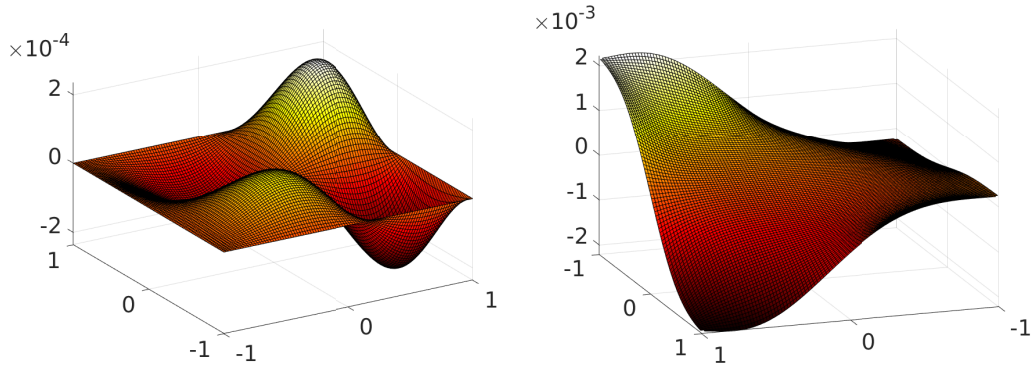


Figure 4.4: Reconstruction error with Dirichlet (left) and Neumann (right) boundary conditions.

$u(x, y) - \tilde{u}(x, y)$ between the model and the reconstruction is displayed in the graph on the left of Figure 4.4. The graph on the right of the same figure shows the error corresponding to exact Neumann conditions. In this case we fixed the value of the solution at the central point of the domain, that is, $\gamma = u(0, 0) = 0$; see Remark 4.3.1. Neumann conditions produced a less accurate reconstruction in proximity of the border of the domain, compared to Dirichlet conditions.

To investigate how deviation from ideal lighting influences numerical results, we repeated the above experiments, with Dirichlet conditions and without noise, positioning the light sources at a finite distance κA from the object, where κ is a scale factor and A is the horizontal width of the observed scene; in this particular example $A = 2$. The synthetic images were generated by a model based on Lambert's law, which considers incident light rays, rather than parallel rays. The light directions were recovered by the procedure described in Section 4.4.

When $\kappa < \infty$ the matrix M in (4.3.3) is full-rank, and this deteriorates the approximation accuracy of the rank-3 factorization constructed in Section 4.4. We measure the closeness of the matrix M to being rank-3 by the ratio σ_3/σ_4 between the third and the

Table 4.1: Influence of the distance between the object and the light source: the unit for the distance κ is the scene width, the ratio σ_3/σ_4 represents "closeness to rank 3", the errors are relative in the Frobenius norm.

κ	σ_3/σ_4	E_{lights}	E_{surface}
∞	$1.85 \cdot 10^{15}$	$1.00 \cdot 10^{-15}$	$2.69 \cdot 10^{-4}$
1000	$2.19 \cdot 10^{3}$	$1.95 \cdot 10^{-4}$	$1.39 \cdot 10^{-3}$
100	$2.18 \cdot 10^{2}$	$1.95 \cdot 10^{-3}$	$1.41 \cdot 10^{-2}$
10	$2.15 \cdot 10^{1}$	$1.95 \cdot 10^{-2}$	$1.45 \cdot 10^{-1}$
1	1.77	$4.52 \cdot 10^{-1}$	3.89

fourth singular value of M. Table 4.1 reports this ratio together with the errors E_{lights} and E_{surface}, for values of κ ranging from ∞ to 1. For example, when $\kappa = 10$ the distance of the light source from the origin is 10 times the width of the observed scene.

It is immediate to observe that when κ takes values close to 1, the error produced in the light vectors approximation is amplified in the object reconstruction, leading to unacceptable results. We observed that the algorithm may fail in some situation, as the deviation from ideality can lead to a non positive definite matrix G in (4.4.3), causing an unrecoverable error. This is one of the drawbacks of the algorithm that must be faces in future research.

The synthetic data set used in this experiment and the reconstructed object of Figure 4.4 are available at the web page `http://bugs.unica.it/cana/software/ps3d`, as a `.mat` file (Matlab data file), and a `.ply` 3D model file, respectively.

4.6.2 Experimental data sets

An experimental data set partially satisfying the assumptions required by the reconstruction method was generated as follows. A seashell (approximate width 10cm) was placed face up on a horizontal desk, with a tripod holding a camera about 1m above the seashell. A black background was placed below the seashell, to reproduce homogeneous Dirichlet boundary conditions for the observed surface. The desk was placed in the open air, under direct sunlight, and rotated in order to take 20 pictures of the seashell with different lighting directions, according to the shooting procedure described in Section 4.5. The resulting data set is displayed in Figure 4.5.

While the relatively small distance between the camera and the object produces images which cannot be represented through the orthographic projection model, the sunlight rays can be assumed to be parallel. So the lighting verifies the assumption of Lambert's model and we expect the data matrix M to be approximately rank-3.

The digital pictures were recorded in *raw* mode at the resolution of 3477×5220 pixels, and for this particular numerical simulation they were scaled to 885×705 pixels. The procedure described in Sections 4.4 and 4.5 identified 20 light vectors, displayed in the graph on the right in Figure 4.5, that are compatible with the sunlight positions during the shooting process.

Integrating the normal field by the Poisson equation (4.3.5), and imposing homogeneous Dirichlet boundary conditions, one obtains the 3D model illustrated in Figure 4.6 by three different views. Compared to the original, the model appears slightly deformed by the deviation from the orthographic projection model, but the reconstruction is quite accurate and the computing time negligible, 2.8 seconds on an Intel Core i7 computer. The deformation

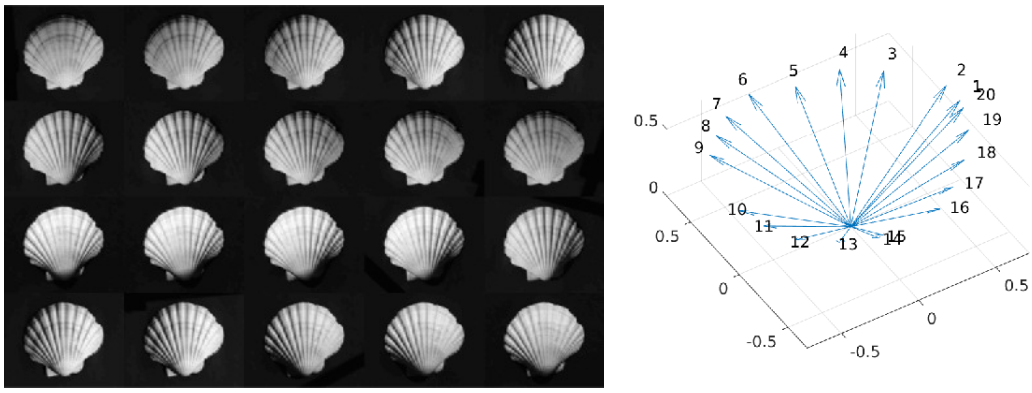


Figure 4.5: The SHELL data set, on the left, consisting in 20 images corresponding to different sunlight directions; on the right we report the light directions identified by the reconstruction algorithm.

