Università degli Studi dell'Insubria
Dipartimento di Scienza e Alta Tecnologia

Edge manipulation techniques in complex networks with applications to communicability and triadic closure

Author:
Francesca Arrigo

Supervisor:
Prof. Michele Benzi

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

in

Computational Mathematics

February 11, 2016

Declaration of Authorship

I, Francesca Arrigo, declare that this thesis titled, "Edge manipulation techniques in complex networks with applications to communicability and triadic closure" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

"It always seems impossible until it's done."

Nelson Mandela

UNIVERSITÀ DEGLI STUDI DELL'INSUBRIA

Abstract

Mathematics
Dipartimento di Scienza e Alta Tecnologia

Doctor of Philosophy

Edge manipulation techniques in complex networks with applications to communicability and triadic closure

by Francesca Arrigo

Complex networks are ubiquitous in our everyday life and can be used to model a wide variety of phenomena. For this reason, they have captured the interest of researchers from a wide variety of fields. In this work, we describe how to tackle two problems that have their focus on the edges of networks.

Our first goal is to develop mathematically inferred, efficient methods based on some newly introduced edge centrality measures for the manipulation of links in a network. We want to make a small number of changes to the edges in order to tune its overall ability to exchange information according to certain goals. Specifically, we consider the problem of adding a few links in order to increase as much as possible this ability and that of selecting a given number of connections to be removed from the graph in order to penalize it as little as possible. Techniques to tackle these problems are developed for both undirected and directed networks. Concerning the directed case, we further discuss how to approximate certain quantities that are used to measure the importance of edges.

Secondly, we consider the problem of understanding the mechanism underlying triadic closure in networks and we describe how communicability distance functions play a role in this process.

Extensive numerical tests are presented to validate our approaches.

Acknowledgements

I want to thank my Advisor, Prof. Michele Benzi, for everything he taught me over these years, for the patience, the support, and for the countless helpful discussions.

I am indebted to Prof. Marco Donatelli, my Tutor, for introducing me to Michele and for being constantly there when I needed help.

I am grateful to Prof. Ernesto Estrada for having taught me to be dedicated and (sometimes) stubborn when doing research. I want to thank Dr. Caterina Fenu for the helpful discussions and fruitful collaboration.

I would like to thank the Department of Mathematics and Computer Science at Emory University for the hospitality offered for more than 15 months over the years 2013, 2014, and 2015.

I also thank two anonymous referees for the careful reading of this Thesis and for the helpful suggestions they provided.

For this work I have been supported by the University of Insubria, Emory University, Gruppo Nazionale di Calcolo Scienifico, and the Society for Industrial and Applied Mathematics.

Contents

List of Figures

List of Tables

Notation

$\mathcal{G}$	graph
$\mathcal{V}$	set of nodes
$\mathcal{E}$	set of edges
n	number of nodes
m	number of edges
A	adjacency matrix of a (di)graph
$\mathscr{A}$	adjacency matrix of a bipartite graph
I	identity matrix (of appropriate size)
$\mathbf{e}_i$	ith column of I (of appropriate size)
$\mathbf{1}$	vector of all ones (of appropriate size)
K	budget of modifications

Introduction

Networks are nowadays ubiquitous in our daily life and we are ourselves part of several networks, such as those describing social interactions.

Complex networks can be described as objects living in the Euclidean space (e.g., power grids, road networks, neural networks, etc.) or as entities living in an abstract space (such as collaboration networks, social interaction networks, etc.). Mathematically, they can be modeled as graphs, i.e., as pairs of sets, one containing the entities (nodes) and the other containing the interactions among them (edges). These graphs are highly sparse and exhibit non-trivial topological features, since they are usually far from being regular but they are not completely random either.

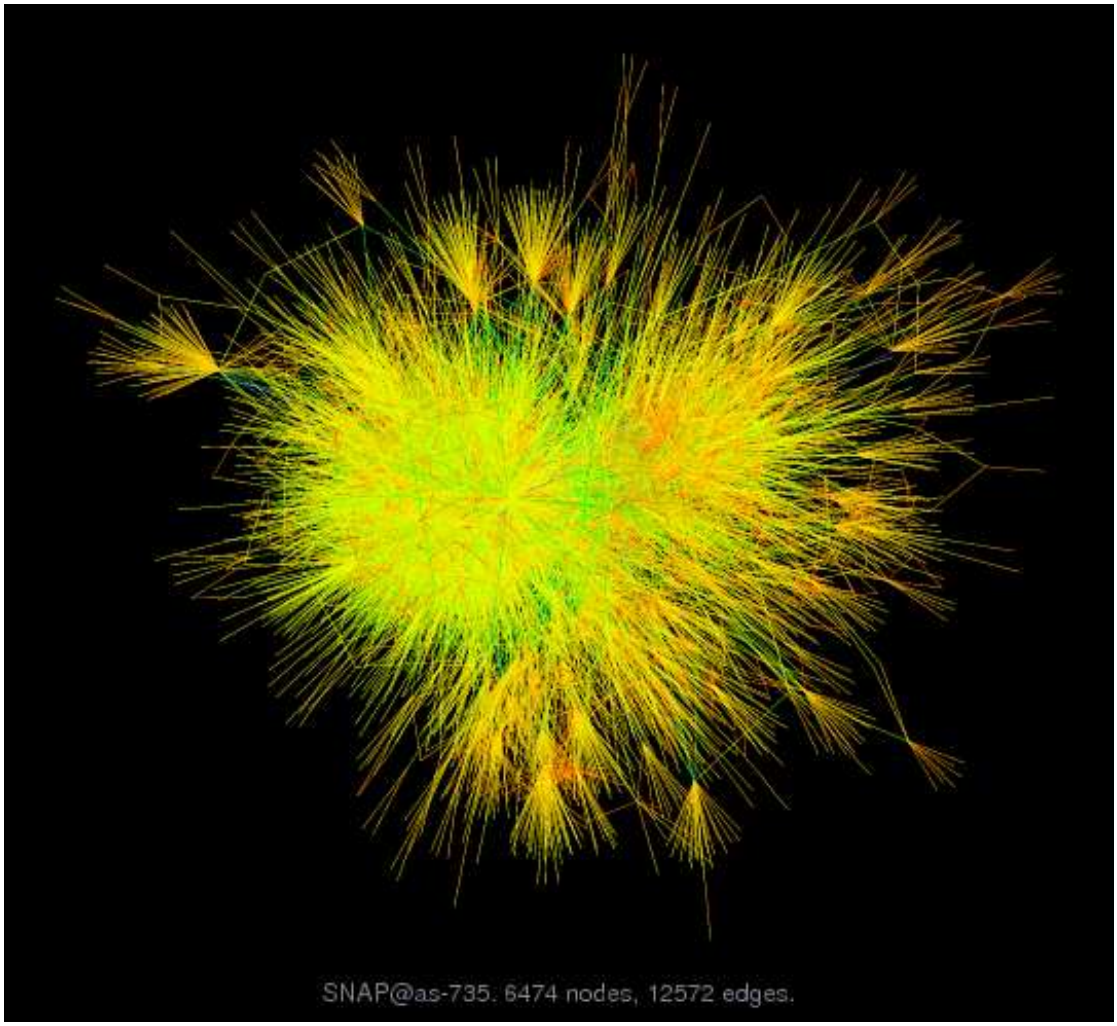
Due to the broad range of possible applications and to the large amount of data available, the field of network science has become increasingly appealing for experts working in various fields, from sociology to physics and mathematics, attracting more and more researchers every year.

Over the years, one of the main objects of study has been the identification of which are the "most important" nodes in a network. To quantify the importance of nodes, several centrality measures have been introduced [13, 18, 42, 46, 59]. On the other hand, not nearly as much work has been done in identifying the most important interactions taking place in a network. Generally speaking, researchers have not put much effort in the study of the properties of edges.

In this Thesis we mainly address two problems that have their focus on the edges of a graph rather than on its nodes.

After reviewing some basic concepts from graph theory, linear algebra, and network science in Chapter 1, we address these two problems in Chapters 2, 3, and 5.

FIGURE 1: Representation of the network as-735. A description of this network can be found in Appendix A. Figure from [26].



The first problem, addressed in Chapters 2 and 3, concerns how to modify the edges in an existing network in order to tune its ability to diffuse information (in the broad sense of the term) among its nodes. More specifically, we describe how to add, remove, or redirect a limited number of connections in the graph in order to increase as much as possible its communication capability when adding or redirecting links, or to penalize this capability as little as possible, when removing connections. To this end, we introduce new edge centrality measures that can be used to guide in the selection of the edges to be added or removed. The developed techniques produce networks that are highly sparse, but yet are very well connected and thus less likely to be disrupted by either random failures or targeted attacks leading to the loss of edges. When describing our heuristics in the case of directed networks (see Chapter 3), we have to face some computational issues that are discussed in depth in Chapter 4. The second problem we address, discussed in

Chapter 5, deals with the prediction of the appearance of edges that close triangles in networks. Indeed, the problem of understanding when two common friends of someone will become friends is of great interest in the field of network science and it is far from being trivial. We propose a communication-driven mechanism for predicting triadic closure in complex networks mathematically formulated on the basis of communicability distance functions that account for the quality of communication between nodes in the network.

Finally, the last chapter is devoted to the conclusive remarks.

Chapter 1

Background

In this chapter we provide some basic definitions, notation, properties, and terminology from graph theory, linear algebra, and network science that will be used throughout this thesis.

1.1 Fundamentals in graph theory

In this section we introduce some definitions and notation associated with graphs.

A *graph* or *network* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is defined by a set of n *nodes* (*vertices*) $\mathcal{V} = \{1, 2, \ldots, n\}$ and a set of m *edges* $\mathcal{E} = \{(i, j) : \ i, j \in \mathcal{V}\}$ between the nodes. The elements in the complement of $\mathcal{E}$, denoted by $\overline{\mathcal{E}}$, will be referred to as *virtual edges*.

Every graph can be represented as a matrix $A = (a_{ij}) \in \mathbb{R}^{n \times n}$, called the *adjacency matrix* of the graph, whose entries are

$$a_{ij} = \begin{cases} \omega_{ij} & \text{if } (i, j) \in \mathcal{E} \\ 0 & \text{otherwise} \end{cases} \qquad \forall i, j \in \mathcal{V},$$

where each $\omega_{ij} \in \mathbb{R}_{>0}$ is the weight of the corresponding edge $(i, j) \in \mathcal{E}$.

A graph is said to be *unweighted* if all its edges have the same weight; without loss of generality, we can then assume that all the weights are unitary and that the associated adjacency matrix is binary.

Unless otherwise specified, all the graphs considered in the remaining of this thesis are assumed to be unweighted.

We define the *downdating* of the edge $(i,j) \in \mathcal{E}$ as the removal of this edge from the network. The resulting graph $\widehat{\mathcal{G}} = (\mathcal{V}, \widehat{\mathcal{E}})$ may be disconnected. Similarly, let $(i,j) \in \overline{\mathcal{E}}$ be a virtual edge for the graph $\mathcal{G}$. We can construct a new graph $\widetilde{\mathcal{G}} = (\mathcal{V}, \widetilde{\mathcal{E}})$ obtained from $\mathcal{G}$ by adding the virtual edge (i,j) to the graph. This procedure will be referred to as the *updating* of the virtual edge (i,j). Finally, the operation of downdating an edge and successively updating a virtual edge will be referred to as *rewiring*.

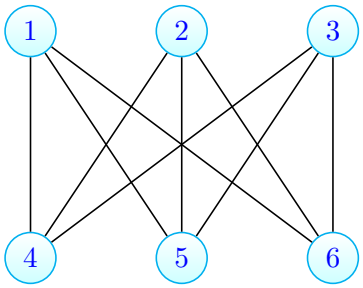
1.1.1 Undirected graphs

A graph is said to be *undirected* if the edges are formed by unordered pairs of vertices, i.e., if $(i,j) \in \mathcal{E} \Leftrightarrow (j,i) \in \mathcal{E}$. If a graph is undirected, then the associated adjacency matrix is symmetric, and conversely.

An edge is said to be *incident* to a vertex i if there exists a node $j \neq i$ such that $(i,j) \in \mathcal{E}$. In an undirected graph, the *degree* of a vertex i, denoted by d_i, is the number of edges incident to i. A *walk* of length k is a sequence of $k+1$ nodes $i_1, i_2, \ldots, i_{k+1}$ such that for all $1 \leq \ell \leq k$ it holds that $(i_\ell, i_{\ell+1}) \in \mathcal{E}$. A *closed walk of length k centered at node i* is a walk of length k in which the first and the last nodes of the sequence are equal to i. It is well known in graph theory that the powers of the adjacency matrix can be used to count walks in the associated graph (see [7, Chapter 3]). More specifically, the diagonal entries of the kth power of A count the number of closed walks of length k centered at each node, while the off-diagonal entries $(A^k)_{ij}$ for $i,j \in \mathcal{V}$ count the number of walks of length k starting at node i and ending at node j. A closed walk of length 3 is called a *triangle*. We will call *triad* every triplet of nodes i, j, and h such that $(i,j),(j,h) \in \mathcal{E}$ but $(i,h) \notin \mathcal{E}$. Hence a triad is a triangle missing one edge. We shall call this virtual edge a *potential edge*.

If all the nodes in the sequence describing a walk are different, than the walk is called a *path*. An undirected graph is *connected* if there is a path connecting every pair of nodes. A graph with unweighted edges, no self-loops (edges from a node to itself), and no multiple edges is said to be *simple*. If the network is simple, then the corresponding adjacency matrix is binary with zeros on the main diagonal.

FIGURE 1.1: Example of 3-regular graph with 6 nodes.



In the following, we briefly recall the definitions of four types of graph.

Definition 1.1. The *complete graph* with n nodes, denoted by K_n, is the simple, undirected graph such that there is an edge from each node to any other node in the network.

Definition 1.2. A *path graph* or *linear graph* with n nodes, denoted by P_n, is a simple, undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ such that

$$\mathcal{E} = \{(i, i+1) | i = 1, 2, \ldots, n-1\}.$$

Definition 1.3. The *empty graph* with n nodes is the graph $\mathcal{G} = (\mathcal{V}, \varnothing)$ which contains n nodes and no edges among them.

Definition 1.4. A *regular graph of degree* k or a *k-regular graph* is a graph in which all the nodes have the same degree, equal to k.

As an example of regular graph, consider the 3-regular graph with six nodes in Figure 1.1

1.1.2 Directed graphs

When the edges of the network are formed by ordered pairs of vertices, the network is said to be *directed* or, equivalently, it is called a *digraph*. In this case, for each edge $(i, j) \in \mathcal{E}$ we call i the *source node* and j the *target node*. If $(i, j) \in \mathcal{E}$, we will write $i \longrightarrow j$. An edge $(i, j) \in \mathcal{E}$ is said to be *unidirectional* if $(j, i) \notin \mathcal{E}$. Clearly, the adjacency matrix associated to a digraph is not symmetric.

Every node $i \in \mathcal{V}$ in a digraph has two types of degree, namely the *in-degree* and the *out-degree*; the first, denoted by $d_{in}(i)$, counts the number of edges of the form $* \longrightarrow i$,

i.e., the number of vertices in $\mathcal{G}$ from which it is possible to reach i in one step. The *out-degree*, on the other hand, counts the number of nodes that can be reached from i in one step, i.e., the number of edges of the form $i \longrightarrow *$, and it is denoted by $d_{out}(i)$.

The notions of walk and path for an undirected network naturally extend to the case of digraphs, as described in the following. A *walk* of length k is a sequence of $k+1$ nodes $\{i_1, i_2, \ldots, i_{k+1}\}$ such that

$$i_1 \longrightarrow i_2 \longrightarrow \cdots \longrightarrow i_{k+1},$$

i.e., such that for all $1 \leq \ell \leq k$ it holds that $(i_\ell, i_{\ell+1}) \in \mathcal{E}$. A *closed walk of length k centered at node i* is a walk of length k in which $i_1 = i_{k+1}$. As in the undirected case, the powers of the adjacency matrix can be used to count walks in the associated graph: the diagonal entries of the kth power of A count the number of closed walks of length k centered at each node, whereas the off-diagonal entries $(A^k)_{ij}$ for $i, j \in \mathcal{V}$ count the number of walks of length k starting at node i and ending at node j. A *path of length k* is a walk of length k with no repeated nodes.

Following [11, 25], we can define two other types of walks.

Definition 1.5. An *alternating walk of length k starting with an out-edge* is a list of nodes $i_1, i_2, \ldots, i_{k+1}$ such that there exists an edge $(i_\ell, i_{\ell+1})$ if ℓ is odd and an edge $(i_{\ell+1}, i_\ell)$ if ℓ is even. An alternating walk starting with an out-edge hence has the form

$$i_1 \longrightarrow i_2 \longleftarrow i_3 \longrightarrow \cdots .$$

Definition 1.6. An *alternating walk of length k starting with an in-edge* is a list of vertices $i_1, i_2, \ldots, i_{k+1}$ such that

$$i_1 \longleftarrow i_2 \longrightarrow i_3 \longleftarrow \cdots ,$$

i.e., such that there exists an edge $(i_\ell, i_{\ell+1})$ if ℓ is even and an edge $(i_{\ell+1}, i_\ell)$ otherwise.

The alternating walks of any length connecting two nodes can be counted by using the powers of two matrices AA^T and A^TA, called the *hub* and *authority matrices*, respectively. The reason for this terminology will become clear later. The entries of the kth power of the hub matrix $[(AA^T)^k]_{ij}$ count the number of alternating walks of length $2k$

starting with an out-edge from node i to node j. Likewise, the entries $[(A^TA)^k]_{ij}$ of the kth power of the authority matrix count the number of alternating walks from node i to node j of length $2k$ starting with an in-edge [11].

A digraph is said to be *strongly connected* if every two nodes in the network are connected through a path of finite length, while it is said to be *weakly connected* if this property holds when the directionality of the links is disregarded.