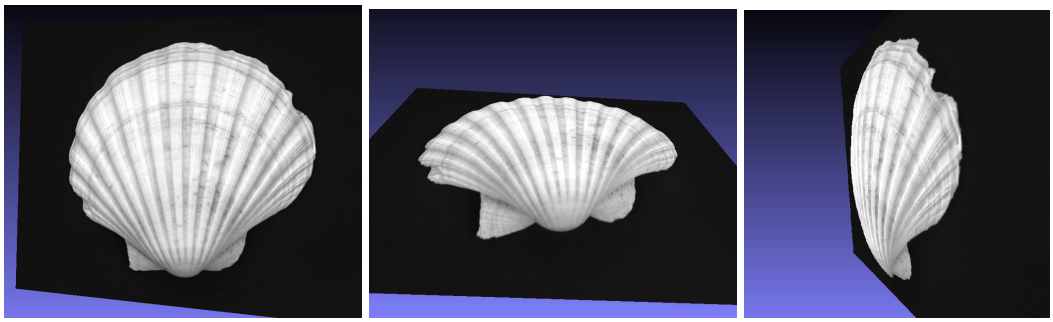


Figure 4.6: Three different views of the 3D surface reconstructed from the SHELL data set.

induced by the camera system could be corrected by camera calibration techniques. The SHELL data set and the reconstructed object are available at the web page `http://bugs.unica.it/cana/software/ps3d`.

As a second experiment based on a real data set, we processed the images of a stela of the Ptolemaic period, exhibited at the Museo Egizio in Torino, Italy (`https://www.museoegizio.it`); see Figure 4.7. We thank the Museo Egizio, in particular Christian Greco, director of the museum, and Marco Rossani, collection manager, for providing us the data set, which is composed by 8 images. A black mask was added around the stela in each image, in order to reproduce Dirichlet boundary conditions. The resolution of the *raw* digital pictures is 7360×4912 pixels. The images were scaled to 1474×2208 pixels to produce the result depicted on the right of Figure 4.7, where the surface is displayed after removing the albedo. The computation took 16 seconds.

The pictures were taken in the exposition room, using an electronic flash as a light source, so they do not satisfy the assumptions of the method, as both the camera and the light source were at a quite small distance from the object. Indeed, while the bas-relief details are accurately reproduced, the reconstructed surface is spherically warped, compared to the flatness of the real stela.

Some details of the reconstructed surface are displayed in Figure 4.8. The left and

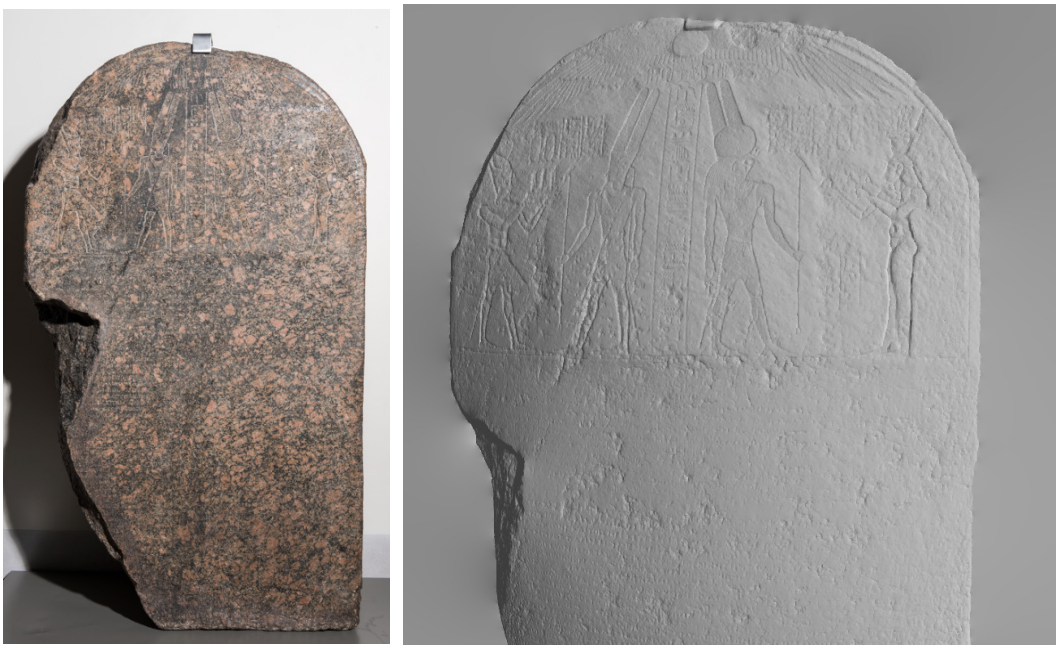


Figure 4.7: On the left, stela in honor of the general Callimachos, mentioning Cleopatra and Caesarion; granite, Ptolemaic period, reign of Cleopatra VII, 39 BC. Thebes, Temple of Karnak (Courtesy of Museo Egizio, Torino, Italy). On the right, 3D reconstruction obtained by the algorithm described in the paper.

central pictures show how albedo removal can lead to a cleaner visualization of an engraving. The image on the right is a part of the reconstruction obtained by processing a 800×800 subimage of the original high resolution pictures. It shows a writing, located in the central part of the stela. To obtain a neat representation of small writings one would need high resolution images only of this part, but this would introduce difficulties in assigning boundary conditions to the Poisson equation. We will face this problem in future work.

4.7 Conclusions and future work

In this chapter we consider an implementation of a method for solving the photometric stereo problem under unknown lighting showing that, under the ideal assumptions upon which Lambert's model is based, it is possible to estimate the lights position if at least 6 images of an object are available, each one with a different light source position. We show that this information can be effectively used to reconstruct a 3D model of the observed object, and propose a procedure to avoid the indetermination in the direction of the normal vectors, which is typical of photometric stereo.