1.1.3 Bipartite graphs

An undirected network $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ is called *bipartite* if it is possible to partition its set of nodes as $\mathscr{V} = \mathcal{V}_1 \cup \mathcal{V}_2$ in such a way that for all $(i, j) \in \mathscr{E}$ it either holds that $i \in \mathcal{V}_1$ and $j \in \mathcal{V}_2$ or $i \in \mathcal{V}_2$ and $j \in \mathcal{V}_1$. Let us call n_1 the cardinality of $\mathcal{V}_1$ and $n_2 = n - n_1$ the cardinality of $\mathcal{V}_2$. It is always possible to label the elements of $\mathscr{V}$ in such a way that $\mathcal{V}_1 = \{1, 2, \ldots, n_1\}$ and $\mathcal{V}_2 = \{n_1 + 1, n_1 + 2, \ldots, n_1 + n_2 = n\}$ and we will assume this convention in the following. The adjacency matrix associated with a bipartite graph has the form:

$$\mathscr{A} = \begin{pmatrix} 0 & B \\ B^T & 0 \end{pmatrix}, \qquad B \in \mathbb{R}^{n_1 \times n_2}.$$

There is a closed general expression for the powers of the adjacency matrix of a bipartite graph:

$$\mathscr{A}^{2k} = \begin{pmatrix} (BB^T)^k & 0 \\ 0 & (B^TB)^k \end{pmatrix}, \qquad \mathscr{A}^{2k+1} = \begin{pmatrix} 0 & B(B^TB)^k \\ (B^TB)^kB^T & 0 \end{pmatrix}.$$

These expressions tell us that there exist no walks of even length between nodes that belong to two different sets, while there are no walks of odd length between nodes that belong to the same set.

Digraphs as bipartite graphs

There is a one-to-one correspondence between bipartite graphs with $n_1 = n_2$ and digraphs (see [5, 11, 19, 30, 31]). Indeed, let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a directed graph and consider the graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$, whose sets of nodes and edges are constructed as follows. The set $\mathscr{V}$ contains $2n$ nodes and is formed as $\mathscr{V} = \mathcal{V} \cup \mathcal{V}'$, where $\mathcal{V}$ is the original set of

nodes and $\mathcal{V}' = \{1' = n+1, 2' = n+2, \ldots, n' = 2n\}$ is a set of copies of the nodes in $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The edges between the elements in $\mathscr{V}$ are undirected and $(i, j') \in \mathscr{E}$ with $i \in \mathcal{V}$ and $j' \in \mathcal{V}'$ if and only if $(i, j) \in \mathcal{E}$ in the original digraph. This graph is bipartite by construction and its adjacency matrix has thus the form

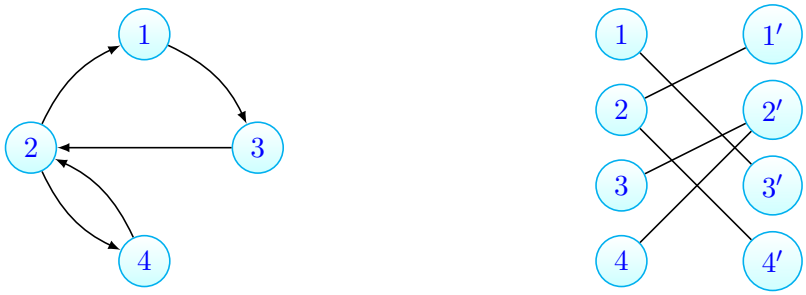
$$\mathscr{A} = \begin{pmatrix} 0 & A \\ A^T & 0 \end{pmatrix}, \tag{1.1}$$

where A is the adjacency matrix of the original digraph.

As an example of this construction, consider the matrix

$$A = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}. \tag{1.2}$$

The associated digraph and the corresponding bipartite graph are then:



Note that in the bipartite graph associated to a digraph the first n nodes, contained in $\mathcal{V}$, can be seen as the original nodes of the digraph when they play their role of sources of information; similarly, the n copies, contained in $\mathcal{V}'$, represent the original nodes in their role of targets.

1.2 Fundamentals in linear algebra

In this section we recall some fundamental notions and results from linear algebra.

Let $A \in \mathbb{R}^{n \times n}$ and assume that $A^T = A$; then the eigenvalues of A are real and we can label them in non-increasing order as: $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$. Furthermore, we can

decompose A into

$$A = Q\Lambda Q^T = \sum_{j=1}^{n} \lambda_j \mathbf{q}_j \mathbf{q}_j^T,$$

where $\Lambda = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$ is a diagonal matrix containing the eigenvalues of A and $Q = [\mathbf{q}_1, \ldots, \mathbf{q}_n] \in \mathbb{R}^{n\times n}$ is orthogonal, where $\mathbf{q}_i$ is a normalized eigenvector associated to the eigenvalue λ_i, for all $i = 1, 2, \ldots, n$. If the matrix A is not symmetric, then it may have complex eigenvalues. In this case we will assume that the eigenvalues are ordered with non-increasing modulus $|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_n|$.

The *spectral radius* of A is defined as $\rho(A) = \max_{i=1,2,\ldots,n} \{|\lambda_i|\}$ while, under the hypothesis that $|\lambda_1| \geq |\lambda_i|$ for all $i \neq 1$, the *spectral gap* of A is defined as $\min_{i\neq 1} |\lambda_1 - \lambda_i|$.

Definition 1.7. A matrix $A \in \mathbb{C}^{n\times n}$ is said to be *reducible* if there exists an $n \times n$ permutation matrix P such that

$$P^T A P = \begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{pmatrix},$$

with square diagonal blocks, *irreducible* otherwise.

Definition 1.8. A matrix $A \in \mathbb{R}^{m\times n}$ is said to be *non-negative* (resp., *positive*) if $a_{ij} \geq 0$ (resp., $a_{ij} > 0$) for all $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$. We will write $A \geq 0$ (resp., $A > 0$).

Moreover, let $B \in \mathbb{R}^{m\times n}$. We will write $A \geq B$ if $A - B \geq 0$.

Theorem 1.9. [94, Theorem 1.18] *Let $A \in \mathbb{R}^{n\times n}$. Then A is irreducible if and only if the graph $\mathcal{G}(A)$ that has A as its adjacency matrix is strongly connected.*

From this result, it follows that the adjacency matrix of any connected and undirected network is irreducible.

Theorem 1.10. [57, Theorem 8.4.4] *Let $A \in \mathbb{R}^{n\times n}$ and suppose that $A \geq 0$ is irreducible. Then*

(i) $\rho(A) > 0$*;*

(ii) $\rho(A)$ *is a simple eigenvalue of* A*;*

(iii) there is a positive vector $\mathbf{x}$ *such that* $A\mathbf{x} = \rho(A)\mathbf{x}$*.*

This theorem is known in the literature as the *Perron–Frobenius* Theorem.

From Theorems 1.9 and 1.10 it follows that if the graph is undirected and connected, then $\rho(A) = \lambda_1 \in \mathbb{R}_+$ and $\lambda_1 > \lambda_2$. Moreover, the eigenvector $\mathbf{q_1}$ associated with the leading eigenvalue, sometimes referred to as the *Perron vector*, can be chosen so that its components $q_1(i)$ are positive for all $i \in \mathcal{V}$.

We can further characterize the spectrum of a non-negative irreducible matrix A if we assume it to be p-cyclic.

Definition 1.11. [93, Definition 1] The matrix $A = (a_{ij}) \in \mathbb{R}^{n\times n}$ is *cyclic of index p* or *p-cyclic* ($p \geq 2$) if and only if, after some symmetric permutation of rows and columns of A, the matrix A assumes the cyclic block form

$$\begin{pmatrix} 0 & 0 & \cdots & 0 & L_1 \\ L_2 & 0 & \cdots & 0 & 0 \\ 0 & L_3 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & L_p & 0 \end{pmatrix}$$

where the zero diagonal blocks are square.

It can be proved (see [93] and references therein) that the characteristic polynomial of a p-cyclic matrix A, denoted by $p_A(\lambda) := \det(A - \lambda I)$, has the form

$$p_A(\lambda) = \lambda^\nu \prod_{i=1}^{\ell} (\lambda^p - \sigma_i^p),$$

where $\nu + \ell p = n$. Using this result, it follows that if a non-negative matrix A is irreducible and p-cyclic, then the matrix A has exactly p eigenvalues of modulus $\rho(A)$, which coincide with the roots of the polynomial $q(\lambda) := \lambda^p - \rho(A)^p$ (see [57, Corollary 8.4.6]).

Corollary 1.12. [57, Theorem 8.4.5] *Let $A \in \mathbb{R}^{n\times n}$ be non-negative and irreducible and let $B \in \mathbb{R}^{n\times n}$. Suppose that $A \geq |B|$. Then $\rho(A) \geq \rho(B)$.*

Theorem 1.13. [57, Theorem 4.3.1] *Let $A, B \in \mathbb{R}^{n\times n}$ be symmetric and let the eigenvalues $\lambda_i(A)$, $\lambda_i(B)$, and $\lambda_i(A+B)$ be arranged in non-increasing order. Then, for each*

$i = 1, 2, \ldots, n$ we have

$$\lambda_i(A) + \lambda_n(B) \leq \lambda_i(A + B) \leq \lambda_i(A) + \lambda_1(B).$$

Suppose now that $A \in \mathbb{C}^{m\times n}$ is a complex matrix of rank r. Then we can factor it as $A = U\Sigma V^*$ using a *singular value decomposition* (*SVD*). The matrix $\Sigma \in \mathbb{R}^{m\times n}$ is diagonal and its entries $\Sigma_{ii} = \sigma_i$ are ordered as $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > \sigma_{r+1} = \cdots = \sigma_q = 0$, where $q = \min\{m, n\}$. The matrices $U = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m] \in \mathbb{C}^{m\times m}$ and $V = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n] \in \mathbb{C}^{n\times n}$ are unitary and contain the *left* and *right singular vectors* of A, respectively. It is well known that the matrix Σ is uniquely determined, while U and V are not. If A is real, then also U and V can be chosen to be real.

From the singular value decomposition of A it follows that $AA^* = U\Sigma\Sigma^T U^*$ and $A^*A = V\Sigma^T\Sigma V^*$. Thus, the singular values of a matrix A are the square roots of the positive eigenvalues of the matrix AA^* or A^*A. Moreover, the left singular vectors of A are the eigenvectors of the matrix AA^*, while the right singular vectors are the eigenvectors of the matrix A^*A.

The singular values of a matrix also emerge (with their opposites) as the eigenvalues of the matrix

$$\mathscr{A} = \begin{pmatrix} 0 & A \\ A^* & 0 \end{pmatrix}.$$

This can be easily seen in the case when $m = n$, where $\mathscr{A}$ can be decomposed as

$$\mathscr{A} = \frac{1}{2}\begin{pmatrix} U & -U \\ V & V \end{pmatrix}\begin{pmatrix} \Sigma & 0 \\ 0 & -\Sigma \end{pmatrix}\begin{pmatrix} U & -U \\ V & V \end{pmatrix}^*.$$

Consider now the matrices $U_r \in \mathbb{C}^{m\times r}$ and $V_r \in \mathbb{C}^{n\times r}$, which contain the first r columns of the matrices U and V, respectively, and let $\Sigma_r \in \mathbb{R}^{r\times r}$ be the $r \times r$ leading block of Σ. Then a *compact SVD* (*CSVD*) of the matrix $A \in \mathbb{C}^{m\times n}$ is

$$A = U_r\Sigma_r V_r^* = \sum_{j=1}^{r} \sigma_j \mathbf{u}_j \mathbf{v}_j^*.$$

1.2.1 Matrix functions

There are several equivalent ways to define $f(A)$ when $A \in \mathbb{C}^{n\times n}$ is a square matrix. We recall here the definition based on the Jordan canonical form. For a more comprehensive study, we refer to [56].

Let $\{\lambda_1, \lambda_2, \ldots, \lambda_s\}$ be the set of distinct eigenvalues of A and let n_i denote the *index* of the ith eigenvalue, i.e., the size of the largest Jordan block associated with λ_i.

Recall that a function f is said to be *defined on the spectrum of* A if the values $f^{(\ell)}(\lambda_i)$ exist for all $\ell = 0, 1, \ldots, n_i - 1$ and for all $i = 1, 2, \ldots, s$, where $f^{(\ell)}$ is the ℓth derivative of the function and $f^{(0)} \equiv f$.

Definition 1.14. [56, Definition 1.2] Let f be defined on the spectrum of $A \in \mathbb{C}^{n\times n}$ and let $Z^{-1}AZ = J = \mathrm{diag}(J_1, J_2, \ldots, J_p)$ be the Jordan canonical form of the matrix, where

$$J_i = J_i(\lambda_i) = \begin{pmatrix} \lambda_i & 1 & & \\ & \lambda_i & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_i \end{pmatrix} \in \mathbb{C}^{m_i\times m_i},$$

$\sum_{j=1}^{p} m_j = n$, and Z is nonsingular. Then

$$f(A) := Zf(J)Z^{-1} = Z\mathrm{diag}(f(J_1), f(J_2), \ldots, f(J_p))Z^{-1}$$

where

$$f(J_i) := \begin{pmatrix} f(\lambda_i) & f'(\lambda_i) & \cdots & \frac{f^{(m_i-1)}(\lambda_i)}{(m_i-1)!} \\ & f(\lambda_i) & \ddots & \vdots \\ & & \ddots & f'(\lambda_i) \\ & & & f(\lambda_i) \end{pmatrix}.$$

If the matrix A is diagonalizable, then the Jordan canonical form reduces to the spectral decomposition $A = Z\Lambda Z^{-1}$, with $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$ and the columns of $Z \in \mathbb{C}^{n\times n}$ eigenvectors of A. In such case,

$$f(A) = Zf(\Lambda)Z^{-1} = Z\mathrm{diag}(f(\lambda_1), f(\lambda_2), \ldots, f(\lambda_n))Z^{-1}.$$

If the function f has a Taylor series expansion, we can use it to describe the associated matrix function, provided that the eigenvalues of the matrix A satisfy certain requirements.

Theorem 1.15. [56, Theorem 4.7] *Let $A \in \mathbb{C}^{n\times n}$ and suppose that f can be expressed as*

$$f(z) = \sum_{k=0}^{\infty} a_k (z - z_0)^k$$

with radius of convergence R. Then $f(A)$ is defined and is given by

$$f(A) = \sum_{k=0}^{\infty} a_k (A - z_0 I)^k$$

if and only if each of the distinct eigenvalues of A $\{\lambda_1, \lambda_2, \ldots, \lambda_s\}$ satisfies one of the following:

(i) $|\lambda_i - z_0| < R$ *for all* $i = 1, 2, \ldots, s$;

(ii) $|\lambda_i - z_0| = R$ *and the series for* $f^{(n_i-1)}(\lambda)$ *is convergent at the point* $\lambda = \lambda_i$, $i = 1, 2, \ldots, s$, *where* n_i *is the index of* λ_i.

1.2.2 Generalized matrix functions

In [55] the authors considered the problem of extending the definition of matrix function to the case of rectangular matrices. Their idea was to use in the definition a generalization of the SVD of the matrix instead of its Jordan canonical form.

Theorem 1.16. *Let $A \in \mathbb{C}^{m\times n}$ be a matrix of rank r and let $\{c_i : i = 1, 2, \ldots, r\}$ be any set of complex numbers satisfying*

$$|c_i|^2 = \sigma_i^2 = \lambda_i(AA^*),$$

where $\lambda_1(AA^*) \geq \lambda_2(AA^*) \geq \cdots \geq \lambda_r(AA^*) > 0$ are the positive eigenvalues of AA^*. Then there exist two unitary matrices $\mathcal{X} \in \mathbb{C}^{m\times m}$ and $\mathcal{Y} \in \mathbb{C}^{n\times n}$ such that $\mathcal{D} = \mathcal{X}^* A \mathcal{Y} \in \mathbb{C}^{m\times n}$ has entries:

$$d_{ij} = \begin{cases} c_i & \text{if } 1 \leq i = j \leq r \\ 0 & \text{otherwise} \end{cases}.$$

From this theorem it follows that, once the non-zero entries of $\mathcal{D}$ are fixed, A can be written as

$$A = \mathcal{X}\mathcal{D}\mathcal{Y}^* = \mathcal{X}_r\mathcal{D}_r\mathcal{Y}_r^*, \tag{1.3}$$

where $\mathcal{D}_r$ is the leading $r \times r$ block of $\mathcal{D}$ and the matrices $\mathcal{X}_r \in \mathbb{C}^{m\times r}$ and $\mathcal{Y}_r \in \mathbb{C}^{n\times r}$ consist of the first r columns of the matrices $\mathcal{X}$ and $\mathcal{Y}$, respectively.

In the following, we will assume that $c_i = \sigma_i$ for all $i = 1, 2, \ldots, r$; this assumption ensures that the decompositions in (1.3) coincide with the SVD and CSVD of the matrix A, respectively. In particular, $\mathcal{D} = \Sigma$, $\mathcal{X} = U$, and $\mathcal{Y} = V$.

Definition 1.17. Let $A \in \mathbb{C}^{m\times n}$ be a rank-r matrix and let $A = U_r\Sigma_r V_r^*$ be its CSVD. Let $f : \mathbb{R} \longrightarrow \mathbb{R}$ be a scalar function such that $f(\sigma_i)$ is well defined for all $i = 1, 2, \ldots, r$. The generalized matrix function $f^\diamond : \mathbb{C}^{m\times n} \longrightarrow \mathbb{C}^{m\times n}$ induced by f is defined as

$$f^\diamond(A) := U_r f(\Sigma_r) V_r^*, \tag{1.4}$$

where $f(\Sigma_r)$ is defined for the square matrix Σ_r according to definition 1.14 as

$$f(\Sigma_r) = \operatorname{diag}(f(\sigma_1), f(\sigma_2), \ldots, f(\sigma_r)).$$

Generalized matrix functions arise naturally when one is computing $f(\mathscr{A})$, where $\mathscr{A}$ is the structured matrix presented earlier in (1.1). Indeed, if one uses the description of matrix function in terms of power series $f(z) = \sum_{k=0}^{\infty} a_k z^k$, it can be proved that, within the radius of convergence:

$$f(\mathscr{A}) = \begin{pmatrix} f_{even}(\sqrt{AA^*}) & f^\diamond_{odd}(A) \\ f^\diamond_{odd}(A^*) & f_{even}(\sqrt{A^*A}) \end{pmatrix}, \tag{1.5}$$

where

$$f(z) = f_{even}(z) + f_{odd}(z) = \sum_{k=0}^{\infty} a_{2k} z^{2k} + \sum_{k=0}^{\infty} a_{2k+1} z^{2k+1}.$$

Indeed, from [56, Chapter 4] it is known that

$$f(\mathscr{A}) = \begin{pmatrix} \sum_{k=0}^{\infty} a_{2k}(AA^*)^k & A\sum_{k=0}^{\infty} a_{2k+1}(A^*A)^k \\ A^*\sum_{k=0}^{\infty} a_{2k+1}(AA^*)^k & \sum_{k=0}^{\infty} a_{2k}(A^*A)^k \end{pmatrix}.$$

By using the facts that $A = U_r \Sigma_r V_r^*$, $A^*A = V_r \Sigma_r^2 V_r^*$, and $AA^* = U_r \Sigma_r^2 U_r^*$ it follows that

$$f(\mathscr{A}) = \begin{pmatrix} f_{even}(\sqrt{AA^*}) & U_r f_{odd}(\Sigma_r) V_r^* \\ V_r f_{odd}(\Sigma_r) U_r^* & f_{even}(\sqrt{A^*A}) \end{pmatrix},$$

and thus (1.5) follows.