The accuracy of the considered method is investigated through numerical experiments on both synthetic and real data sets. Our experiments show that the algorithm is accurate under ideal conditions, and that the lack from ideality produces, as expected, a spherical deformation on the reconstructed surfaces.

Our future research work will be devoted to improving the method, in order to make

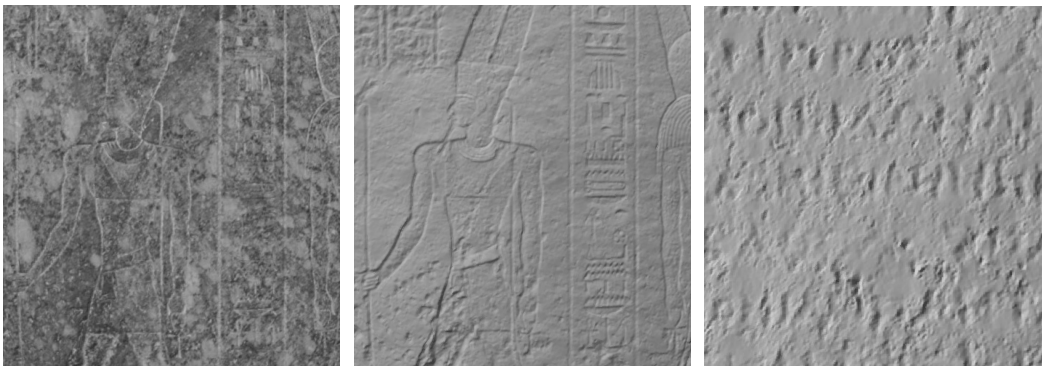


Figure 4.8: Three details of the Ptolemaic stela reconstruction: the first two show the same area with and without albedo, the third one a part containing a writing.

it applicable to real shooting conditions, in particular when the light sources cannot be positioned sufficiently far away from the observed object. Moreover, real time processing of high resolution pictures requires a reduction in the computing time, especially for what concerns the data matrix factorization and the solution of the final large linear system. Another aspect that needs further work is the treatment of the boundary conditions required for the solution of equation (4.3.5), whose knowledge is not available in many applicative situations.

Conclusion

In this last part we summarise the results described in the whole Thesis, and focus on some topics we are working on and planning to develop in future.

We have explored some of the applications of Numerical Linear Algebra in the field of Archaeology. In particular, we have proposed a Matlab toolbox for solving the seriation problem with possible application to all the other problems involving the sorting of similarity (or dissimilarity) data. Our software contains an implementation of the PQ-tree data structure which encodes all the possible orderings that lead to a solution of the problem. In case of *perfect data*, the spectral algorithm uses the entries of the Fiedler vector of the Laplacian matrix associated to the problem for finding all the possible reordering of the nodes (representing the units we want to reorder) in the considered seriation problem. When the data are inconsistent (or *imperfect*) the seriation problem belongs to the NP-hard complexity class and the spectral algorithm can be applied as a heuristic to find an approximate solution. In the presence of imperfect seriation data, we highlight that if the Laplacian matrix associated to the problem has a multiple Fiedler value then this situation may have influence on the computation of an approximate solution to the seriation problem.

Regarding the **seriation problem**, future work involves the study of seriation in a noisy setting. In particular, we are currently working on the situation when the Laplacian of the similarity matrix associated with the considered seriation problem, has multiple Fiedler value with the aim to delineate the possible constraints to the solution of the problem. Still in a noisy setting, another line of research is based on investigating the asymptotic behaviour of a block matrix representing a bipartite graph associated to a given seriation problem, constructed from matrices obtained by properly normalizing the rows and columns of the given binary data matrix. The purpose is to construct a spectral algorithm for solving the seriation problem in the presence of imperfect data, by using the approach, described in Chapter 3, used for approximating a given network by a bipartite graph.

Besides the study of the seriation problem in noisy settings, we are planning to work also on seriation by node metrics.

In regards to the **spectral bipartization method** presented in Chapter 3, another future development concerns its generalization for discovering approximate multi-partite structure.

In relation to the **photometric stereo problem** we have examined a standard approach for its solution, based on the Poisson formulation of the problem. We have then explored the PS problem under unknown lighting, showing that it is possible to estimate the lights location if at least 6 images of an object are available, each one corresponding to a different light direction. Future work will be devoted to improving the method, in order to make it applicable in real situations when the lights cannot be positioned sufficiently far away from the object whose 3d shape has to be reconstruct. Another aspect that need to be further investigated, concern a reduction in the computing time required by the processing of high

resolution pictures and the treatment of the boundary conditions required for the solution of the Poisson equation considering for example *reflective* boundary conditions.

Bibliography

[1] J. Ackermann and M. Goesele. "A survey of photometric stereo techniques". In: *Foundations and Trends® in Computer Graphics and Vision* 9.3-4 (2015), pp. 149–254.

[2] A. J. B. Anderson. "Ordination methods in ecology". In: *The Journal of Ecology* (1971), pp. 713–726.

[3] M. J. Anderson and T. J. Willis. "Canonical analysis of principal coordinates: a useful method of constrained ordination for ecology". In: *Ecology* 84.2 (2003), pp. 511–525.

[4] N. Archip, R. Rohling, P. Cooperberg, H. Tahmasebpour, and S.K. Warfield. "Spectral clustering algorithms for ultrasound image segmentation". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2005, pp. 862–869.

[5] A. S. Asratian, T. M. J. Denley, and R. Häggkvist. *Bipartite graphs and their applications*. Vol. 131. Cambridge university press, 1998.

[6] J. E. Atkins, E. G. Boman, and B. Hendrickson. "A spectral algorithm for seriation and the consecutive ones problem". In: *SIAM J. Comput.* 28.1 (1998), pp. 297–310.

[7] G. A. Atkinson and E. R. Hancock. "Recovery of surface orientation from diffuse polarization". In: *IEEE transactions on image processing* 15.6 (2006), pp. 1653–1664.

[8] J. Baglama and L. Reichel. "An implicitly restarted block Lanczos bidiagonalization method using Leja shifts". In: *BIT* 53 (2013), pp. 285–310.

[9] J. Baglama and L. Reichel. "Augmented implicitly restarted Lanczos bidiagonalization methods". In: *SIAM J. Sci. Comput.* 27 (2005), pp. 19–42.

[10] R. B. Bapat. *Graphs and matrices*. Vol. 27. Springer, 2010.

[11] J. A. Barcelo and I. Bogdanovic. *Mathematics and Archaeology*. CRC Press, 2015.