Remark 1.18. If $f(0) = 0$ and the matrix $A \in \mathbb{C}^{n\times n}$ is Hermitian and positive semidefinite, then the generalized matrix function $f^{\diamond}(A)$ reduces to the classical definition of matrix function $f(A)$ (see [55, 56]).

Remark 1.19. The Moore–Penrose pseudo-inverse of a matrix [57, p. 421] $A \in \mathbb{C}^{m\times n}$, denoted by $A^{\dagger}$, can be expressed as $f^{\diamond}(A^*)$, where $f(z) = z^{-1}$.

1.3 Bounds on bilinear forms via quadrature rules

In this section we review the technique exposed in [49] to approximate bilinear forms.

Let $A \in \mathbb{R}^{n\times n}$ be symmetric and let $\mathbf{z}, \mathbf{w} \in \mathbb{R}^n$ be two vectors. Let $f : \mathbb{R} \longrightarrow \mathbb{R}$ be a function defined on the spectrum of A.

Bounds on bilinear forms $\mathbf{z}^T f(A)\mathbf{w}$ can be derived based on Gauss–type quadrature rules when f is a *completely monotonic* (c.m.) function on the interval $[\alpha, \beta]$ containing the spectrum of A by working on a matrix derived from p steps of the symmetric Lanczos iteration (see [12, 49]).

Definition 1.20. A function $f : \mathbb{R} \longrightarrow \mathbb{R}$ is said to be *completely monotonic (c.m.)* on $[\alpha, \beta]$ if it is continuous on $[\alpha, \beta]$ and infinitely differentiable on (α, β) with

$$(-1)^{\ell} f^{(\ell)}(t) \geq 0, \qquad t \in (\alpha, \beta), \forall \ell = 0, 1, \ldots,$$

where $f^{(\ell)}$ denotes the ℓth derivative of f and $f^{(0)} \equiv f$.

The starting point is to observe that bilinear forms $\mathbf{z}^T f(A)\mathbf{w}$ can be thought of as Riemann–Stieltjes integrals with respect to the spectral measure associated with the

symmetric matrix A and with the vectors $\mathbf{z}$ and $\mathbf{w}$:

$$\mathbf{z}^T f(A)\mathbf{w} = \int_\alpha^\beta f(\lambda)\,\mathrm{d}m(\lambda), \quad m(\lambda) = \begin{cases} 0, & \text{if } \lambda < \alpha = \lambda_n \\ \sum_{j=i+1}^n \omega_j \zeta_j, & \text{if } \lambda_{i+1} \le \lambda < \lambda_i \\ \sum_{j=1}^n \omega_j \zeta_j, & \text{if } \beta = \lambda_1 \le \lambda \end{cases}$$

where $A = Q\Lambda Q^T$, $\boldsymbol{\omega} = Q^T\mathbf{w} = (\omega_i)$, and $\boldsymbol{\zeta} = Q^T\mathbf{z} = (\zeta_i)$.

This integral can be approximated by means of Gauss–type quadrature rules, which can be used to obtain lower and upper bounds on the bilinear forms of interest. For example, one can approximate the integral associated with the bilinear form by using the Gauss rule:

$$\int_\alpha^\beta f(\lambda)\,\mathrm{d}m(\lambda) = \sum_{j=1}^p c_j f(t_j), \tag{1.6}$$

where the nodes $\{t_j\}_{j=1}^p$ and weights $\{c_j\}_{j=1}^p$ are to be determined.

The bound for the bilinear form obtained using the Gauss rule is described in the following theorem.

Theorem 1.21. [49, Theorem 6.3] *Suppose f is such that $f^{(2\ell)}(\xi) \ge 0$, $\forall \ell$, $\forall \xi \in (\alpha, \beta)$, and let*

$$L_G[f] = \sum_{j=1}^p c_j f(t_j).$$

The Gauss rule is exact for polynomials of degree less than or equal to $2p-1$ and we have

$$L_G[f] \le \int_\alpha^\beta f(\lambda)\,\mathrm{d}m(\lambda).$$

Moreover $\forall p$ there exists $\eta \in [\alpha, \beta]$ such that

$$\int_\alpha^\beta f(\lambda)\,\mathrm{d}m(\lambda) - L_G[f] = \frac{f^{(2p)}(\eta)}{(2p)!} \int_\alpha^\beta \left[\prod_{j=1}^p (\lambda - t_j)\right]^2 dm(\lambda).$$

Alternatively, one can use the Gauss–Radau quadrature rule:

$$\int_a^b f(\lambda)\,\mathrm{d}m(\lambda) = \sum_{j=1}^p c_j f(t_j) + v_1 f(\tau_1), \tag{1.7}$$

where the nodes $\{t_j\}_{j=1}^p$ and the weights $\left\{\{c_j\}_{j=1}^p, v_1\right\}$ are to be determined, whereas τ_1 is prescribed and is equal to either α or β. The Gauss–Radau bounds are described in the following theorem.

Theorem 1.22. [49, Theorem 6.4] *Suppose f is such that $f^{(2\ell+1)}(\xi) \leq 0$ for all ℓ and for all $\xi \in (\alpha, \beta)$. Let U_{GR} be defined as*

$$U_{GR}[f] = \sum_{j=1}^{p} c_j^{\alpha} f(t_j^{\alpha}) + v_1^{\alpha} f(\alpha),$$

$c_j^{\alpha}, v_1^{\alpha}, t_j^{\alpha}$ being the weights and nodes in (1.7) with $\tau_1 = \alpha$, and let L_{GR} be defined as

$$L_{GR}[f] = \sum_{j=1}^{p} c_j^{\beta} f(t_j^{\beta}) + v_1^{\beta} f(\beta),$$

$c_j^{\beta}, v_1^{\beta}, t_j^{\beta}$ being the weights and nodes in (1.7) with $\tau_1 = \beta$. The Gauss–Radau rule is exact for polynomials of degree less than or equal to $2p$ and satisfies

$$L_{GR}[f] \leq \int_{\alpha}^{\beta} f(\lambda)\, \mathrm{d}m(\lambda) \leq U_{GR}[f].$$

Moreover, for all p there exist $\eta_U, \eta_L \in [\alpha, \beta]$ such that

$$\int_{\alpha}^{\beta} f(\lambda)\, \mathrm{d}m(\lambda) - U_{GR}[f] = \frac{f^{(2p+1)}(\eta_U)}{(2p+1)!} \int_{\alpha}^{\beta} (\lambda - \alpha) \left[\prod_{j=1}^{p} (\lambda - t_j^{\alpha})\right]^2 \mathrm{d}m(\lambda),$$

$$\int_{\alpha}^{\beta} f(\lambda)\, \mathrm{d}m(\lambda) - L_{GR}[f] = \frac{f^{(2p+1)}(\eta_L)}{(2p+1)!} \int_{\alpha}^{\beta} (\lambda - \beta) \left[\prod_{j=1}^{p} (\lambda - t_j^{\beta})\right]^2 \mathrm{d}m(\lambda).$$

The explicit computation of nodes and weights appearing in the quadrature rules can be avoided. Indeed, the evaluation of the quadrature rules is mathematically equivalent to the computation of orthogonal polynomials via a three-term recurrence, or, equivalently, to the computation of entries and spectral information on a certain tridiagonal matrix via the Lanczos algorithm. In fact, the right-hand side of (1.6) can be computed from the relation (see [49, Theorem 6.6]):

$$\sum_{j=1}^{p} c_j f(t_j) = \mathbf{e}_1^T f(J_p) \mathbf{e}_1,$$

where

$$J_p = \begin{pmatrix} \omega_1 & \gamma_1 & & \\ \gamma_1 & \omega_2 & \ddots & \\ & \ddots & \ddots & \gamma_{p-1} \\ & & \gamma_{p-1} & \omega_p \end{pmatrix}$$

is a tridiagonal matrix whose eigenvalues are the Gauss nodes, whereas the weights are given by the squares of the first entry of the normalized eigenvectors of J_p. Similarly, the right hand side of (1.7) can be computed as

$$\sum_{j=1}^{p} c_j f(t_j) + v_1 f(\tau_1) = \mathbf{e}_1^T f(J_{p+1}) \mathbf{e}_1, \tag{1.8}$$

where

$$J_{p+1} = \begin{pmatrix} J_p & \gamma_p \mathbf{e}_p \\ \gamma_p \mathbf{e}_p^T & \omega_{p+1} \end{pmatrix}$$

is a tridiagonal matrix constructed so as to have τ_1 among its eigenvalues. More specifically, in order to construct such J_{p+1}, one uses the Lanczos algorithm to compute J_p and the element γ_p; then, one derives $\omega_{p+1} = \tau_1 + \mathbf{e}_p^T \mathbf{x}^{(\tau_1)}$, where $\mathbf{x}^{(\tau_1)}$ solves the tridiagonal linear system $(J_p - \tau_1 I)\mathbf{x}^{(\tau_1)} = \gamma_p^2 \mathbf{e}_p$ (see [49, p. 89]).

An efficient implementation of this technique is provided in G. Meurant's `mmq` toolbox for MATLAB [72]. This toolbox, adapted to handle sparsity, will be used for some of the numerical experiments presented.

1.4 Fundamentals in network science

In this section we briefly describe some characteristic features of (undirected) complex networks and we describe two models that can be used to construct graphs with these features. Moreover, we review some of the most popular walk-based node centrality measures.

1.4.1 Some properties of complex networks

A complex network can be mathematically described as a graph. Most of the graphs of interest to network scientists are highly sparse and exhibit non-trivial topological

features, in the sense that they are not purely regular nor completely random. However, there are some characteristics which are independent of the particular type of graph under study and that are shared by almost all complex networks.

Let us consider the case of undirected graphs. It is well known that several complex networks (such as the Internet, the World Wide Web, or some social networks) exhibit a degree distribution that is heavy-tailed and, in most cases, approximately follows a power law [79, p. 247ff]. More specifically, the fraction of nodes of degree k decays asymptotically with the degree as $P(k) \sim k^{-\gamma}$, where $\gamma \in [2, 3]$. The graphs having this property are called *scale-free* and their set of nodes contains a few elements that have a large number of connections and a large number of nodes that have a low degree. As an example, the histograms in Figure 1.2 show the degree distribution of two real world networks: ca-GrQc and ca-HepTh (see [26]).

An important feature of real world networks is that they usually have a high *clustering coefficient*. This quantity measures the degree of transitivity of the network, i.e., the extent to which the nodes in the network tend to participate in triangles. It can be computed by averaging over all nodes their *local clustering coefficients*:

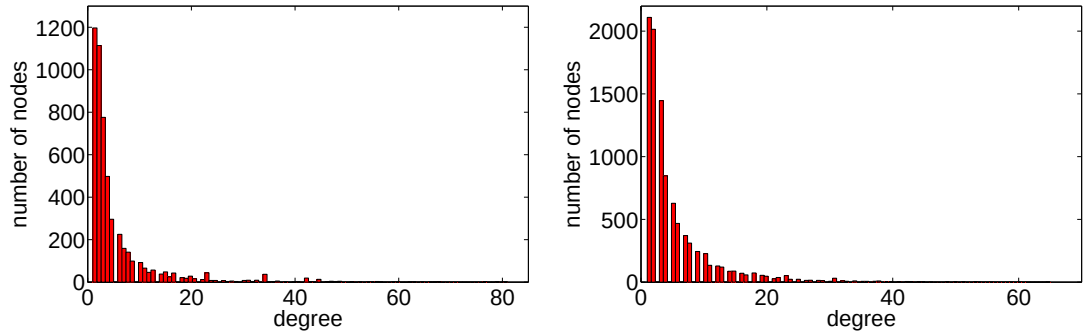
$$C_i = \frac{t_i}{d_i(d_i - 1)},$$

where $t_i = \frac{1}{2}(A^3)_{ii}$ is the number of triangles in which node i participates and $d_i = (A^2)_{ii}$ is its degree. Note that if $d_i < 2$ for some i, then C_i is undefined (as it equals $\frac{0}{0}$). In such cases the local clustering coefficient of the corresponding node can be set to 0, as the node clearly does not participate in any triangle. The (average) clustering coefficient of a graph described by the adjacency matrix A is:

$$\overline{C} = \frac{1}{n}\sum_{j=1}^{n} C_j = \frac{1}{n}\sum_{\substack{j=1 \\ d_j \geq 2}}^{n} \frac{t_j}{d_j(d_j - 1)}. \tag{1.9}$$

Complex networks often display also a small *average path length*. This quantity measures the average number of steps the information has to take in order to get from its source to the target, by only going through paths. Before giving its mathematical description, we need to introduce the concept of *shortest path distance* or *geodesic distance* between two nodes. The geodesic distance between two distinct nodes i and j, $\mathrm{d}(i, j)$, is defined

FIGURE 1.2: Degree distribution of two real-world networks. (Figure from [13].)



(a) Network: ca-GrQc

(b) Network: ca-HepTh

as the length of the shortest path connecting these nodes. Moreover, we assume by convention that $\mathrm{d}(i,i) = 0$, so that the introduced quantity is indeed a distance. Using this definition, we can define the average path length of a connected network as:

$$\overline{\ell} = \frac{1}{2m} \sum_{i,j \in \mathcal{V}} \mathrm{d}(i,j). \tag{1.10}$$

The notion of geodesic distance is strictly related to the so called *small world property*. This feature, shared by most complex networks, ensures that every node can be reached in only a few steps from every other node in the graph. This property can be formulated by saying that complex networks have, in general, a small *diameter*, which is defined as:

$$\mathrm{diam}(\mathcal{G}) = \max_{i,j \in \mathcal{V}} \mathrm{d}(i,j).$$

Typically, networks with the small world property are such that $\mathrm{diam}(\mathcal{G}) = \mathcal{O}(\log n)$. Moreover, they also display a large average clustering coefficient.

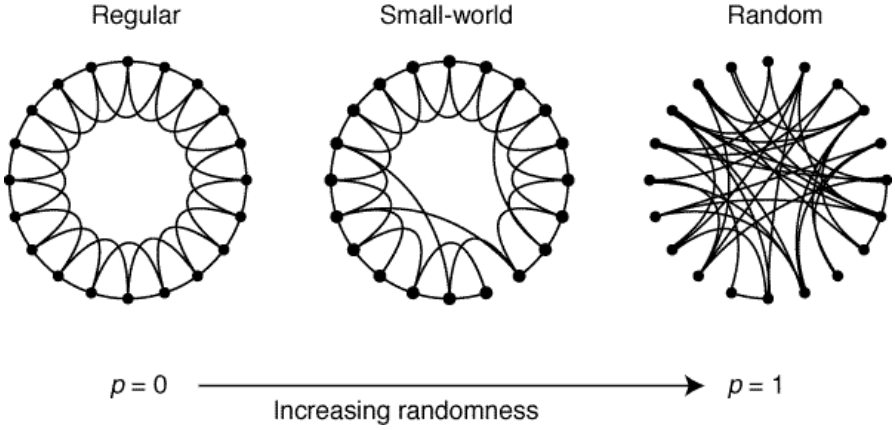
1.4.2 The Watts–Strogatz model

A well known generative network model is the *Watts–Strogatz model* [96]. It generates graphs that have a high clustering coefficient and a small diameter, i.e., the small world property.

The n nodes are originally arranged in a ring and connected to their $k \ll n$ nearest neighbors. Then each node is considered independently and each one of its incident edges is either rewired with probability p or left as it is, with probability $(1-p)$. This

t

FIGURE 1.3: The Watts–Strogatz model: how the topology of the network changes with the probability p. (Figure from [96].)



model interpolates between a k-regular graph ($p = 0$), where $\text{diam}(\mathcal{G}) = \mathcal{O}(n)$ and thus no small world effect can be observed, and a random network ($p = 1$), which has a small diameter, but has a low clustering coefficient $\overline{C} = \mathcal{O}(n^{-1})$ (see Figure 1.3).

The CONTEST [87, 88] toolbox for MATLAB contains a function that generates instances of the Watts–Strogatz model. The adjacency matrix associated with a graph of this type is generated by the function `smallw(n,k,p)`. Here n is the number of nodes in the network, k is the number of nearest neighbors each node is originally connected to, and $p \in [0, 1]$ is a probability. The default parameter settings, which we will use in our tests, are $k = 2$ and $p = 0.1$. Note that this code does not allow the formation of self-loops or multiple edges.

1.4.3 The Barabási–Albert model

The best known generative network model that generates an undirected scale-free graph with the small world property is the *Barabási–Albert*, or *preferential attachment model* [8]. In this model the nodes are added one by one to the network and they are connected to a certain subset of other nodes in the graph. Each node i creates exactly $d \geq 1$ connections, where d is a fixed integer, and an edge is created between i and j with a probability that is proportional to d_j, the degree of node j. This method ensures that, when the procedure ends, no nodes have degree less than d.

The function used to build the adjacency matrices associated with this type of graphs in CONTEST is `pref(n,d)`, where n is the number of nodes and $d \geq 1$ is the number

TABLE 1.1: Node centrality measures.

Centrality measure	Undirected graph	Directed graph	
		source	target
degree	$d_i = (A\mathbf{1})_i$	$d_{out}(i) = (A\mathbf{1})_i$	$d_{in}(i) = (A^T\mathbf{1})_i$
eigenvector	$\mathbf{q}_1(i)$	$\mathbf{x}_1(i)$	$\mathbf{y}_1(i)$
f-subgraph	$\mathbf{e}_i f(A)\mathbf{e}_i$	$\mathbf{e}_i f(\mathscr{A})\mathbf{e}_i$	$\mathbf{e}_{i'} f(\mathscr{A})\mathbf{e}_{i'}$
f-total communicability	$[f(A)\mathbf{1}]_i$	$[f(A)\mathbf{1}]_i$	$\left[f(A^T)\mathbf{1}\right]_i$
HITS	—	$\mathbf{u}_1(i)$	$\mathbf{v}_1(i)$

of edges each new node is given when it is first introduced to the network. The default parameter setting, which we will use in the tests, is $d = 2$.

1.4.4 Centrality measures

We now review some of the most popular walk-based node centrality measures that will be used throughout this thesis.

In the general setting, a node centrality measure is a function

$$C : \mathcal{V} \longrightarrow \mathbb{R}_{\geq 0}$$

which is invariant under graph isomorphism[1] and that assigns to each node in the graph a nonnegative score that quantifies its importance within the network. In undirected networks, each node will be assigned one score, while it will be assigned two in the directed case: one to describe its importance as a spreader of information and one to describe its importance as a receiver.

The centrality measures introduced in the remaining of this subsection are summarized in Table 1.1.

In the list that will follow we omitted some very popular measures of centrality, such as the *closeness* and the *betweenness centrality* (see, e.g., [79, Chapter 7]). We decided to do so because these measures do not have a simple formulation in terms of matrix properties, as they are based on the assumption that communication always occurs through paths, and in this thesis we are concerned predominantly with linear algebraic techniques.

[1]Two graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ with associate adjacency matrices A_1 and A_2 are isomorphic if there exists a permutation matrix P such that $A_1 = PA_2P^T$ (see [47, Lemma 8.1.1]).

Degree centrality

The easiest and most local measure of centrality, in both the undirected and directed case, is the *degree centrality*. It assigns to each node a score that is equal to the number of its neighbors. In the undirected case, the degree centrality of node i is defined as d_i. In the directed case, the two quantities used are $d_{out}(i) = |\{j \in \mathcal{V} : (i,j) \in \mathcal{E}\}|$, namely the number of edges of the form $i \to *$, for the importance as a source, and $d_{in}(i) = |\{j \in \mathcal{V} : (j,i) \in \mathcal{E}\}|$, i.e., the number of edges of the form $* \to i$, for the importance as a target of node i.

As it is clear from their definitions, the measures of centrality based on the degree of the nodes account for too little information to be able to clearly identify the most important nodes in the network. For example, if we think of two complete graphs K_n ($n > 3$) linked through a path P_3, one of the most important nodes in the network is the middle node in P_3. Indeed, the information has to go necessarily through this node in order to flow from one complete graph to the other. However, the degree of this node is the lowest, since it equals 2 while the degrees of the other nodes equal $n > 3$ (for the other two nodes in the path) or $n - 1 > 2$ (for the remaining nodes in the graph).

Eigenvector centrality

In [18] the author introduced the *eigenvector centrality* of nodes. For each node i, it is defined as the ith component of the eigenvector associated with the leading eigenvalue of the adjacency matrix. When the undirected network is connected, from Theorem 1.10 it follows that there is a unique normalized eigenvector $\mathbf{q}_1$ associated with the simple eigenvalue $\lambda_1 > 0$ that can be chosen to have positive entries, and thus the eigenvector centrality of nodes is well defined.

The eigenvector centrality of a node i can be interpreted as the average amount of time a random walker spends in a given node as the length of the walks tends to infinity (see [34, p. 127] and references therein). Furthermore, from $A\mathbf{q}_1 = \lambda_1 \mathbf{q}_1$ it follows that

$$q_1(i) = \frac{1}{\lambda_1} \sum_{j=1}^{n} a_{ij} q_1(j) = \frac{1}{\lambda_1} \sum_{j:(i,j)\in\mathcal{E}} q_1(j),$$

and thus the centrality of a node i can be expressed as a linear combination of the centralities of its adjacent nodes. Thus, in some sense, the eigenvector centrality extends the degree centrality of nodes in order to account for more information than simply the number of immediate neighbors. Indeed, we now take into account the importance of the neighboring nodes to determine the eigenvector centrality of each node.

For the case of digraphs, for each node i one can consider the ith components of the right and left eigenvectors associated with λ_1 as measures of centrality as broadcaster and receiver of information, respectively. These centrality measures are well defined only for the case of strongly connected networks, when we can ensure that the two eigenvectors associated with the simple eigenvalue λ_1 can be chosen to be non-negative.

The computation of this centrality index will be carried out, in all our numerical tests, using the MATLAB built-in function `eigs`.

f-subgraph centrality

The *f-subgraph centrality* of node i is defined, in the undirected case, as [42]:

$$\mathbf{e}_i^T f(A)\mathbf{e}_i,$$

where $f : \mathbb{R} \longrightarrow \mathbb{R}$ is a function defined on the spectrum of A. The two most widely used f-subgraph centrality measures are:

- *(exponential) subgraph centrality* of a node i: $\left(e^A\right)_{ii}$;
- *resolvent subgraph centrality* of a node i: $\left[(I-\alpha A)^{-1}\right]_{ii}$, with $\alpha \in (0, \lambda_1^{-1})$.

Remark 1.23. The parameter α in the definition of the resolvent subgraph centrality has to fall in the interval $(0, \lambda_1^{-1})$ so that the power expansion

$$(I-\alpha A)^{-1} = \sum_{k=0}^{\infty} (\alpha A)^k$$

is convergent and non-negative [59].

The f-subgraph centrality of a node accounts for how much information departing from a certain node will return to it. This interpretation follows from the Maclaurin expansion

of the matrix function of interest and from the fact that the powers of the adjacency matrix can be used to count (closed) walks taking place in the network. Indeed, if $f(z) = \sum_{k=0}^{\infty} a_k z^k$ with $a_k \geq 0$ for all k, the f-subgraph centrality of a node i can be expressed, as

$$\mathbf{e}_i^T f(A)\mathbf{e}_i = \sum_{k=0}^{\infty} a_k (A^k)_{ii},$$

where $a_k \to 0$ as $k \to \infty$. Therefore, we are counting all the closed walks of any lengths centered at node i and we are giving more importance to shorter ones.

Remark 1.24. As for the off-diagonal entries of the matrix $f(A)$, they represent the *communicabilities* between nodes, i.e., a measure of the amount of information two nodes in the network exchange if we allow information to flow along walks of up to infinite length. Following [39], given $i, j \in \mathcal{V}$ we will call *f-communicability between nodes i and j* the bilinear form $\mathbf{e}_i^T f(A)\mathbf{e}_j$. Specifically, we will call *subgraph communicability between nodes i and j* the quantity $\left(e^A\right)_{ij}$.

A generalization of this family of centrality measures to the directed case was given in [11]. The authors proposed to work on the $2n \times 2n$ matrix (1.1) of the bipartite graph associated with the digraph under study. Thus, if $f : \mathbb{R} \longrightarrow \mathbb{R}$ is a function defined on the spectrum of $\mathscr{A}$, then the first n diagonal entries:

$$\mathbf{e}_i^T f(\mathscr{A})\mathbf{e}_i, \qquad i \in \mathcal{V}$$

provide centrality measures for the nodes in their role of broadcasters; similarly, the last n diagonal entries

$$\mathbf{e}_{i'}^T f(\mathscr{A})\mathbf{e}_{i'}, \qquad i' \in \mathcal{V}'$$

provide the centrality indices for the nodes in their role of receivers of information.

To estimate these quantities in the numerical tests we make use of the `mmq` [72] toolbox for MATLAB, which uses five Lanczos steps per estimate to compute the Gauss–Radau quadrature rule associated with the bilinear form of interest (see Section 1.3 and [49] for more details).

Starting from this type of centrality measures, in [32] a global index for the networks was introduced. This value, later called the *Estrada index* of the graph, is defined as

$$EE(A) = \mathrm{Tr}(e^A) = \sum_{j=1}^{n} e^{\lambda_j} = \sum_{j=1}^{n} (e^A)_{jj}, \tag{1.11}$$

where Tr denotes the trace of the matrix.

From its definition, it is clear that the Estrada index accounts for the total returnability of the information in the network. Moreover, as we will later see in Chapter 2, this index can also be interpreted as the *partition function* of the network (see [34, 38]).

Node f-total communicability

The notion of *node f-total communicability* was first introduced for undirected networks in [13], although a special case of node f-total communicability already existed in the literature since sixty years (see [59]).

Let $f : \mathbb{R} \longrightarrow \mathbb{R}$ be a function defined on the spectrum of A, then the *f-total communicability of node i* is defined as:

$$[f(A)\mathbf{1}]_i.$$

In [11] the authors focused on two particular cases: the exponential-based total communicability $(e^A\mathbf{1})_i$, which is usually referred to as *node total communicability*, and the well-known *Katz centrality* [59]:

$$\left[(I - \alpha A)^{-1}\mathbf{1}\right]_i, \qquad \text{with } \alpha \in \left(0, \lambda_1^{-1}\right).$$

When considering the node f-total communicability, each vertex is assigned a score given by a weighted sum of walks to every node in the network, including the node itself. Indeed, if $f(z) = \sum_{k=0}^{\infty} a_k z^k$ with $a_k \geq 0$ for all k, the f-total communicability of a node i can be expressed, as

$$\mathbf{e}_i^T f(A)\mathbf{1} = \sum_{k=0}^{\infty} a_k (A^k\mathbf{1})_i,$$

where $a_k \to 0$ as $k \to \infty$. Thus, this score quantifies the ability of a node to spread information across the network and the returnability of the information to the node itself.

The definition of node f-total communicability, provided for the case of undirected networks, can be easily extended to the case of digraphs. The centralities of the nodes as sources are obtained by applying the function f to the matrix A, while the centralities of the nodes as targets are obtained by applying the function to the matrix A^T.

The computation of the vector $f(A)\mathbf{1}$ of node f-total communicabilities can be performed efficiently using Krylov methods (see [56, Chapter 13]). In this work, we have used S. Güttel's implementation: the `funm_kryl` toolbox [52] for MATLAB. In the numerical tests we make use of this toolbox with the default parameter settings.

Similarly to what has been done for the f-subgraph centrality measures, one can define a global index for the network associated to the node f-total communicability. For a generic function f defined on the spectrum of the adjacency matrix, the *f-total (network) communicability* is defined as:

$$TC(A, f) := \mathbf{1}^T f(A)\mathbf{1}. \tag{1.12}$$

This index is used to quantify the overall ability of a network to diffuse information.

When the function $f(t) = e^t$ is considered, we will simply write $TC(A)$ and we will call this index the *total (network) communicability*. As it is clear from its definition, the value of $TC(A)$ may be very large. Several normalizations have been proposed; the simplest is the normalization by the number of nodes n (see [13]), which we will use throughout this thesis. It is easy to prove that the normalized total communicability satisfies [13]:

$$\frac{EE(A)}{n} \leq \frac{TC(A)}{n} \leq e^{\lambda_1}, \tag{1.13}$$

where the lower bound is attained by the empty graph with n nodes, whereas the upper bound is attained by the complete graph K_n.

HITS

The last centrality measure we review here is defined only for the case of digraphs and was introduced in [60] by Kleinberg, who stated that in directed networks there exist two types of important nodes: *hubs* and *authorities*. In particular, each node can be assigned a hub score and an authority score, which quantify its ability of playing these two roles. Good hubs are those nodes which better broadcast information, while good authorities are those which better receive information. These two types of importance for vertices are strongly related through a recursive definition: the importance of a node as a hub is proportional to the importance as authorities of the nodes it points to. Similarly, the importance of a vertex as an authority depends on the importance as hubs of the nodes that point to it. This recursive definition is highlighted in the implementation of the HITS algorithm (see [60]), which makes use of the eigenvectors corresponding to the leading eigenvalue of the symmetric positive semi-definite matrices AA^T and A^TA to rank the nodes as hubs and authorities, respectively. The eigenvectors associated to σ_1^2 are $\mathbf{u}_1$ and $\mathbf{v}_1$, the first left and right singular vectors of the matrix A. In the following, these vectors will be referred to as the *hub* and *authority vector*, respectively.

Chapter 2

Tuning the total communicability of undirected graphs

The total communicability provides a good measure of how efficiently information (in the broad sense of the term) is diffused across the network. Typically, very high values of $TC(A)$ are observed for highly optimized infrastructure networks (such as airline routes or computer networks) and for highly cohesive social and information networks (like certain type of collaboration networks). Conversely, the total network communicability is relatively low for spatially extended, grid-like networks (such as many road networks) or for networks that consist of two or more communities with poor communication between them (such as the Zachary network).[1]

Moreover, the total communicability is closely related to the *natural connectivity* (or *free energy*) of the network, while being dramatically easier to compute; see Section 2.5 below. Sparse networks with high values of $TC(A)$ are very well connected and thus less likely to be disrupted by either random failures or targeted attacks leading to the loss of edges. This justifies trying to design sparse networks with high values of the total communicability.

Finally, in view of the bounds (1.13), the evolution of the total communicability under network modifications is closely tied to the evolution of the dominant eigenvalue λ_1. This quantity plays a crucial role in network analysis, for example in the definition of

[1] Numerical values of the normalized total network communicability for a broad collection of networks are reported in Section 2.4 and in [13].

the *epidemic threshold*; see, for instance, [79, p. 664] and [92]. In particular, a decrease in the total network communicability can be expected to lead to an increase in the epidemic threshold.

Thus, being able of modifying an existing graph so as to tune its total network communicability can be of interest in several settings. In this chapter, which is mainly based on the results presented in [4], we discuss how to add edges to a network so as to increase as much as possible its total communicability, and how to remove connections in order not to penalize this index too much. The redirection of existing edges is also considered.

The remaining of the chapter is organized as follows. In Section 2.1 we describe bounds for the total communicability via the Gauss–Radau quadrature rule and we show how these bounds change when a rank-two modification of the adjacency matrix is performed. Section 2.2 is devoted to the introduction of methods to controllably modify the graph in order to adjust the value of its total communicability. Numerical studies to assess the effectiveness and performance of the techniques introduced are provided in Section 2.4 for both synthetic and real-world networks. In Section 2.5 we discuss the evolution of a popular measure of network connectivity, known as the *free energy* (or *natural connectivity*), when the same modifications are performed. This section provides further evidence that motivates the use of the total communicability as a measure of connectivity. Finally, in Section 2.6 we discuss how the same problems can be tackled when the *resolvent-based total communicability*, defined as $TC(A, f_\alpha) = \mathbf{1}^T(I - \alpha A)^{-1}\mathbf{1}$, is considered as target function.

2.1 Bounds via quadrature rules

In [13] the authors provide simple bounds on the normalized total network communicability, see equation (1.13). More refined bounds for this index can be obtained by means of quadrature rules as described in [10, 12, 44, 49].

Indeed, bounds on bilinear forms $\mathbf{u}^T f(A)\mathbf{v}$ can be derived based on Gauss–type quadrature rules when f is a completely monotonic function on the interval $[\alpha, \beta]$ containing the spectrum of A by working on a 2×2 matrix derived from one step of the symmetric Lanczos iteration (see [12, 49]). Observe that the exponential function e^x is not completely monotonic, while e^{-x} is. This means that, in order to compute bounds on the

normalized total communicability, we need to use the function $f(x) = e^{-x}$ and therefore to work on the matrix $-A$:

$$\frac{TC(A)}{n} = \left(\frac{\mathbf{1}}{\sqrt{n}}\right)^T e^{-(-A)} \left(\frac{\mathbf{1}}{\sqrt{n}}\right).$$

The following theorem contains our result on the bounds for the normalized total communicability.

Theorem 2.1. *Let A be the adjacency matrix of an unweighted and undirected network. Then*

$$\Phi\left(\beta, \omega_1 + \frac{\gamma_1^2}{\omega_1 - \beta}\right) \leq \frac{TC(A)}{n} \leq \Phi\left(\alpha, \omega_1 + \frac{\gamma_1^2}{\omega_1 - \alpha}\right)$$

where $[\alpha, \beta]$ is an interval containing the spectrum of $-A$ (i.e., $\alpha \leq -\lambda_1$ and $\beta \geq -\lambda_n$), $\omega_1 = -\mu = -\frac{1}{n}\sum_{i=1}^n d_i$ is the negative mean of the degrees, $\gamma_1 = \sigma = \sqrt{\frac{1}{n}\sum_{k=1}^n (d_k - \mu)^2}$ is the standard deviation, and

$$\Phi(x, y) = \frac{c\,(e^{-x} - e^{-y}) + xe^{-y} - ye^{-x}}{x - y}, \qquad c = \omega_1. \tag{2.1}$$

Proof. First we derive an explicit expression for the right-hand side of equation (1.8) when $f(x) = e^{-x}$ and

$$J_2 = \begin{pmatrix} \omega_1 & \gamma_1 \\ \gamma_1 & \omega_2 \end{pmatrix}$$

with the help of the Lagrange interpolation formula for the evaluation of matrix functions [56, p. 6]. Let μ_1 and μ_2 be distinct eigenvalues of a given 2×2 matrix $B = (b_{ij})$, then

$$e^{-B} = \frac{e^{-\mu_1}}{\mu_1 - \mu_2}(B - \mu_2 I) + \frac{e^{-\mu_2}}{\mu_2 - \mu_1}(B - \mu_1 I)$$

where I is the 2×2 identity matrix. It follows that

$$\mathbf{e}_1^T \left(e^{-B}\right) \mathbf{e}_1 = \frac{b_{11}(e^{-\mu_1} - e^{-\mu_2}) + \mu_1 e^{-\mu_2} - \mu_2 e^{-\mu_1}}{\mu_1 - \mu_2}.$$

Next, we build explicitly the matrix J_2 and compute its eigenvalues. The values of $\omega_1 = -\mu$ and $\gamma_1 = \sigma$ are derived applying one step of Lanczos iteration to the matrix $-A$ with starting vectors $\mathbf{x}_{-1} = \mathbf{0}$ and $\mathbf{x}_0 = \frac{1}{\sqrt{n}}\mathbf{1}$. We want to compute the value of ω_2 in such a way that the matrix J_2 has the prescribed eigenvalue $\tau_1 = \alpha$ or $\tau_1 = \beta$. Note

that $\gamma_1 = 0$ if and only if the graph is regular. In such case we simply take $\omega_2 = \tau_1$ and the matrix J_2 is diagonal with eigenvalues $\mu_1 = -\mu$ and $\mu_2 = \tau_1$. Thus, let us assume $\gamma_1 \neq 0$. In order to compute the value for ω_2, we use the three-term recurrence for orthogonal polynomials:

$$\gamma_j p_j(\lambda) = (\lambda - \omega_j)p_{j-1}(\lambda) - \gamma_{j-1}p_{j-2}(\lambda), \quad j = 1, 2, \ldots, p,$$

with $p_{-1}(\lambda) \equiv 0$, $p_0(\lambda) \equiv 1$ to impose that $p_2(\tau_1) = 0$ and hence derive $\omega_2 = \tau_1 - \frac{\gamma_1}{p_1(\tau_1)}$. Using the same recurrence, we also find that $p_1(\tau_1) = \frac{\tau_1 - \omega_1}{\gamma_1}$ which is nonzero, since the zeros of orthogonal polynomials satisfying the three-term recurrence are distinct and lie in the interior of $[\alpha, \beta]$ (see [49, Theorem 2.14]).