[12] S. T. Barnard, A. Pothen, and H. Simon. "A spectral algorithm for envelope reduction of sparse matrices". In: *Numer. Linear Algebra Appl.* 2.4 (1995), pp. 317–334.

[13] S. Barsky and M. Petrou. "The 4-source photometric stereo technique for three-dimensional surfaces in the presence of highlights and shadows". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 25.10 (2003), pp. 1239–1252.

[14] R. Basri, D. Jacobs, and I. Kemelmacher. "Photometric stereo with general, unknown lighting". In: *Int. J. Comput. Vis.* 72.3 (2007), pp. 239–257.

[15] E. Beltrami. “Sulle funzioni bilineari”. In: *Giornale di Matematiche ad Uso degli Studenti Delle Universita* 11.2 (1873), pp. 98–106.

[16] A. Berman and R. J. Plemmons. *Nonnegative matrices in the mathematical sciences*. Vol. 9. Siam, 1994.

[17] N. Biggs, N. L. Biggs, and B. Norman. *Algebraic graph theory*. Vol. 67. Cambridge university press, 1993.

[18] Å. Björck. *Numerical Methods for Least Squares Problems*. Philadelphia: SIAM, 1996. ISBN: 0-89871-360-9.

[19] M. Bóna. *Combinatorics of permutations*. Chapman and Hall/CRC, 2016.

[20] J. A. Bondy and U. S. R. Murty. *Graph theory with applications*. Vol. 290. Macmillan London, 1976.

[21] K. S. Booth and G. S. Lueker. “Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms”. In: *J. Comput. Syst. Sci.* 13.3 (1976), pp. 335–379.

[22] S. P. Borgatti and M. G. Everett. “Models of core/periphery structures”. In: *Social networks* 21.4 (2000), pp. 375–395.

[23] A. G. Bors, E. R. Hancock, and R. C. Wilson. “Terrain analysis using radar shape-from-shading”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25.8 (2003), pp. 974–992.

[24] D. P. Bovet and P. Crescenzi. *Introduction to the Theory of Complexity*. Prentice Hall London, 1994.

[25] G. W. Brainerd. “The place of chronological ordering in archaeological analysis”. In: *Am. Antiq.* 16.4 (1951), pp. 301–313.

[26] J. C. Brower and K. M. Kile. “Seriation of an original data matrix as applied to paleoecology”. In: *Lethaia* 21.1 (1988), pp. 79–93.

[27] R. A. Brualdi. *Introductory combinatorics*. Pearson Education India, 1977.

[28] M. J. Brusco and S. Stahl. *Branch-and-bound applications in combinatorial data analysis*. Springer Science & Business Media, 2006.

[29] M. J. Brusco and D. Steinley. “Clustering, seriation, and subset extraction of confusion data”. In: *Psychol. Methods* 11.3 (2006), pp. 271–286.

[30] D. Bu, Y. Zhao, L. Cai, H. Xue, X. Zhu, H. Lu, J. Zhang, S. Sun, L. Ling, and N. Zhang. “Topological structure analysis of the protein–protein interaction network in budding yeast”. In: *Nucleic acids research* 31.9 (2003), pp. 2443–2450.

[31] B. L. Buzbee, G. H. Golub, and C. W. Nielson. “On direct methods for solving Poisson’s equations”. In: *SIAM Journal Numer. Anal.* 7.4 (1970), pp. 627–656.

[32] G. Caraux and S. Pinloche. “PermutMatrix: a graphical environment to arrange gene expression profiles in optimal linear order”. In: *Bioinformatics* 21.7 (1979). Package available at http://www.atgc-montpellier.fr/permutmatrix/, pp. 1280–1281.

[33] T. F. Chan and D. C. Resasco. “A domain-decomposed fast Poisson solver on a rectangle”. In: *SIAM J. Sci. Stat. Comput.* 8.1 (1987), s14–s26.

[34] C.-P. Chen and C.-S. Chen. “The 4-source photometric stereo under general unknown lighting”. In: *Computer Vision–ECCV 2006*. Vienna: Springer, 2006, pp. 72–83.

[35] L. Chen, Q. Yu, and B. Chen. “Anti-modularity and anti-community detecting in complex networks”. In: *Information Sciences* 275 (2014), pp. 293–313.

[36] V. Chepoi and B. Fichet. “Recognition of Robinsonian dissimilarities”. In: *J. Classif.* 14.2 (1997), pp. 311–325.

[37] P. H. Christensen and L. G. Shapiro. “Three-dimensional shape from color photometric stereo”. In: *Int. J. Comput. Vis.* 13.2 (1994), pp. 213–227.

[38] T. Christof, M. Oswald, and G. Reinelt. “Consecutive ones and a betweenness problem in computational biology”. In: *International Conference on Integer Programming and Combinatorial Optimization*. Springer. 1998, pp. 213–228.

[39] F. R. K. Chung and F. C. Graham. *Spectral graph theory*. 92. American Mathematical Soc., 1997.

[40] E. N. Coleman Jr. and R. Jain. “Obtaining 3-dimensional shape of textured and specular surfaces using four-source photometry”. In: *Computer graphics and image processing* 18.4 (1982), pp. 309–328.

[41] A. Concas, C. Fenu, and G. Rodriguez. “PQser: a Matlab package for spectral seriation”. In: *Numerical Algorithms* 80.3 (2019), pp. 879–902.

[42] A. Concas, S. Noschese, L. Reichel, and G. Rodriguez. “A spectral method for bipartizing a network and detecting a large anti-community”. In: *Journal of Computational and Applied Mathematics* (2019).

[43] E. Cuthill and J. McKee. “Reducing the bandwidth of sparse symmetric matrices”. In: *Proceedings of the 1969 24th national conference*. ACM. 1969, pp. 157–172.

[44] D. Cvetković, P. Rowlinson, and S. K. Simić. “Signless Laplacians of finite graphs”. In: *Linear Algebra and its applications* 423.1 (2007), pp. 155–171.

[45] D. Cvetković, S. Simic, and P. Rowlinson. *An introduction to the theory of graph spectra*. Cambridge University Press, 2009.

[46] D. Cvetković, D. M. Cvetković, P. Rowlinson, and S. Simic. *Eigenspaces of graphs*. 66. Cambridge University Press, 1997.

[47] D. M. Cvetković, M. Doob, and H. Sachs. *Spectra of graphs*. Vol. 10. Academic Press, New York, 1980.

[48] P. J. Davis. *Circulant Matrices*. New York: Wiley, 1979.

[49] N. M. M. De Abreu. “Old and new results on algebraic connectivity of graphs”. In: *Linear Algebra Appl.* 423.1 (2007), pp. 53–73.