Finally, the matrix

$$J_2 = \begin{pmatrix} \omega_1 & \gamma_1 \\ \gamma_1 & \tau_1 - \frac{\gamma_1^2}{\tau_1 - \omega_1} \end{pmatrix}$$

has (distinct) eigenvalues $\mu_1 = \tau_1$ and $\mu_2 = \omega_1 + \frac{\gamma_1^2}{\omega_1 - \tau_1}$. This, together with Theorem 1.22 and the relation (1.8), concludes the proof. □

Following the same procedure, analogous bounds can be found for the adjacency matrix of the graph after performing a downdate or an update. These results are summarized in the following Corollaries.

Corollary 2.2. [Downdating] *Let $\widehat{A} = A - \mathbf{e}_i\mathbf{e}_j^T - \mathbf{e}_j\mathbf{e}_i^T$ be the adjacency matrix of an unweighted and undirected network obtained after the downdate of the edge $(i, j) \in \mathcal{E}$ from the matrix A. Let $\omega_1 = -\mu = -\frac{1}{n}\sum_{j=1}^n d_j$ and $\gamma_1 = \sigma = \sqrt{\frac{1}{n}\sum_{j=1}^n (d_j - \mu)^2}$, where d_i is the degree of node i in the original graph. Then*

$$\Phi\left(\beta_-, \omega_- + \frac{\gamma_-^2}{\omega_- - \beta_-}\right) \leq \frac{TC(\widehat{A})}{n} \leq \Phi\left(\alpha_-, \omega_- + \frac{\gamma_-^2}{\omega_- - \alpha_-}\right)$$

where

$$\begin{cases} \omega_- = \omega_1 + \frac{2}{n}; \\ \gamma_- = \sqrt{\gamma_1^2 - \frac{2}{n}\left(d_i + d_j - 1 + 2\omega_1 + \frac{2}{n}\right)} \end{cases},$$

α_- and β_- are approximation of the smallest and largest eigenvalues of $-\widehat{A}$ respectively, and Φ is defined as in (5.4) *with $c = \omega_-$.*

Note that if bounds α and β for the extremal eigenvalues of the original matrix are known, we can then use $\alpha_- = \alpha$ and $\beta_- = \beta + 1$. Indeed, if we order the eigenvalues of $\widehat{A}$ in non–increasing order $\widehat{\lambda}_1 \geq \widehat{\lambda}_2 \geq \cdots \geq \widehat{\lambda}_n$ we obtain, as a consequence of Theorem 1.13, that

$$\alpha - 1 \leq -\lambda_1 - 1 < -\widehat{\lambda}_1 \leq -\widehat{\lambda}_2 \leq \cdots \leq -\widehat{\lambda}_n \leq -\lambda_n + 1 \leq \beta + 1.$$

Furthermore, Corollary 1.12 ensures that, when performing a downdate, the largest eigenvalue of the adjacency matrix cannot increase; hence, we deduce the more stringent bounds $\alpha \leq -\widehat{\lambda}_1 \leq -\widehat{\lambda}_2 \leq \cdots \leq -\widehat{\lambda}_n \leq \beta + 1$.

Similarly, we can derive bounds for the normalized total communicability of the matrix $\widetilde{A}$ obtained from the matrix A after performing the update of the virtual edge $(i, j) \in \overline{\mathcal{E}}$.

Corollary 2.3. [Updating] *Let $\widetilde{A} = A + \mathbf{e}_i\mathbf{e}_j^T + \mathbf{e}_j\mathbf{e}_i^T$ be the adjacency matrix of an unweighted and undirected network obtained after the update of the virtual edge (i, j) in the network associated with the matrix A. Let $\omega_1 = -\mu = -\frac{1}{n}\sum_{j=1}^{n} d_j$ and $\gamma_1 = \sigma = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(d_j - \mu)^2}$, where d_i is the degree of node i in the original graph. Then*

$$\Phi\left(\beta_+, \omega_+ + \frac{\gamma_+^2}{\omega_+ - \beta_+}\right) \leq \frac{TC(\widetilde{A})}{n} \leq \Phi\left(\alpha_+, \omega_+ + \frac{\gamma_+^2}{\omega_+ - \alpha_+}\right)$$

where

$$\begin{cases} \omega_+ = \omega_1 - \frac{2}{n}; \\ \gamma_+ = \sqrt{\gamma_1^2 + \frac{2}{n}\left(d_i + d_j + 1 + 2\omega_1 - \frac{2}{n}\right)} \end{cases},$$

α_+ and β_+ are bounds for the smallest and largest eigenvalues of $-\widetilde{A}$ respectively, and Φ is defined as in (5.4) *with $c = \omega_+$.*

Notice that again, if bounds α and β for the extremal eigenvalues of $-A$ are known, we can then take $\alpha_+ = \alpha - 1$ and $\beta_+ = \beta$. In fact, the spectrum of the rank-two symmetric perturbations $\pm\left(\mathbf{e}_i\mathbf{e}_j^T + \mathbf{e}_j\mathbf{e}_i^T\right)$ is $\{\pm 1, 0\}$ and hence we can use Theorem 1.13 as before and then improve the upper bound using Corollary 1.12.

In the next section we will see how the new bounds can be used to guide the updating and downdating process.

2.2 Modifications of the adjacency matrix

In this section we develop techniques that allow us to tackle the following problems.

(P1) Downdate: select K edges that can be downdated from the network without disconnecting it and that cause the smallest drop in the total communicability of the graph;

(P2) Update: select K edges to be added to the network (without creating self–loops or multiple edges) so as to increase as much as possible the total communicability of the graph;

(P3) Rewire: select K edges to be rewired in the network so as to increase as much as possible the value of $TC(A)$. The rewiring process must not disconnect the network or create self–loops or multiple edges in the graph.

As we will show below, (P3) can be solved using combinations of methods developed to solve (P1) and (P2). Hence, we first focus on the downdate and the update separately. Note that to decrease as little as possible the total communicability when removing an edge it would suffice to select $(i^*, j^*) \in \mathcal{E}$ so as to minimize the quantities

$$\mathbf{1}^T A^k \mathbf{1} - \mathbf{1}^T (A - \mathbf{e}_i \mathbf{e}_j^T - \mathbf{e}_j \mathbf{e}_i^T)^k \mathbf{1} \qquad \forall k = 1, 2, \ldots,$$

since $TC(A) = \sum_{k=0}^{\infty} \frac{\mathbf{1}^T A^k \mathbf{1}}{k!}$. Similarly, to increase as much as possible $TC(A)$ by addition of a virtual edge, it would suffice to select $(i^*, j^*) \in \overline{\mathcal{E}}$ that maximizes the differences

$$\mathbf{1}^T (A + \mathbf{e}_i \mathbf{e}_j^T + \mathbf{e}_j \mathbf{e}_i^T)^k \mathbf{1} - \mathbf{1}^T A^k \mathbf{1} \qquad \forall k = 1, 2, \ldots$$

However, it is easy to show that in general one cannot find a choice for (i^*, j^*) that works for all such k. Numerical experiments on small synthetic graphs showed that in general the optimal edge selection for $k = 2$ is different from the one for $k = 3$. Indeed, we have considered 100 synthetic examples built using the functions `pref(100)` and `smallw(100)` from the CONTEST toolbox for MATLAB (see Section 1.4 for more details). For each one of these matrices, we have selected the update that maximizes the difference for $k = 2$ and the one which maximizes the difference for $k = 3$. Then, we have computed how many times the two selected edges coincide. We have iterated this

procedure 10 times. On average, the selected edges are the same 89.6% of the time, when using the preferential attachment model, and 33.3% of the time, when using the small world model. From this simple test it clearly follows that it is unlikely that one can find a simple "closed form solution" to the problem, and we need to develop approximation techniques.

The majority of the heuristics we will develop are based on new edge centrality measures. The idea underlying these is that it seems reasonable to assume that an edge is more likely used as communication channel if its adjacent nodes are given a lot of information to spread. We thus introduce three new centrality measures for edges based on this principle: edges connecting important nodes are themselves important.

Definition 2.4. For any $i, j \in \mathcal{V}$ we define the *edge subgraph centrality* of an existing/virtual edge (i, j) as

$$^{e}\mathcal{SC}(i,j) = \left(e^{A}\right)_{ii}\left(e^{A}\right)_{jj}.$$

This definition, based on the subgraph centrality of nodes, exploits the fact that the matrix exponential is symmetric positive definite and hence $(e^A)_{ii}(e^A)_{jj} > (e^A)_{ij}^2$. Therefore, the diagonal elements of e^A somehow control its off-diagonal entries, hence they may contain enough information to infer the "payload" of the edges or of the virtual edges of interest.

Definition 2.5. For any $i, j \in \mathcal{V}$ we define the *edge total communicability centrality* of an existing/virtual edge (i, j) as

$$^{e}\mathcal{TC}(i,j) = [e^{A}\mathbf{1}]_i[e^{A}\mathbf{1}]_j.$$

It is important to observe that when the spectral gap $\lambda_1 - \lambda_2$ is "large enough", then the subgraph centrality $\left(e^{A}\right)_{ii}$ and the total communicability centrality $[e^A\mathbf{1}]_i$ are essentially determined by $e^{\lambda_1}q_1(i)^2$ and $e^{\lambda_1}q_1(i)\|\mathbf{q}_1\|_1$, respectively (see, e.g., [13, 14, 33]); it follows that in this case the two centrality measures introduced and a centrality measure based on the eigenvector centrality for nodes can be expected to provide similar rankings.

This is especially true when attention is restricted to the top edges (or nodes). This observation motivates the introduction of the following edge centrality measure.

Definition 2.6. For any $i, j \in \mathcal{V}$ we define the *edge eigenvector centrality* of an existing/virtual edge (i, j) as

$$^{e}\mathcal{EC}(i,j) = q_1(i)q_1(j).$$

As a further justification for this definition, note that

$$\lambda_1 - 2\,(^{e}\mathcal{EC}(i,j)) \leq \widehat{\lambda}_1 \leq \lambda_1, \qquad \widetilde{\lambda}_1 \geq \lambda_1 + 2\,(^{e}\mathcal{EC}(i,j))\,,$$

where $\widehat{\lambda}_1$ is the leading eigenvalue of the adjacency matrix $\widehat{A} = A - \mathbf{e}_i\mathbf{e}_j^T - \mathbf{e}_j\mathbf{e}_i^T$ of the graph obtained after the downdate of $(i, j) \in \mathcal{E}$, and $\widetilde{\lambda}_1$ is the leading eigenvalue of the adjacency matrix $\widetilde{A} = A + \mathbf{e}_i\mathbf{e}_j^T + \mathbf{e}_j\mathbf{e}_i^T$ of the graph obtained after the update of $(i, j) \in \overline{\mathcal{E}}$. These inequalities show that the edge eigenvector centrality of an (virtual) edge (i, j) is strictly connected to the change in the value of the leading eigenvalue of the adjacency matrix, which influences the evolution of the total communicability when we modify A. Indeed, since we can write

$$TC(A) = \sum_{k=1}^{n} e^{\lambda_k}(\mathbf{q}_k^T\mathbf{1})^2,$$

it is clear that the main contribution to the value of $TC(A)$ is likely to come from the term $e^{\lambda_1}\|\mathbf{q}_1\|_1^2$, especially when $\lambda_1 \gg \lambda_2$.

Remark 2.7. The edge eigenvector centrality has been used in [89, 92] to devise edge removal techniques aimed to reduce significantly λ_1, so as to increase the *epidemic threshold* of networks.

Note that we defined these measures of centrality for both existing and virtual edges (as in [15]). The reason for this as well as the justification for these definitions will become clear in the next subsections.

We now discuss how to use these definitions to deal with the problems previously described.

Algorithm 1: Downdating algorithm with connectivity check.

Data: Initial graph $\mathcal{G}$, $K \in \mathbb{N}$, $greedy \in \{0, 1\}$
Result: Set $\mathcal{S}$ of K edges to be removed
$\mathcal{S} = \varnothing$;
$c = 0$;
$\mathcal{E}$ = list of edges in the graph;
if *greedy* **then**
  while $(c < K)$ && $(\mathcal{E} \neq \varnothing)$ **do**
    $found\_edge = 0$;
    Compute the centrality measure of interest $\forall (i, j) \in \mathcal{E}$;
    while $(found\_edge == 0)$ && $(\mathcal{E} \neq \varnothing)$ **do**
      $\mathcal{G}' = \mathcal{G}$;
      s = element in $\mathcal{E}$ with the smallest centrality;
      Downdate s from $\mathcal{G}'$;
      if $\mathcal{G}'$ *is connected* **then**
        $\mathcal{G} = \mathcal{G}'$;
        $found\_edge = 1$;
        $\mathcal{S} = \mathcal{S} \cup \{s\}$;
        $c = c + 1$;
      end
      $\mathcal{E} = \mathcal{E} \setminus \{s\}$;
    end
  end
else
  $l = 1$;
  Compute edge centrality measure of interest $\forall (i, j) \in \mathcal{E}$;
  Sort the edges in ascending order;
  while $(c < K)$ && $(l \leq |\mathcal{E}|)$ **do**
    $\mathcal{G}' = \mathcal{G}$;
    $s = l$th edge in the sorted array;
    Downdate s from $\mathcal{G}'$;
    if $\mathcal{G}'$ *is connected* **then**
      $\mathcal{G} = \mathcal{G}'$;
      $c = c + 1$;
      $\mathcal{S} = \mathcal{S} \cup \{s\}$;
    end
    $l = l + 1$;
  end
end
Return $\mathcal{S}$.

(P1) Downdate

The downdate of any edge in the network will result in a reduction of its total communicability. Note that since we are focusing on the case of connected networks, we will only perform downdates that keep the resulting graph connected. In practice, it is desirable to further restrict the choice of downdates to a subset of all existing edges, on the basis of criteria to be discussed shortly.

An "optimal" approach would select at each step of the downdating process a candidate

Algorithm 2: Downdating algorithm without connectivity check.

Data: Initial graph $\mathcal{G}$, $K \in \mathbb{N}$, *greedy* $\in \{0, 1\}$
Result: Set $\mathcal{S}$ of K edges to be removed
$\mathcal{S} = \varnothing$;
$c = 0$;
$\mathcal{E}$ = list of edges in the graph;
if *greedy* **then**
  for *iter* $= 1 : K$ **do**
    Compute the centrality measure of interest $\forall (i, j) \in \mathcal{E}$;
    s = element in $\mathcal{E}$ with the smallest centrality;
    Downdate s from $\mathcal{G}$;
    $\mathcal{S} = \mathcal{S} \cup \{s\}$;
    $\mathcal{E} = \mathcal{E} \setminus \{s\}$;
  end
else
  Compute edge centrality measure of interest $\forall (i, j) \in \mathcal{E}$;
  Sort the edges in ascending order;
  $\mathcal{S}$ = top K elements in the sorted array;
end
Return $\mathcal{S}$.

edge corresponding to the minimum decrease of communicability.[2] Note that for large networks this method is too costly to be practical. For this reason we aim to develop inexpensive techniques that will hopefully give close-to-optimal results. Nevertheless, for small networks we will use the "optimal" approach (where we systematically try all feasible edges and delete the one causing the least drop in total communicability) as a baseline method against which we compare the various algorithms discussed below. This method will be henceforth referred to as `optimal`.

The next methods we introduce perform the downdate of the lowest ranked existing edge according to the edge centrality measures previously introduced whose removal does not disconnect the network. We will refer to these methods as `subgraph`, `nodeTC`, and `eigenvector`, which are based on definitions 2.4, 2.5, and 2.6, respectively. From the point of view of the communicability, these methods downdate an edge connecting two nodes which are peripheral (i.e., have low centrality) and therefore are not expected to give a large contribution to the spread of information along the network. Hence, the selected edge is connecting two nodes whose ability to exchange information is already very low, and we do not expect the total communicability to suffer too much from this edge removal. This observation also suggests that such downdates can be repeatedly applied without the need to recompute the ranking of the edges after each downdate. As long as the number of downdates performed remains small compared to the total

[2]Strictly speaking, this would correspond to a greedy algorithm which is only locally optimal. In general, this is unlikely to result in "globally optimal" network communicability. In the following, the term "optimal" will be understood in this limited sense only.

number of edges, we expect good results at a greatly reduced total cost. Note also that such downdates can be performed simultaneously rather than sequentially. We will refer to these variants as `subgraph.no`, `nodeTC.no`, and `eigenvector.no`.

Finally, we consider a technique motivated by the bounds obtained via quadrature rules derived in Section 1.3. From the expression for the function Φ in the special case of the downdate (cf. Corollary 2.2), we infer that a potentially good choice may be to remove the edge having incident nodes i, j for which the sum d_i+d_j is minimal, if its removal does not disconnect the network. Indeed, this choice reduces the upper bound only slightly and the total communicability may mirror this behavior. Another way to justify this strategy is to observe that it is indeed the optimal strategy if we approximate e^A with its second-order approximation $I + A + \frac{1}{2}A^2$ in the definition of total communicability. This technique will be henceforth referred to as `degree`. We note that a related measure, namely, the average of the out-degrees $\frac{d_i+d_j}{2}$, was proposed in [15] as a measure for the centrality of an edge (i, j) in directed graphs.