[50] P. Dempsey and M. Baumhoff. “The statistical use of artifact distributions to establish chronological sequence”. In: *American Antiquity* 28.4 (1963), pp. 496–509.

[51] M. Desai and V. Rao. “A characterization of the smallest eigenvalue of a graph”. In: *Journal of Graph Theory* 18.2 (1994), pp. 181–194.

[52] R. Dessì, C. Mannu, G. Rodriguez, G. Tanda, and M. Vanzi. “Recent improvements in photometric stereo for rock art 3D imaging”. In: *Digital Applications in Archaeology and Cultural Heritage (DAACH)* 2 (2015), pp. 132–139.

[53] S.B. Deutsch and J. J. Martin. “An ordering algorithm for analysis of data arrays”. In: *Operations Research* 19.6 (1971), pp. 1350–1362.

[54] R. Diestel. “Graph theory. 2005”. In: *Grad. Texts in Math* 101 (2005).

[55] C. Ding and X. He. “Linearized cluster assignment via spectral ordering”. In: *Proceedings of the twenty-first international conference on Machine learning*. ACM. 2004, p. 30.

[56] M. Dom. “Algorithmic aspects of the consecutive-ones property”. In: (2009).

[57] W. E. Donath and A. J. Hoffman. “Algorithms for partitioning of graphs and computer logic based on eigenvectors of connection matrices”. In: *IBM Technical Disclosure Bulletin* 15.3 (1972), pp. 938–944.

[58] J. E. Doran, J. Doran, and F. R. Hodson. *Mathematics and computers in archaeology*. Harvard University Press, 1975.

[59] J.-D. Durou, M. Falcone, and M. Sagona. “Numerical methods for shape-from-shading: A new survey with benchmarks”. In: *Computer Vision and Image Understanding* 109.1 (2008), pp. 22–43.

[60] C. R. Dyer. “Volumetric scene reconstruction from multiple views”. In: *Foundations of Image Understanding*. Vienna: Springer, 2001, pp. 469–489.

[61] D. Earle. “Dendrogram seriation in data visualisation: algorithms and applications”. PhD thesis. National University of Ireland Maynooth, 2010.

[62] C. Eckart and G. Young. “The approximation of one matrix by another of lower rank”. In: *Psychometrika* 1.3 (1936), pp. 211–218.

[63] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. “Cluster analysis and display of genome-wide expression patterns”. In: *P. Natl. Acad. Sci. U.S.A.* 95 (1998), pp. 14863–14868.

[64] E. Estrada. *The Structure of Complex Networks: Theory and Applications*. Oxford University Press, 2012.

[65] E. Estrada and J. Gómez-Gardeñes. “Network bipartivity and the transportation efficiency of european passenger airlines”. In: *Physica D: Nonlinear Phenomena* 323 (2016), pp. 57–63.

[66] E. Estrada and D. J. Higham. “Network properties revealed through matrix functions”. In: *SIAM Rev.* 52.4 (2010), pp. 696–714.

[67] E. Estrada and P. Knight. *A First Course in Network Theory*. Oxford University Press, 2015.

[68] E. Estrada and J. A. Rodríguez-Velázquez. “Spectral measures of bipartivity in complex networks”. In: *Physical Review E* 72.4 (2005), p. 046105.

[69] D. Fasino and F. Tudisco. “A modularity based spectral method for simultaneous community and anti-community detection”. In: *Linear Algebra and its Applications* 542 (2018), pp. 605–623.

[70] M. Fiedler. “A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory”. In: *Czech. Math. J.* 25.4 (1975), pp. 619–633.

[71] M. Fiedler. “Algebraic connectivity of graphs”. In: *Czech. Math. J.* 23.2 (1973), pp. 298–305.

[72] M. Fiedler. “Laplacian of graphs and algebraic connectivity”. In: *Banach Center Publ.* 25.1 (1989), pp. 57–70.

[73] F. Fogel, A. d’Aspremont, and M. Vojnovic. “Serialrank: Spectral ranking using seriation”. In: *Advances in Neural Information Processing Systems 27*. 2014, pp. 900–908.

[74] F. Fogel, R. Jenatton, F. Bach, and A. d’Aspremont. “Convex relaxations for permutation problems”. In: *SIAM J. Matrix Anal. Appl.* 36.4 (2015), pp. 1465–1488.

[75] E. Forsyth and L. Katz. “A matrix approach to the analysis of sociometric data: preliminary report”. In: *Sociometry* 9.4 (1946), pp. 340–347.

[76] D. Fulkerson and O. Gross. “Incidence matrices and interval graphs”. In: *Pac. J. Math.* 15.3 (1965), pp. 835–855.

[77] A. George. *Computer implementation of the finite element method*. Tech. rep. STANFORD UNIV CA DEPT OF COMPUTER SCIENCE, 1971.

[78] A. George and A. Pothen. “An analysis of spectral envelope reduction via quadratic assignment problems”. In: *SIAM Journal on Matrix Analysis and Applications* 18.3 (1997), pp. 706–732.

[79] J. J. Gibson. “The perception of the visual world.” In: (1950).

[80] D. Gleich. *MatlabBGL - A Matlab Graph Library*. URL: https://www.cs.purdue.edu/homes/dgleich/packages/matlab_bgl/.

[81] C. Godsil and G.F. Royle. *Algebraic graph theory*. Vol. 207. Springer Science & Business Media, 2013.

[82] G. H. Golub, A. Hoffman, and G. W. Stewart. “A generalization of the Eckart-Young-Mirsky matrix approximation theorem”. In: *Linear Algebra and its applications* 88 (1987), pp. 317–327.

[83] G. H. Golub and C. F. Van Loan. *Matrix Computation*. 1989.

[84] D. S. Greenberg and S. Istrail. “Physical mapping by STS hybridization: Algorithmic strategies and the challenge of software evaluation”. In: *J. Comput. Biol.* 2.2 (1995), pp. 219–273.

[85] M. Hahsler, K. Hornik, and C. Buchta. “Getting things in order: an introduction to the R package seriation”. In: *J. Stat. Softw.* 25.3 (2008), pp. 1–34.

[86] Ø. Hammer, D. A.T. Harper, and P. D. Ryan. “PAST: paleontological statistics software package for education and data analysis”. In: *Palaeontologia electronica* 4.1 (2001), p. 9.

[87] T. C. Havens and J. C. Bezdek. “An efficient formulation of the improved visual assessment of cluster tendency (iVAT) algorithm”. In: *IEEE Transactions on Knowledge and Data Engineering* 24.5 (2011), pp. 813–822.