Algorithms 1 and 2 contain the pseudo-code for our downdating techniques with and without the connectivity check. Both these algorithms can be used with or without the recomputation of the rankings for the edges after each modification has been performed. They require as inputs the initial graph $\mathcal{G}$ (typically in the form of its adjacency matrix A), a budget K, i.e., the number of modifications one wants to perform, and a Boolean *greedy*, which indicates whether the rankings of the edges have to be recomputed after each modification ($greedy = 1$) or not ($greedy = 0$).

(P2) Update

Most real world networks are characterized by low average degree. As a consequence, the adjacency matrices of such networks are sparse ($m = \mathcal{O}(n)$). For the purpose of selecting a virtual edge to be updated, this implies that we have approximately $\frac{1}{2}\left(n^2 - cn\right)$ possible choices if we want to avoid the formation of multiple edges or self–loops (here c is a moderate constant). Each one of these possible updates will result in an increase of the total communicability of the network, but not every one of these will result in a significant increment.

Algorithm 3: Updating algorithm.

Data: Initial graph $\mathcal{G}$, $K \in \mathbb{N}$, $S \subset \mathcal{V}$ nodes in the subgraph, $greedy \in \{0, 1\}$
Result: Set $\mathcal{S}$ of K edges to be added
$\mathcal{S} = \varnothing$;
Ω = list of virtual edges in the subgraph containing nodes in S;
if *greedy* **then**
  for *iter = 1 : K* **do**
    Compute edge centrality measure of interest $\forall (i, j) \in \Omega$;
    s = element in Ω having the largest centrality;
    Update s in $\mathcal{G}$;
    $\mathcal{S} = \mathcal{S} \cup \{s\}$;
    $\Omega = \Omega \setminus \{s\}$;
  end
else
  Compute edge centrality measure of interest $\forall (i, j) \in \Omega$;
  Sort the edges in descending order;
  $\mathcal{S}$= top K elements in the sorted array;
end
Return $\mathcal{S}$.

One natural updating technique is to connect two nodes having high centralities, i.e., add the virtual edge having the highest ranking according to the corresponding edge centrality. Its incident nodes, being quite central, can be expected to have an important role in the spreading of information along the network; on the other hand, the communication between them may be relatively poor (think for example of the case where the two nodes sit in two distinct communities). Hence, giving them a preferential communication channel, such as an edge between them, should result in a better spread of information along the whole network. Again, we will use the labels `subgraph`, `nodeTC`, and `eigenvector` to describe these updating strategies. As before, in order to reduce the computational cost, we also test the effectiveness of these techniques without the recomputation of the ranking of the virtual edges after each update. These variants (referred to as `subgraph.no`, `nodeTC.no`, and `eigenvector.no`) are expected to return good results as well, since the selected update should not radically change the ranking of the edges. Indeed, they make central nodes even more central, and the ranking of the edges remains consequently almost unchanged. Note again that these updates can be performed simultaneously rather than sequentially.

As for the case of downdating, the bounds via quadrature rules derived in Section 1.3 suggest an updating technique, i.e., adding the virtual edge (i, j) for which $d_i + d_j$ is maximal. Indeed, such a choice would maximize the lower bound on the total communicability, see Corollary 2.3. Again, this choice can also be justified by noting that it is optimal if e^A is replaced by its quadratic Maclaurin approximant. We will again use the

TABLE 2.1: Brief description of the techniques used to tackle the downdating and updating problems.

Method	Downdate: $(i,j) \in \mathcal{E}$	Update: $(i,j) \notin \mathcal{E}$
`optimal`	$\arg\min\{TC(A) - TC(\widehat{A})\}$	$\arg\max\{TC(\widetilde{A}) - TC(A)\}$
`subgraph(.no)`	$\arg\min\{{}^{e}\mathcal{SC}(i,j)\}$	$\arg\max\{{}^{e}\mathcal{SC}(i,j)\}$
`eigenvector(.no)`	$\arg\min\{{}^{e}\mathcal{EC}(i,j)\}$	$\arg\max\{{}^{e}\mathcal{EC}(i,j)\}$
`nodeTC(.no)`	$\arg\min\{{}^{e}\mathcal{TC}(i,j)\}$	$\arg\max\{{}^{e}\mathcal{TC}(i,j)\}$
`degree`	$\arg\min\{d_i + d_j\}$	$\arg\max\{d_i + d_j\}$

label `degree` to refer to this updating strategy.

All these techniques will be compared with the `optimal` one, based on systematically trying all feasible virtual edges and selecting at each step the one resulting in the largest increase of the total communicability. Due to the very high cost of this brute force approach, we will use it only on small networks.

Algorithm 3 describes the pseudo-code for our updating techniques. It can be used with or without the recomputation of the rankings for the edges after each modification has been performed. The Boolean *greedy* is used to discriminate between these two options. The algorithm requires as inputs the original graph $\mathcal{G}$ (typically in the form of its adjacency matrix A), a budget K, i.e., the number of modifications one wants to perform, and the Boolean *greedy*.

The heuristics introduced to tackle (P1) and (P2) are summarized in Table 2.1.

(P3) Rewire

As we have already noted, there are situations in which the rewire of an edge may be preferable to the addition of a new one. There are various possible choices for the rewiring strategy to follow. The greatest part of those found in literature are variants of random rewiring (see for example [16, 65]). In this work, on the other hand, we are interested in devising mathematically informed rewiring strategies. For comparison purposes, however, we will compare our rewiring methods to the random rewire method, `random`, which downdates an edge (chosen uniformly at random among all edges whose removal does not disconnect the network) and then updates a virtual edge, also chosen uniformly at random.

Combining the various downdating and updating methods previously introduced we obtain different rewiring strategies based on the centralities of edges and on the bounds for

the total communicability. Concerning the methods based on the edge subgraph, eigenvector, and total communicability centrality, we note that since a single downdate does not dramatically change the communication capability of the network, we do not need to recompute the centralities and the ranking of the edges after each downdating step, at least as long as the number of rewired edges remains relatively small (numerical experiments not shown here support this claim). On the other hand, after each update we may or may not recalculate the edge centralities. As before, we use `subgraph/subgraph.no`, `eigenvector/eigenvector.no` and `nodeTC/nodeTC.no` to refer to these three variants of rewiring. Additionally, we introduce another rewiring strategy, henceforth referred to as `node`, based on the subgraph centrality of the nodes. In this method we disconnect the most central node from the least central node among its immediate neighbors; then we connect it to the most central node among those it is not linked to. It is worth emphasizing that this strategy is philosophically different from the previous ones based on the edge subgraph centrality in the downdating phase (the updating step is the same). In fact, in those methods we use information on the nodes in order to deduce some information on the edges connecting them; on the other hand, the `node` algorithm does not take into account the potentially high "payload" of the edges involved, whose removal may result in a dramatic drop in the total communicability.

2.3 Computational aspects

There are several important points to keep in mind when implementing the methods described in the previous subsection. First of all, for the downdates, updates, and rewires based on the edge subgraph centrality we need to compute the diagonal entries of e^A. This is the most expensive part of these methods. There are, however, techniques that can be used to rapidly estimate the diagonal entries of e^A and to quickly identify the top ℓ nodes, where $\ell \ll n$; see [10, 13, 44] and references therein. It should be pointed out that very high accuracy is not required or warranted by the problem. We also recall that the same techniques (based on quadrature rules and the Lanczos process) can be used to compute the total communicability $\mathbf{1}^T e^A \mathbf{1}$ quickly (typically in $\mathcal{O}(n)$ work), although such computation is actually not required by any of the algorithms tested here except by the `optimal` strategy, which is only used (for small networks) as a baseline method.

Such methods can also be used for rapidly estimating the node total communicability centralities, $TC(i) = [e^A \mathbf{1}]_i = \mathbf{e}_i^T e^A \mathbf{1}$.

Secondly, when performing an update or the updating phase of a rewire, it makes sense to work with a subset of the set of all virtual edges $\overline{\mathcal{E}}$. Indeed, for a sparse network $\overline{\mathcal{E}}$ contains $\mathcal{O}(n^2)$ edges and for large n this is prohibitive. Due to the particular selection criteria we want to use, it is reasonable to restrict ourselves to the virtual edges in the subgraph of our network that are incident to a subset S of nodes containing a certain percentage of the top nodes, ranked according to some centrality measure. We found that for the larger networks considered in this chapter, using just the top 1% of the nodes ranked using the eigenvector centrality yields very good results.

Next, we derive the computational costs for the downdating techniques used in this chapter. Let m be the number of edges in the network and let $K \ll n$, assumed bounded independently of n as $n \to \infty$, be the maximum number of downdates we want to perform; in this work, the maximum value of K we consider is 2000 (used for the three largest networks in our data set).

In the `optimal` method we remove each edge in turn, compute the total communicability after each downdate, and then choose the downdate which caused the least decrease in $TC(A)$; assuming that the cost of computing $TC(A)$ is $\mathcal{O}(n)$, we find a total cost of $\mathcal{O}(Kmn)$ for K updates. Since $m = \mathcal{O}(n)$, this amounts to $\mathcal{O}(Kn^2)$.

Next, we consider the cost of techniques based on subgraph centralities. The cost of computing the node subgraph centralities is not easy to assess in general, since it depends on network properties and on the approximation technique used. If a rank-k approximation is used [44], the cost is approximately $\mathcal{O}(kn)$; hence, the cost is linear in n if k is independent of n, which is appropriate for many types of networks. Computing the edge centralities requires another $m = \mathcal{O}(n)$ operations, and sorting the edges by their centralities costs approximately $m \ln m$ comparisons. Note that sorting is only necessary in the `subgraph.no` variant of the algorithm; indeed, with `subgraph` we recompute the centralities after each update and instead of sorting the result we only need to identify the edge of minimum centrality at each step, which can be done in $\mathcal{O}(m)$ work. Summarizing, the cost of `subgraph` is $\mathcal{O}(K(n+m)) = \mathcal{O}(Kn)$ for K downdates if we assume the subgraph centralities can be computed in $\mathcal{O}(n)$ time, and the cost for `subgraph.no` is $\mathcal{O}(n+m) = \mathcal{O}(n)$ plus a pre-processing cost of $\mathcal{O}(m \ln m)$ $(= \mathcal{O}(n \ln n))$ comparisons

for sorting the edge centralities. Although the asymptotic cost of `subgraph.no` appears to be higher than that of `subgraph` (due to the $n \ln n$ term), in practice one finds that `subgraph.no` runs invariably much faster than `subgraph` for all cases tested here.

The costs associated with `eigenvector` and `eigenvector.no` scale like those of `subgraph` and `subgraph.no`, assuming that the dominant eigenvector $\mathbf{q}_1$ of a large sparse $n \times n$ adjacency matrix can be approximated in $\mathcal{O}(n)$ time. For many real world networks this is a reasonable assumption, since in practice we found that running a fixed number of Lanczos steps will give a sufficiently good approximation of $\mathbf{q}_1$. The prefactors can be expected to be much smaller for the methods based on eigenvector centrality than for those based on subgraph centrality.

The costs for `nodeTC` and `nodeTC.no` are comparable to those for `eigenvector` and `eigenvector.no`, with the same asymptotic scalability.

Finally, the cost of `degree` is $\mathcal{O}(Km)$ and hence also $\mathcal{O}(Kn)$ for a sparse network.

Note that the cost of checking that the connectivity is preserved after each downdate does not affect these asymptotic estimates; indeed, using A^* search [54] this can be done in $\mathcal{O}(m)$ time and hence the additional cost is only $\mathcal{O}(n)$ for a sparse network. Of course, if the removal of an edge is found to disconnect the network we do not perform the downdate and move on to the next candidate edge.

We consider next the computational cost for the updating strategies. As before we let K, assumed bounded independently of n as $n \to \infty$, be the maximum number of updates we want to perform.

It can be easily shown that the `optimal` method costs $\mathcal{O}(Kn^3)$ operations. To estimate the cost of the remaining methods, we assume that the set $S \subset \mathcal{V}$ consisting of the top $\ell = |S|$ nodes (ranked according to some centrality measure) is known. The cost of determining this set is asymptotically dominated by the term $\mathcal{O}(n \ln n)$, as we saw. As already mentioned, ℓ will be equal to some fixed percentage of the total number of nodes in the network.

Both `subgraph` and `eigenvector` cost $\mathcal{O}(K\ell n)$ operations, provided a low rank approximation (of fixed rank) is used to estimate the subgraph centralities. The same holds for `nodeTC`. Typically, the prefactor will be larger for the former method. Since we assumed that $\ell = \mathcal{O}(n)$ (albeit with a very small prefactor, like 10^{-2}) these methods exhibit an

TABLE 2.2: Description of the Data Set.

NAME	n	m	λ_1	λ_2	$\lambda_1 - \lambda_2$
Zachary	34	78	6.726	4.977	1.749
Sawmill	36	62	4.972	3.271	1.701
social3	32	80	5.971	3.810	2.161
dolphins	62	159	7.193	5.936	1.257
Minnesota	2640	3302	3.2324	3.2319	0.0005
USAir97	332	2126	41.233	17.308	23.925
as-735	6474	12572	46.893	27.823	19.070
Erdös02	5534	8472	25.842	12.330	13.512
ca-HepTh	8638	24806	31.034	23.004	8.031
as-22july2006	22963	48436	71.613	53.166	18.447
usroad-48	126146	161950	3.911	3.840	0.071

$\mathcal{O}(n^2)$ scaling. In practice this is somewhat misleading, since the quadratic scaling is not observed until n is quite large.

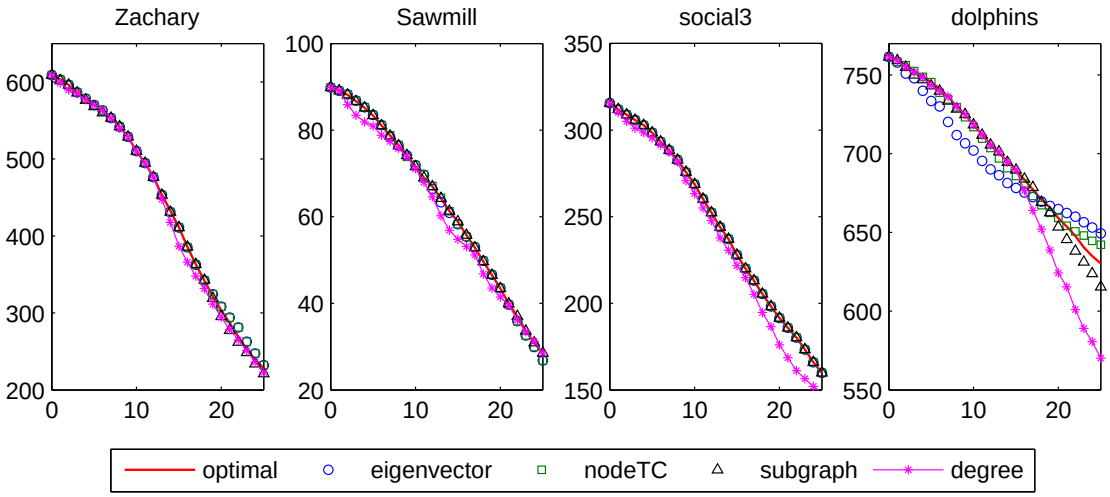
Finally, `subgraph.no`, `eigenvector.no`, and `nodeTC.no` all cost $\mathcal{O}((K + \ell)n)$ while `degree` costs $\mathcal{O}(K\ell) = \mathcal{O}(n)$. Again, the former cost is asymptotically quadratic but the actual cost is dominated by the linear part until n becomes quite large. We note that we can obtain an asymptotically linear scaling by imposing an upper bound on ℓ, i.e., on the fraction of nodes that we are willing to include in the working subset S of nodes. We stress that because of the widely different prefactors for the various methods, these asymptotic estimates should only be taken as indicative. In the next section we present timings showing the linear scaling behavior of the various heuristics in practice, at least for the networks considered here.

2.4 Numerical results

In this section we discuss the results of numerical studies performed in order to assess the effectiveness and efficiency of the proposed techniques. The tests have been performed on both synthetic and real-world networks, as described below.

All experiments were performed using Matlab Version 7.12.0.635 (R2011a) on an IBM ThinkPad running Ubuntu 12.04.5 LTS, a 2.5 GHZ Intel Core i5 processor, and 3.7 GiB of RAM.

FIGURE 2.1: Evolution of the normalized total communicability vs number of downdates performed on small networks.



2.4.1 Real-world networks

The real-world networks used in the tests (see Table 2.2) come from a variety of sources and their description can be found in Appendix A. For each network, Table 2.2 reports the number of nodes (n), the number of edges (m), the two largest eigenvalues, and the spectral gap. We use the first eight networks to test all methods described in the previous section (except for `optimal`, which is only applied to the four smallest networks) and the last three to illustrate the performance of the most efficient among the methods tested.

We begin by showing results for the four smallest networks. Figure 2.1 displays the results obtained with the downdating methods `optimal`, `eigenvector`, `nodeTC`, `subgraph`, and `degree`. The results for `eigenvector.no`, `subgraph.no`, and `nodeTC.no` are virtually indistinguishable from those obtained with `eigenvector`, `subgraph`, and `nodeTC` and are therefore not shown. At each step we modify the network by downdating an edge and we then compute and plot the new value of the normalized total communicability. The tests consist of $K = 25$ modifications.

Figure 2.1 shows that our methods all perform similarly and give results that are in most cases very close to those obtained with `optimal`, and occasionally even better, as is the case for `eigenvector(.no)` and `nodeTC(.no)` on the dolphins network after a sufficient number of downdates have been performed. This result may seem puzzling at first, however, it can be easily explained by noticing that `eigenvector` and `nodeTC` select a different edge from that selected by `optimal` at the third and sixth downdate step,

FIGURE 2.2: Evolution of the normalized total communicability vs number of updates performed on small networks.