[88] H. Hayakawa. “Photometric stereo under a light source with arbitrary motion”. In: *JOSA A* 11.11 (1994), pp. 3079–3089.

[89] S. E. M. Herrera, A. Malti, O. Morel, and A. Bartoli. “Shape-from-Polarization in laparoscopy”. In: *2013 IEEE 10th International Symposium on Biomedical Imaging*. IEEE. 2013, pp. 1412–1415.

[90] D. J. Higham, G. Kalna, and M. Kibble. “Spectral clustering and its use in bioinformatics”. In: *J. Comput. Appl. Math.* 204.1 (2007), pp. 25–37.

[91] N. J. Higham. *Matrix nearness problems and applications*. Citeseer, 1988.

[92] F. R. Hodson, D. G. Kendall, and P. Tautu. *Mathematics in the Archaeological and Historical Sciences*. Edimburg: Edimburg University Press, 1971.

[93] W. Hoff and N. Ahuja. “Surfaces from stereo: Integrating feature matching, disparity estimation, and contour detection”. In: *IEEE transactions on pattern analysis and machine intelligence* 11.2 (1989), pp. 121–136.

[94] F. Hole and M. Shaw. *Computer analysis of chronological seriation*. Vol. 53. 3. Rice University, 1967.

[95] P. Holme, F. Liljeros, C. R. Edling, and B. J. Kim. “Network bipartivity”. In: *Physical Review E* 68.5 (2003), p. 056107.

[96] B. K. P. Horn. “Height and gradient from shading”. In: *International journal of computer vision* 5.1 (1990), pp. 37–75.

[97] B. K. P. Horn. “Obtaining shape from shading information”. In: *Shape from Shading*. MIT Press. 1989, pp. 123–171.

[98] B. K. P. Horn. “Shape from shading: A method for obtaining the shape of a smooth opaque object from one view”. In: (1970).

[99] L. Hubert. “Seriation using asymmetric proximity measures”. In: *British Journal of Mathematical and Statistical Psychology* 29.1 (1976), pp. 32–52.

[100] L. Hubert, P. Arabie, and J. Meulman. *Combinatorial data analysis: Optimization by dynamic programming*. Vol. 6. SIAM, 2001.

[101] P. Ihm. “A Contribution to the History of Seriation in Archaeology”. In: *Classification—the Ubiquitous Challenge*. Springer, 2005, pp. 307–316.

[102] H. Jeong, S. P. Mason, A. L. Barabási, and Z. N. Oltvai. “Lethality and centrality in protein networks”. In: *Nature* 411.6833 (2001), p. 41.

[103] C. Jordan. “Mémoire sur les formes bilinéaires.” In: *Journal de mathématiques pures et appliquées* 19 (1874), pp. 35–54.

[104] L. Jørgensen. *Bækkegård and Glasergård: two cemeteries from the Late Iron Age on Bornholm*. Vol. 8. Akademisk forlag, 1990.

[105] L. Jørgensen. “En kronologi for yngre romersk og ældre germansk jernalder på Bornholm. Jørgensen, Lars”. In: *Simblegård-Trelleborg. Danske gravfund fra førromersk jernalder til vikingetid* (1989).

[106] M. Juvan and B. Mohar. “Optimal linear labelings and eigenvalues of graphs”. In: *Discrete Appl. Math.* 36.2 (1992), pp. 153–168.

[107] D. G. Kendall. “A mathematical approach to seriation”. In: *Philos. Trans. R. Soc. A-Math. Phys. Eng. Sci.* 269.1193 (1970), pp. 125–134.

[108] D. G. Kendall. “A statistical approach to Flinders–Petries sequence-dating”. In: *Bull. Int. Stat. Inst.* 40.2 (1963), pp. 657–681.

[109] D. G. Kendall. “Incidence matrices, interval graphs and seriation in archeology”. In: *Pac. J. Math.* 28.3 (1969), pp. 565–570.

[110] D. G. Kendall, F. R. Hodson, and P. Tautu. “Mathematics in the Archaeological and historical sciences”. In: *Edinburgh Uni* (1971).

[111] R. Klette, K. Schlüns, and A. Koschan. *Computer Vision: Three-dimensional Data from Images*. Singapore: Springer, 1998.

[112] J. J. Koenderink. “The structure of images”. In: *Biological cybernetics* 50.5 (1984), pp. 363–370.

[113] D. König. “Über graphen und ihre anwendung auf determinantentheorie und mengenlehre”. In: *Mathematische Annalen* 77.4 (1916), pp. 453–465.

[114] T. C. Koopmans and M. Beckmann. “Assignment problems and the location of economic activities”. In: *Econometrica: journal of the Econometric Society* (1957), pp. 53–76.

[115] R. Kozera. “Existence and uniqueness in photometric stereo”. In: *Appl. Math. Comput.* 44.1 (1991), pp. 1–103.

[116] F. A. Kuentzer, A. S. Pereira, A. M. Amory, G. Perrone, S. R. M. Silva, J. .M. Dinis, and R. .M. C. Almeida. “Optimization and analysis of seriation algorithm for ordering protein networks”. In: *2014 IEEE International Conference on Bioinformatics and Bioengineering*. IEEE. 2014, pp. 231–237.

[117] R. S. Kuzara, G. R. Mead, and K. A. Dixon. “Seriation of Anthropological Data: A Computer Program for Matrix-Ordering”. In: *American Anthropologist* 68.6 (1966), pp. 1442–1455.

[118] S. Lang. *Linear Algebra*. Springer, 1992.

[119] M. Laurent and M. Seminaroti. “A Lex-BFS-based recognition algorithm for Robinsonian matrices”. In: *Discret. Appl. Math.* 222 (2017), pp. 151–165.

[120] M. Laurent and M. Seminaroti. “Similarity-First Search: a new algorithm with application to Robinsonian matrix recognition”. In: *SIAM Discret. Math.* 31.3 (2017), pp. 1765–1800.

[121] M. Laurent and M. Seminaroti. “The quadratic assignment problem is easy for Robinsonian matrices with Toeplitz structure”. In: *Operations Research Letters* 43.1 (2015), pp. 103–109.

[122] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney. “Statistical properties of community structure in large social and information networks”. In: *Proceedings of the 17th international conference on World Wide Web*. ACM. 2008, pp. 695–704.

[123] I. Liiv. “Seriation and matrix reordering methods: an historical overview”. In: *Stat. Anal. Data Min.* 3.2 (2010), pp. 70–91.

[124] I. Liiv, R. Opik, J. Ubi, and J. Stasko. “Visual matrix explorer for collaborative seriation”. In: *Wiley Interdisciplinary Reviews: Computational Statistics* 4.1 (2012), pp. 85–97.

[125] W. H. Liu and A. H. Sherman. “Comparative analysis of the Cuthill–McKee and the reverse Cuthill–McKee ordering algorithms for sparse matrices”. In: *SIAM Journal on Numerical Analysis* 13.2 (1976), pp. 198–213.

[126] P. M. Magwene, P. Lizardi, and J. Kim. “Reconstructing the temporal ordering of biological samples using microarray data”. In: *Bioinformatics* 19.7 (2003), pp. 842–850.

[127] C. Mannu, G. Rodriguez, G. Tanda, and M. Vanzi. “Nuovi sviluppi nelle tecniche di stereofotometria 3D di incisioni e rilievi. Applicazioni nella tomba XV di Sos Furrighesos, Sardegna”. In: *Prospects for the Prehistoric Art Research, 50 Years Since the Founding of Centro Camuno. Proceedings of the XXVI Valcamonica Symposium, September 9–12, 2015*. Ed. by F. Troletti. ISBN: 978-1-4673-1159-5. Capo di Ponte, Italy: Centro Camuno di Studi Preistorici, 2015, pp. 285–288.

[128] W. H. Marquardt. “Advances in archaeological seriation”. In: *Advances in archaeological method and theory*. Elsevier, 1978, pp. 257–314.

[129] *Matlab ver. 8.4*. The MathWorks, Inc. Natick, MA, 2014.

[130] D. Mavroeidis and E. Bingham. “Enhancing the stability and efficiency of spectral ordering with partial supervision and feature selection”. In: *Knowledge and information systems* 23.2 (2010), pp. 243–265.

[131] W. T. Jr McCormick, S. B. Deutsch, J. J. Martin, and P. J. Schweitzer. “Identification of Data Structures and Relationships by Matrix Reordering Techniques”. In: *ERIC* (1969).

[132] R. Mecca and J.-D. Durou. “Unambiguous photometric stereo using two images”. In: *International Conference on Image Analysis and Processing*. Springer. 2011, pp. 286–295.

[133] R. Mecca and M. Falcone. “Uniqueness and approximation of a photometric shape-from-shading model”. In: *SIAM J. Imaging Sci.* 6 (2013), pp. 616–659.

[134] J. Meidanis and E. G. Munuera. “A simple linear time algorithm for binary phylogeny”. In: *Proc. of the XV International Conference of the Chilean Computing Society*. 1995, pp. 275–283.

[135] J. Meidanis, O. Porto, and G. P. Telles. “On the consecutive ones property”. In: *Discrete Applied Mathematics* 88.1-3 (1998), pp. 325–354.

[136] B. G. Mirkin and S. N. Rodin. *Graphs and Genes*. Vol. 11. Biomathematics. Springer-Verlag, 1984.

[137] L. Mirsky. “Symmetric gauge functions and unitarily invariant norms”. In: *The quarterly journal of mathematics* 11.1 (1960), pp. 50–59.

[138] D. Miyazaki and K. Ikeuchi. “Photometric stereo under unknown light sources using robust SVD with missing data”. In: *2010 IEEE International Conference on Image Processing*. IEEE. 2010, pp. 4057–4060.

[139] B. Mohar, Y. Alavi, G. Chartrand, and O.R. Oellermann. “The Laplacian spectrum of graphs”. In: *Graph theory, combinatorics, and applications* 2.871-898 (1991), p. 12.

[140] Bojan Mohar. “Eigenvalues, diameter, and mean distance in graphs”. In: *Graphs and combinatorics* 7.1 (1991), pp. 53–64.

[141] O. Morozova, V. Morozov, B. G. Hoffman, C. D. Helgason, and M. A. Marra. “A seriation approach for visualization-driven discovery of co-expression patterns in Serial Analysis of Gene Expression (SAGE) data”. In: *PloS one* 3.9 (2008), e3205.

[142] J. L. Morrison, R. Breitling, D. J. Higham, and D. R. Gilbert. “A lock-and-key model for protein–protein interactions”. In: *Bioinformatics* 22.16 (2006), pp. 2012–2019.

[143] S. K. Nayar, K. Ikeuchi, and T. Kanade. "Surface reflection: physical and geometrical perspectives". In: *IEEE Transactions on Pattern Analysis & Machine Intelligence* 7 (1991), pp. 611–634.

[144] M. Newman. *Networks*. Oxford university press, 2018.

[145] A. Y. Ng, M. I. Jordan, and Y. Weiss. "On spectral clustering: Analysis and an algorithm". In: *Advances in neural information processing systems*. 2002, pp. 849–856.

[146] M. J. O'Brien and R. L. Lyman. *Seriation Stratigraphy and Index Fossils: The Backbone of Archaeological Dating*. Kluwer, 2002.

[147] M. Oswald and G. Reinelt. "The simultaneous consecutive ones problem". In: *Theor. Comput. Sci.* 410.21-23 (2009), pp. 1986–1992.

[148] *Pajek datasets- Notre Dame Self-Organized Networks Database*. 2004. URL: http://vlado.fmf.uni-lj.si/pub/networks/data/ND/NDnets.htm.

[149] G. Palla, I. Derényi, I. Farkas, and T. Vicsek. "Uncovering the overlapping community structure of complex networks in nature and society". In: *nature* 435.7043 (2005), p. 814.

[150] T. Papadhimitri and P. Favaro. "A new perspective on uncalibrated photometric stereo". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2013, pp. 1474–1481.

[151] C. H. Papadimitriou. *Computational complexity*. John Wiley and Sons Ltd., 2003.

[152] W. M. F. Petrie. "Sequences in Prehistoric Remains". In: *J. R. Anthropol. Inst.* 29 (1899), pp. 295–301.

[153] P. Piana Agostinetti and M. Sommacal. "Il problema della seriazione in archeologia". In: *Rivista di Scienze Preistoriche* LV (2005), pp. 29–69.

[154] A. Pothen, H. D. Simon, and K. P. Liou. "Partitioning sparse matrices with eigenvectors of graphs". In: *SIAM journal on matrix analysis and applications* 11.3 (1990), pp. 430–452.

[155] P. Préa and D. Fortin. "An optimal algorithm to recognize Robinsonian dissimilarities". In: *J. Classif.* 31.3 (2014), p. 351.

[156] U. N. Raghavan, R. Albert, and S. Kumara. "Near linear time algorithm to detect community structures in large-scale networks". In: *Physical review E* 76.3 (2007), p. 036106.