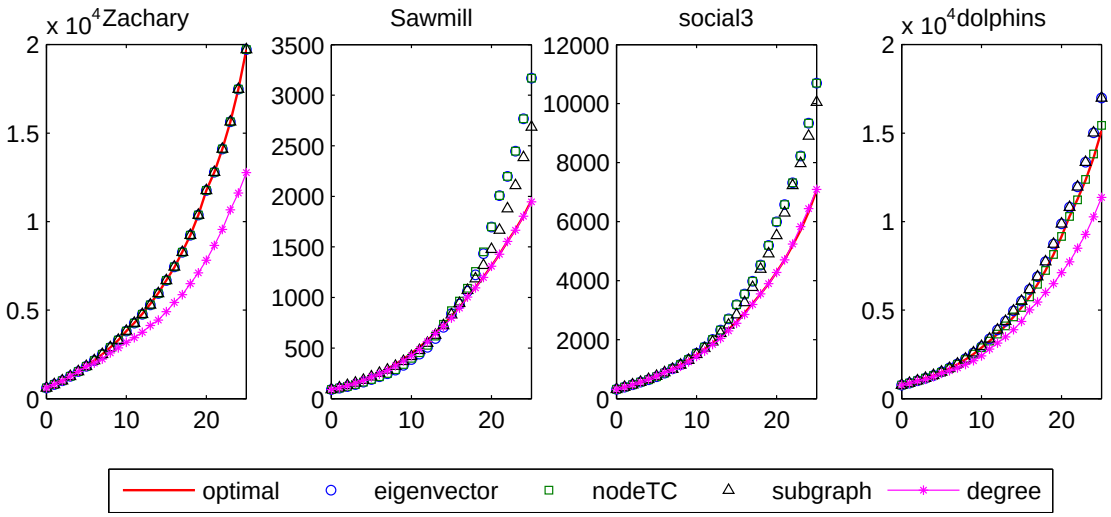
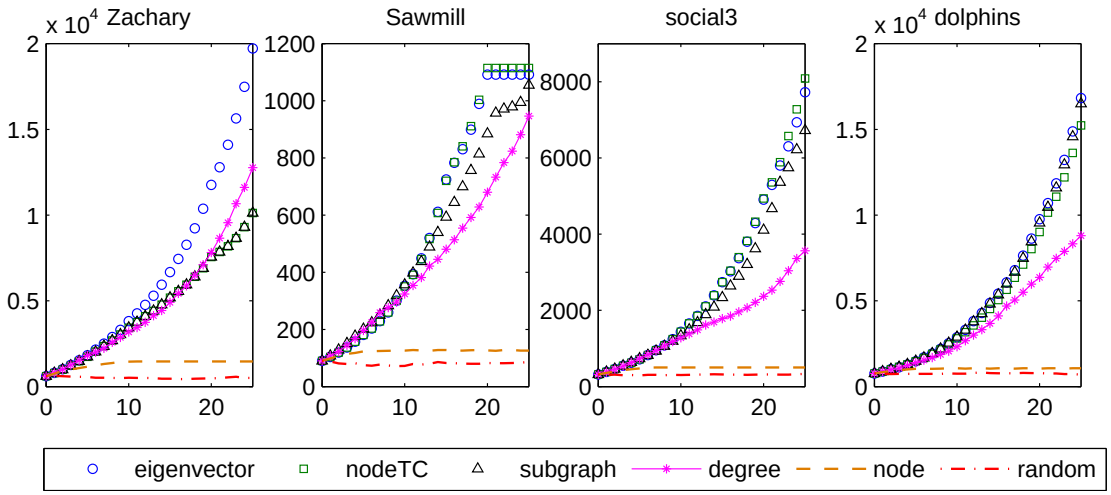


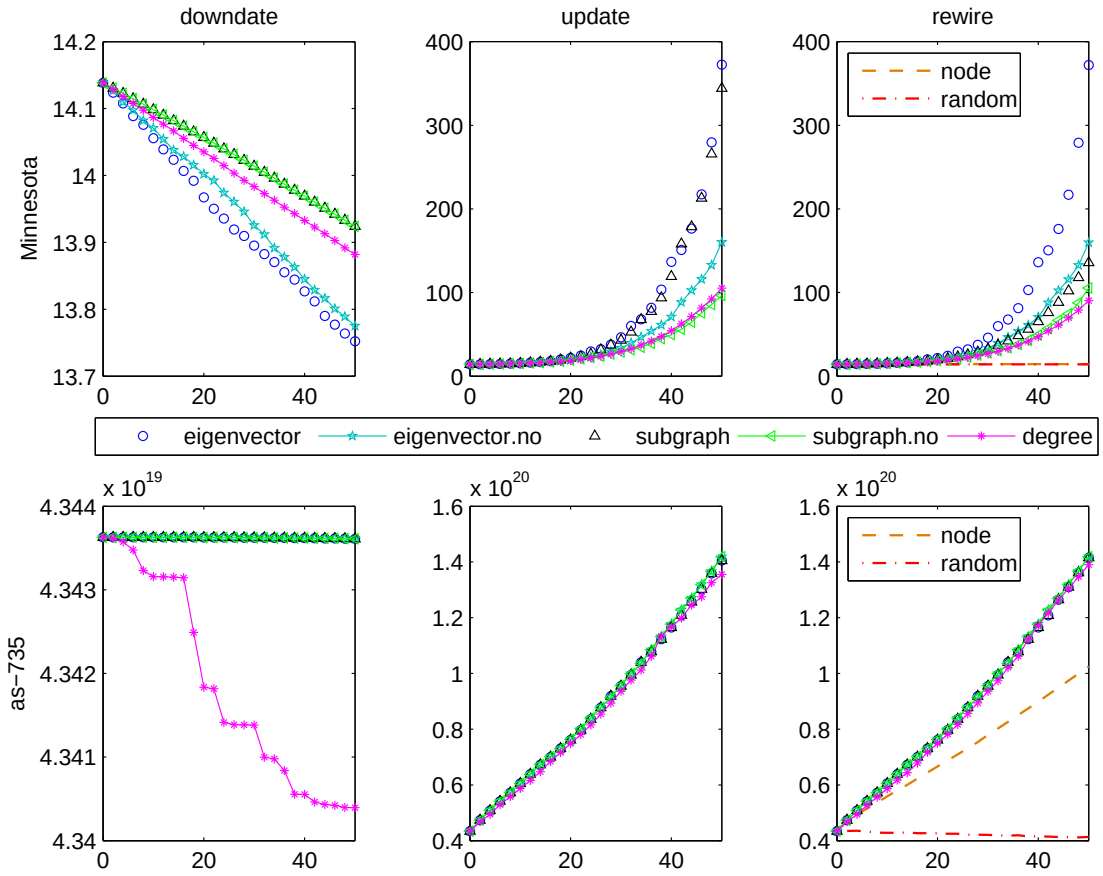
FIGURE 2.3: Evolution of the normalized total communicability vs number of rewires performed on small networks.



respectively. Hence, from that point on the adjacency matrices on which the methods work are different, and the choice performed by the `optimal` method may no longer be optimal for the graphs manipulated by `eigenvector` and `nodeTC`. Note that even the simple heuristic `degree` seems to perform well, except perhaps on the dolphins network after 15 or so downdate steps. Overall, the methods based on eigenvector and total communicability centrality appear to perform best in view of their efficacy and low cost.

The results for the updating methods are reported in Figure 2.2. As for the downdating methods, `subgraph.no`, `eigenvector.no`, and `nodeTC.no` return results that are virtually identical to those obtained using `subgraph`, `eigenvector`, and `nodeTC`, therefore we

FIGURE 2.4: Evolution of the normalized total communicability vs number of downdates, updates and rewires for networks Minnesota and as735.

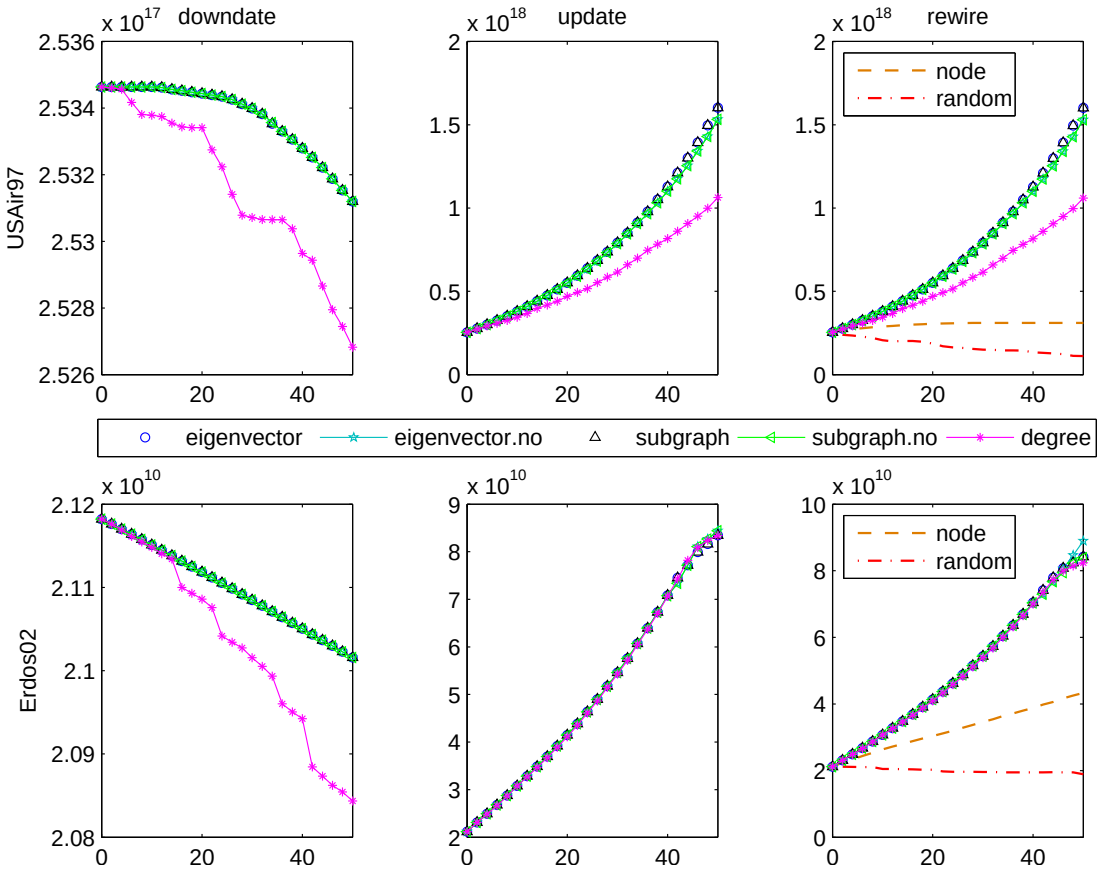


omit them from the figure. Once again we see that the methods based on eigenvector, subgraph, and total communicability centrality give excellent results, whereas `degree` is generally not as effective.

Rewiring results are displayed in Figure 2.3. Clearly, the methods making use of edge centrality perform quite well, in contrast to `random` rewiring (which is only included as a base for comparison). Note also the poor performance of `node`, showing that the use of edge centralities (as opposed to node centralities alone) is indispensable in this context.

Next, we consider the medium size networks (Minnesota, as735, USAir97, and Erdös02). For these networks the set $\overline{\mathcal{E}}$ (the complement of the set $\mathcal{E}$ of edges) is large enough that performing an extensive search for the edge to be updated is expensive. Hence, we form the set S of the top 10% of the nodes ordered according to the eigenvector centrality and we restrict our search to virtual edges incident to these nodes only. An exception is the network USAir97 where we have used the set S corresponding to

FIGURE 2.5: Evolution of the normalized total communicability vs number of downdates, updates and rewires for networks USAir97 and Erdös02.

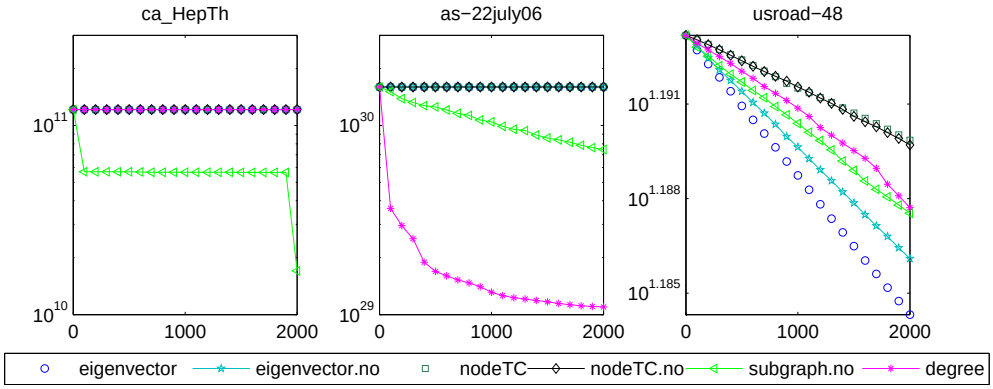


the top 20% of the nodes, since in the case of 10% this set contained only 52 virtual edges. In Figures 2.4 and 2.5 we show results for the methods `eigenvector`, `eigenvector.no`, `subgraph`, `subgraph.no` and `degree`. These results confirm the effectiveness of the `eigenvector` and `subgraph` algorithms and their less expensive variants `eigenvector.no` and `subgraph.no` in nearly all cases; similar results were obtained with `nodeTC` and `nodeTC.no` (not shown). The only exception is in the downdating of the Minnesota network, where the eigenvector-based techniques give slightly worse results. This fact is easily explained in view of the tiny spectral gap characterizing this and similar networks[3] (see Table 2.2). Because of this property, eigenvector centrality is a poor approximation of subgraph centrality and cannot be expected to give results similar to those obtained with `subgraph` and `subgraph.no`.

The results also show that the inexpensive `degree` method does not perform as well on these networks, except perhaps on Minnesota. The relatively poor performance of this

[3]Small spectral gaps are typical of large, grid-like networks such as the road networks or the graphs corresponding to triangulation or discretization of physical domains.

FIGURE 2.6: Downdates for large networks: normalized total communicability vs. number of modifications.



method is due to the fact that the information used by this method to select an edge for downdating is too local.

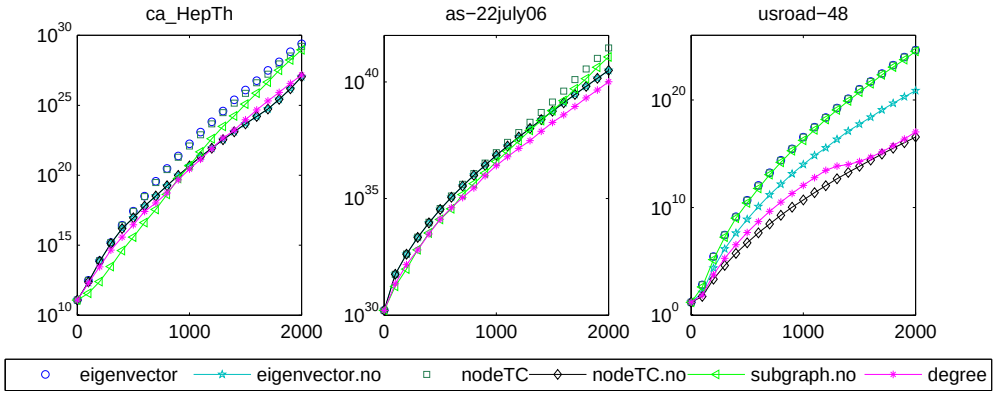
Note, however, the scale on the vertical axis in Figures 2.4 and 2.5, suggesting that for these networks (excluding perhaps Minnesota) all the edge centrality-based methods perform well with only very small relative differences between the resulting total communicabilities.

Overall, these results indicate that the edge centrality-based methods, especially the inexpensive `eigenvector.no` and `nodeTC.no` variants, are an excellent choice in almost all cases. In the case of downdating networks with small spectral gaps, `subgraph.no` may be preferable but at a higher cost.

The behavior of the `degree` method depends strongly on the network on which it is used. Our tests indicate that it behaves well in some cases (for example, Erdös02) but poorly in others (Minnesota). We speculate that this method may perform adequately on scale-free networks (such as Erdös02) where a high degree is an indication of centrality in spreading information across the network.

Some comments on the difference in the results for updating as compared to those for rewiring (downdating followed by updating) are in order. Recall that our downdating strategies aim to reduce as little as possible the decrease in the value of the total communicability, whereas the updating techniques aim to increase this index as much as possible. With this in mind, it is not surprising to see that the trends of the evolution of the total communicability after rewiring reflect those obtained with the updating strategies. The values obtained using the updates are in general higher than those obtained

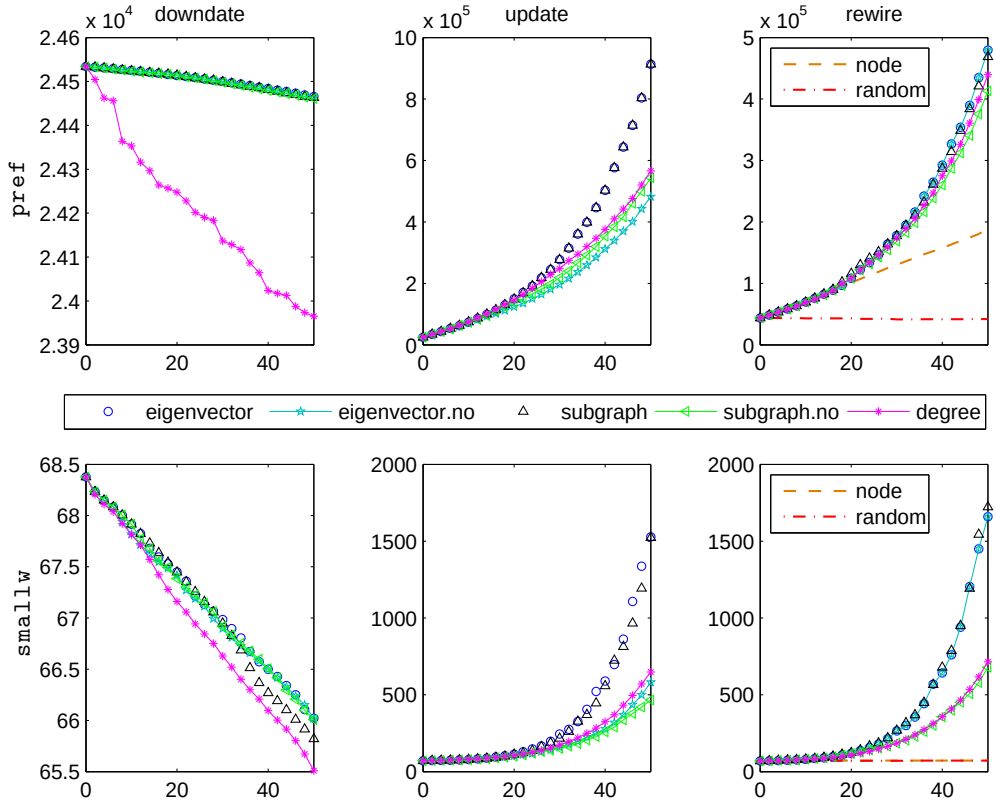
FIGURE 2.7: Updates for large networks: normalized total communicability vs. number of modifications.



using the rewiring strategies, since updating implies the addition of edges whereas in rewiring the number of edges remains the same. The difference is especially pronounced for the small networks (except for dolphins), where the effects of downdates has a greater impact, leading to a decrease of up nearly 70% of the original value of the total communicability after $K = 25$ downdates (cf. Figure 2.1). It is important to stress that the methods based on the edge eigenvector and total communicability centrality appear to be more stable than the others under rewiring and to dampen the effect of the downdates even for small networks.