[157] M. Redivo-Zaglia and G. Rodriguez. "smt: a Matlab toolbox for structured matrices". In: *Numer. Algorithms* 59.4 (2012). DOI: 10.1007/s11075-011-9527-9, pp. 639–659.

[158] W. S. Robinson. "A method for chronologically ordering archaeological deposits". In: *Am. Antiq.* 16.4 (1951), pp. 293–301.

[159] M. P. Rombach, M. A. Porter, J. H. Fowler, and P. J. Mucha. "Core-periphery structure in networks". In: *SIAM Journal on Applied mathematics* 74.1 (2014), pp. 167–190.

[160] R. Roth. “On the eigenvectors belonging to the minimum eigenvalue of an essentially nonnegative symmetric matrix with bipartite graph”. In: *Linear Algebra and its Applications* 118 (1989), pp. 1–10.

[161] E. Schmidt. “Zur Theorie der linearen und nichtlinearen Integralgleichungen”. In: *Integralgleichungen und Gleichungen mit unendlich vielen Unbekannten*. Springer, 1989, pp. 190–233.

[162] M. Seminaroti. “Combinatorial Algorithms for the Seriation Problem”. PhD thesis. CentER, Tilburg University, 2016.

[163] M. Seston. “Dissimilarités de Robinson: algorithmes de reconnaissance et d’approximation”. PhD thesis. Aix Marseille 2, 2008.

[164] J. Shi and J. Malik. “Normalized cuts and image segmentation”. In: *Departmental Papers (CIS)* (2000), p. 107.

[165] A. Shuchat. “Matrix and network models in archaeology”. In: *Mathematics Magazine* 57.1 (1984), pp. 3–14.

[166] M. Sipser. “The history and status of the P versus NP question”. In: *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*. ACM. 1992, pp. 603–618.

[167] G. Slabaugh, B. Culbertson, T. Malzbender, and R. Schafer. “A survey of methods for volumetric scene reconstruction from photographs”. In: *Proceedings of the 2001 Eurographics Conference on Volume Graphics*. Vienna: Springer, 2001, pp. 81–100.

[168] D. A. Spielman and S. H. Teng. “Spectral partitioning works: Planar graphs and finite element meshes”. In: *Proceedings of 37th Conference on Foundations of Computer Science*. IEEE. 1996, pp. 96–105.

[169] G. W. Stewart. “On the early history of the singular value decomposition”. In: *SIAM review* 35.4 (1993), pp. 551–566.

[170] G. Stocchino. “Modelli Matematici e Algoritmi Numerici per la Photometric Stereo”. Bachelor’s Thesis in Mathematics, University of Cagliari. Available at http://bugs.unica.it/~gppe/did/tesi/15stocchino.pdf. 2015.

[171] J. J. Sylvester. “A new proof that a general quadric may be reduced to its canonical form (that is, a linear function of squares) by means of a real orthogonal substitution”. In: *Messenger of Mathematics* 19 (1889), pp. 1–5.

[172] A. Taylor, J. K. Vass, and D. J. Higham. “Discovering bipartite substructure in directed networks”. In: *LMS Journal of Computation and Mathematics* 14 (2011), pp. 72–86.

[173] G. P. Telles and J. Meidanis. “Building PQR trees in almost-linear time.” In: *Electronic Notes in Discrete Mathematics* 19 (2005), pp. 33–39.

[174] Y. J. Tien, Y. S. Lee, H. M. Wu, and C. H. Chen. “Methods for simultaneously identifying coherent local clusters with smooth global patterns in gene expression profiles”. In: *BMC bioinformatics* 9.1 (2008), p. 155.

[175] C. Tomasi and T. Kanade. “Shape and motion from image streams under orthography: a factorization method”. In: *International journal of computer vision* 9.2 (1992), pp. 137–154.

[176] L. N. Trefethen and D. Bau III. *Numerical linear algebra*. Vol. 50. Siam, 1997.

[177] F. Ulupinar and R. Nevatia. "Inferring shape from contour for curved surfaces". In: *[1990] Proceedings. 10th International Conference on Pattern Recognition*. Vol. 1. IEEE. 1990, pp. 147–154.

[178] J. Van Leeuwen. *Handbook of theoretical computer science*. Vol. 1. Elsevier, 1990.

[179] P. Van Mieghem. *Graph spectra for complex networks*. Cambridge University Press, 2010.

[180] M. Vanzi, C. Mannu, R. Dessì, G. Rodriguez, and G. Tanda. "Photometric stereo for 3D mapping of carvings and relieves: case studies on prehistorical art in Sardinia". In: *XVII Seminário Internacional de Arte Rupestre de Mação*. Vol. Visual Projections N. 3. Mação, Portugal: Ângulo Repositório Didáctico (ISSN 1645-8214), 2014.

[181] G. Vogiatzis and C. Hernández. "Practical 3d reconstruction based on photometric stereo". In: *Computer vision*. Springer, 2010, pp. 313–345.

[182] H. Weyl. "Das asymptotische Verteilungsgesetz der Eigenwerte linearer partieller Differentialgleichungen (mit einer Anwendung auf die Theorie der Hohlraumstrahlung)". In: *Mathematische Annalen* 71.4 (1912), pp. 441–479.

[183] J. Wilcock, J. E. Doran, and F. R. Hodson. "Mathematics and computers in archaeology. Edinburgh: University Press, 1975. 392 pp., 99 figs.£ 8.00." In: *Antiquity* 51.202 (1977), pp. 158–159.

[184] E. M. Wilkinson. "Archaeological seriation and the travelling salesman problem". In: *Mathematics in the archaeological and historical sciences* (1971), pp. 276–283.

[185] R. J. Woodham. "Photometric method for determining surface orientation from multiple images". In: *Opt. Eng.* 19.1 (1980), pp. 191139–191139.

[186] F. Xu and K. Mueller. "Real-time 3D computed tomographic reconstruction using commodity graphics hardware". In: *Physics in Medicine & Biology* 52.12 (2007), p. 3405.

[187] B. Yang, W. Cheung, and J. Liu. "Community mining from signed social networks". In: *IEEE transactions on knowledge and data engineering* 19.10 (2007), pp. 1333–1348.

[188] M. Yannakakis. "Edge-deletion problems". In: *SIAM Journal on Computing* 10.2 (1981), pp. 297–309.

[189] A. Yuille and D. Snow. "Shape and albedo from multiple images using integrability". In: *cvpr*. Vol. 97. Citeseer. 1997, p. 158.

[190] R. Zhang, P.-S. Tsai, J. E. Cryer, and M. Shah. "Shape-from-shading: a survey". In: *IEEE transactions on pattern analysis and machine intelligence* 21.8 (1999), pp. 690–706.