Finally, in Figures 2.6 and 2.7 we show results for the three largest networks in our data set (ca-HepTh, as-22july06, and usroad-48). In the case of the updating, we have selected the virtual edges among those in the subgraph containing the top 1% of nodes ranked according to the eigenvector centrality. We compare the following methods: `eigenvector`, `eigenvector.no`, `nodeTC`, `nodeTC.no`, `subgraph.no`, and `degree`; random downdating was also tested and found to give poor results. Note that network usroad-48 behaves similarly to Minnesota; this is not surprising in view of the fact that these are both road networks with a tiny spectral gap. Looking at the scale on the vertical axis, however, it is clear that the decrease in total communicability is negligible with all the methods tested here. The results on these networks confirm the general trend observed so far; in particular, we note the excellent behavior of `nodeTC` and `nodeTC.no`.

FIGURE 2.8: Evolution of the total communicability when 50 downdates, updates or rewires are performed on two synthetic networks with $n = 1000$ nodes.



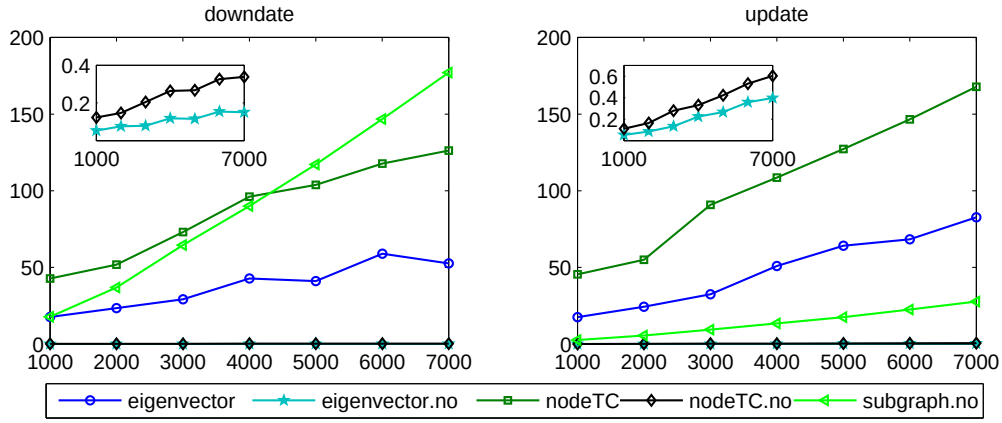
2.4.2 Synthetic networks

The synthetic examples used in the tests were produced using the CONTEST toolbox for Matlab (see [87, 88]). We tested two types of graphs: the preferential attachment (Barabási–Albert) model and the small world (Watts–Strogatz) model, see Section 1.4.

We have used matrices with $n = 1000$ nodes which were built using the default values for the functions `pref` and `smallw` previously described in Section 1.4. We used $d = 2$ in the Barabási–Albert model and $k = 2$, $p = 0.1$ in the Watts–Strogatz model.

The results for our tests are presented in Figure 2.8. These results agree with what we have seen previously on real-world networks. Interestingly, `degree` does not perform well for the downdate when working on the preferential attachment model; this behavior reflects what we have seen for the networks USAir97, as-735, and Erdös02, which are indeed scale-free networks.

FIGURE 2.9: Timings in seconds for scale-free graphs of increasing size (500 modifications).



2.4.3 Timings for synthetic networks

We have performed some experiments with synthetic networks of increasing size in order to assess the scalability of the various methods introduced in this chapter. A sequence of seven adjacency matrices corresponding to Barabási–Albert scale-free graphs was generated using the CONTEST toolbox. The order of the matrices ranges from 1000 to 7000; the average degree is kept constant at 5. A fixed number of modifications ($K = 500$) was carried out on each network.

We used the built-in Matlab function `eigs` (with the default settings) to approximate the dominant eigenvector of the adjacency matrix A, the Matlab toolbox `mmq` [72] to estimate the diagonal entries of e^A (with a fixed number of five nodes in the Gauss–Radau quadrature rule, hence five Lanczos steps per estimate), and the toolbox `funm_kryl` to compute the vector $e^A\mathbf{1}$ of total communicabilities, also with the default parameter settings.

The results are shown in Figure 2.9. The approximate (asymptotic) linear scaling behavior of the various methods (in particular of `nodeTC.no` and `eigenvector.no`, which are by far the fastest, see the insets) is clearly displayed in these plots.

2.4.4 Timings for larger networks

In Tables 2.3-2.4 we report the timings for various methods when $K = 2000$ downdates and updates are selected for the three largest networks listed in Table 2.2.

TABLE 2.3: Timings in seconds for $K = 2000$ downdates performed on the three largest networks in Table 2.2.

	ca–HepTh	as–22july06	usroad–48
`eigenvector`	278.13	599.83	11207.39
`eigenvector.no`	0.07	1.79	4.08
`nodeTC`	553.04	1234.49	2634.27
`nodeTC.no`	0.34	0.83	1.34
`subgraph.no`	107.36	383.34	1774.07
`subgraph.no`*	246.44	808.24	10322.41
`degree`	29.67	53.42	153.52

TABLE 2.4: Timings in seconds for $K = 2000$ updates performed on the three largest networks in Table 2.2

	ca–HepTh	as–22july06	usroad–48
`eigenvector`	192.8	436.9	1599.5
`eigenvector.no`	0.19	0.33	5.85
`nodeTC`	561.9	1218.8	2932.
`nodeTC.no`	0.30	0.55	1.59
`subgraph.no`	3.13	7.20	121.4
`degree`	11.1	12.4	175.8

The timings presented refer to the selection of the edges to be downdated or updated, which dominates the computational effort. For the method `subgraph.no` in the case of downdates, we restricted the search of candidate edges to a subset of $\mathcal{E}$ in order to reduce costs. For the three test networks we used 40%, 45%, and 15% of the nodes, respectively, chosen by taking those with lowest eigenvector centrality, and the corresponding edges. We found the results to be very close to those obtained working with the complete set $\mathcal{E}$, but at a significantly lower cost (especially for the largest network). The timings when the complete set $\mathcal{E}$ was considered correspond to the label `subgraph.no`* in Table 2.3.

These results clearly show that algorithms `nodeTC.no` and `eigenvector.no` are orders of magnitude faster than the other methods; method `subgraph.no`, while significantly more expensive, is still reasonably efficient and can be expected to give better results in some cases (e.g., on networks with a very small spectral gap). It is worth mentioning that in principle it is possible to greatly reduce the cost of this method using parallel processing, since each subgraph centrality can be computed independently of the others. The `degree` algorithm, on the other hand, cannot be recommended in general since it gives somewhat inferior results. The remaining methods `eigenvector`, `nodeTC`, and `subgraph` (not shown here) are prohibitively expensive for large networks, at least when the number K of modifications is high (as it is here).

We also observe that downdating is generally a more expensive process than updating,

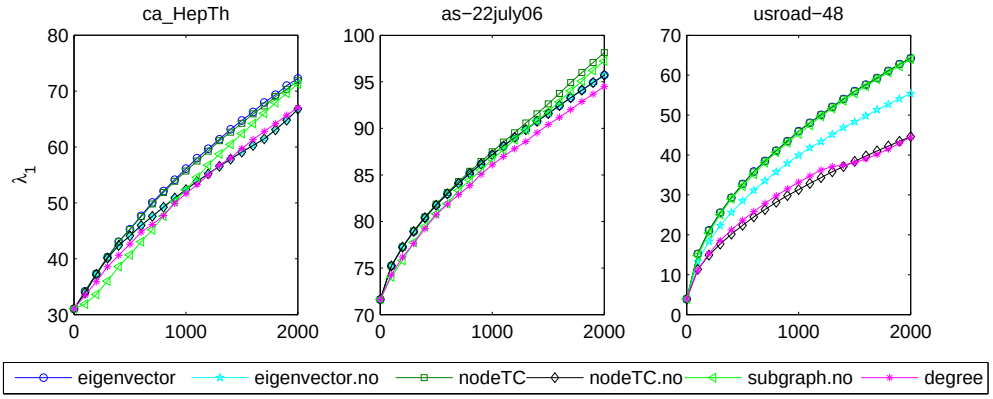
since in the latter case the edges are to be chosen among a fairly small subset of all virtual edges, whereas in the downdating process we work on the whole set $\mathcal{E}$ of existing edges (or on a large subset of $\mathcal{E}$). For some methods the difference in cost becomes significant when the networks are sufficiently large and the number of modifications to be performed is high.

Summarizing, the method labelled `nodeTC.no` is the fastest and gives excellent results, quite close to those of the more expensive methods, and therefore we can recommend its use for the type of problems considered here. The methods labelled `eigenvector.no` and `subgraph.no` are also effective and may prove useful in some settings, especially for updating.

2.5 Evolution of other connectivity measures

In this section we want to highlight another facet of the methods we have introduced for (approximately) optimizing the total communicability. In particular, we look at the evolution of other network properties under our updating strategies. When building or modifying a network, there are various features that one may want to achieve. Typically, there are two main desirable properties: first, the network should do a good job at spreading information, i.e., have a high total communicability; second, the network should be robust under targeted attacks or random failure, which is equivalent to the requirement that it should be difficult to "isolate" parts of the network, i.e., the network should be "well connected". This latter property can be measured by means of various indices. One such measure is the spectral gap $\lambda_1 - \lambda_2$. As a consequence of the Perron–Frobenius Theorem, adding an edge to a connected network causes the dominant eigenvalue λ_1 of A to increase (cf. Corollary 1.12). Figure 2.10 displays the evolution of the leading eigenvalue of the adjacency matrix as we add links to the three largest networks in our dataset. These results show that when a network is updated using one of our techniques, the first eigenvalue increases rapidly with the number of updates. On the other hand, the second eigenvalue λ_2 tends to change little with each update and it may even decrease (recall that the matrix $\mathbf{e}_i\mathbf{e}_j^T + \mathbf{e}_j\mathbf{e}_i^T$ being added to A in an update is indefinite). Therefore, the spectral gap $\lambda_1 - \lambda_2$ widens rapidly with the number of

FIGURE 2.10: Evolution of the leading eigenvalue when $K = 2000$ updates are performed on the three largest networks in Table 2.2.



updates.[4] It has been suggested by some authors (see, e.g., [33, 82]) that a large spectral gap is typical of complex networks with good expansion properties.

Here we focus on a related measure, the natural connectivity. In particular, we investigate the effect of our proposed methods of network updating on the evolution of this index.

2.5.1 Tracking the free energy (or natural connectivity)

In [100] the authors introduced a measure of network connectivity which is based on an intuitive notion of robustness and whose analytical expression has a clear meaning and can be derived from the eigenvalues of A; see also [99]. The idea underlying this index is that a network is more robust if there exists more than one route to get from one node to another; this property ensures that if a route become unusable, there is an alternative way to get from the source of information to the target. Therefore, intuitively a network is more robust if it has a lot of (apparently) redundant routes connecting its vertices or, equivalently, if each of its nodes is involved in a lot of closed walks. The natural connectivity aims to measure this property by taking advantage of the fact that a measure for the total number of closed walks in a network already exists: the Estrada index (see Subsection 1.4.4). Normalizing its value and taking the natural logarithm,

[4] This fact, incidentally, may serve as further justification for the effectiveness of algorithms like `nodeTC.no` and `eigenvector.no`.

one obtains the *natural connectivity* (or *natural eigenvalue*) of the graph,

$$\overline{\lambda}(A) = \ln\left(\frac{EE(G)}{n}\right) = \ln\left(\frac{1}{n}\sum_{j=1}^{n} e^{\lambda_j}\right),$$

which can be seen as an "average" eigenvalue and changes monotonically when an edge is downdated or updated in the graph (see [100]). Coarse bounds on this index are readily obtained:

$$0 \leq \overline{\lambda}(A) \leq \ln((n-1)e^{-1} + e^{n-1}) - \ln n.$$

The lower bound is attained by the empty graph, while the upper bound is attained by the complete graph, as a straightforward computation shows. Using the results in [10] we obtain more refined bounds via quadrature rules:

$$\ln\left(\frac{1}{n}\sum_{i=1}^{n}\frac{\beta^2 e^{\frac{d_i}{\beta}} + d_i e^{-\beta}}{\beta^2 + d_i}\right) \leq \overline{\lambda}(A) \leq \ln\left(\frac{1}{n}\sum_{i=1}^{n}\frac{\alpha^2 e^{\frac{d_i}{\alpha}} + d_i e^{-\alpha}}{\alpha^2 + d_i}\right),$$

where $[\alpha, \beta]$ is an interval containing the spectrum of $-A$ and d_i is the degree of node i.

It turns out, however, that essentially the same index was already present in the literature. Indeed, the natural connectivity is only one of the possible interpretations that can be given to the logarithm of the (normalized) Estrada index. Another, earlier interpretation was given in [38], where the authors related this quantity to the Helmholtz free energy of the network $F = -\ln(EE(G))$.

Let us briefly recall here their approach. Consider a network in which every edge is weighted by a parameter $\beta > 0$ and consider its adjacency matrix βA. The eigenvalues of this new matrix are $\beta\lambda_j$ for all $j = 1, 2, \ldots, n$ and its Estrada index becomes $EE(G, \beta) = \mathrm{Tr}(e^{\beta A})$, where Tr denotes the trace. This index can be interpreted as the *partition function* of the corresponding complex network:

$$Z(G, \beta) := EE(G, \beta) = \mathrm{Tr}(e^{\beta A}).$$

Form the standpoint of quantum statistical mechanics, $\mathscr{H} = -A$ is the system Hamiltonian and $\beta = \frac{1}{k_B T}$ is the inverse temperature, with k_B the Boltzmann constant and T the absolute temperature. It is well known [34, 40] that β can be understood as a measure of the "strength" of the interactions between pairs of vertices; the higher the

temperature (i.e., the lower the value of β), the weaker the interactions. The eigenvalues λ_i (for $i = 1, \ldots, n$) give the possible energy levels, each corresponding to a different state of the system.

The probability that the system is found in a particular state can be obtained by considering the Maxwell–Boltzmann distribution:

$$p_i = \frac{e^{\beta\lambda_i}}{EE(G,\beta)}, \quad i = 1, \ldots, n.$$

Using this notation and the fact that the Estrada index can be seen as the partition function of the system, in [38] the authors define the *Gibbs entropy* of the network as:

$$S(G,\beta) = -k_B \sum_{i=1}^{n} p_i \ln(p_i) = -k_B \beta \sum_{i=1}^{n} (\lambda_i p_i) + k_B \ln(EE(G,\beta))$$

where in the last equality we have used the fact that $\sum_i p_i = 1$.

Using now the standard relation $F = H - TS$ that relates the *Helmholtz free energy* F, the *total energy* of the network H, the Gibbs entropy S, and the absolute temperature of the system T, the authors derive:

$$\begin{cases} H(G,\beta) = \sum_{i=1}^{n} \lambda_i p_i, \\ F(G,\beta) = -\beta^{-1} \ln(EE(G,\beta)). \end{cases}$$

It is then clear that if we set $\beta = 1$ and let $F := F(G,1)$, then

$$\overline{\lambda} = \ln(EE(G)) - \ln(n) = -F - \ln(n).$$

Therefore, the behavior of F is completely described by that of $\overline{\lambda}$ (and conversely) as the graph is modified by adding or removing links.

The natural connectivity has been recently used (see [23]) to derive manipulation algorithms that directly optimize this robustness measure. In particular, the updating algorithm introduced in [23] appears to be superior to existing heuristics, such as those proposed in [16, 45, 85]. This algorithm, which costs $\mathcal{O}(mt + Kd_{max}^2 t + Knt^2)$ where $d_{max} = \max_{i \in V} d_i$ and t is the (user-defined) number of leading eigenpairs, selects K edges to be added to the network as described in Algorithm 4.

Algorithm 4: Updating algorithm from [23].

Data: A adjacency matrix and $K \in \mathbb{N}$
Result: Set S of K edges to be added
$S = \varnothing$;
Compute the top t eigenpairs $(\lambda_k, \mathbf{q}_k)$ of A;
for *iter = 1 : K* **do**
  Compute $d_{\max} = \max(d_i)$, the largest degree of A ;
  Find the set C of $d_{\max}$ nodes with the highest eigenvector centrality;
  Select the edge $(i^*, j^*) \in \overline{\mathcal{E}}$ that maximizes

$$e^{\lambda_1}\left(e^{2\mathbf{q}_1(i)\mathbf{q}_1(j)} + \sum_{h=2}^{t} e^{\lambda_h - \lambda_1} e^{2\mathbf{q}_h(i)\mathbf{q}_h(j)}\right)$$

  and such that $i^*, j^* \in C$, $i^* \neq j^*$;
  $S = S \cup \{(i^*, j^*)\}$, $\mathcal{E} = \mathcal{E} \cup \{(i^*, j^*)\}$;
  Update A;
  Update the top t eigenpairs as

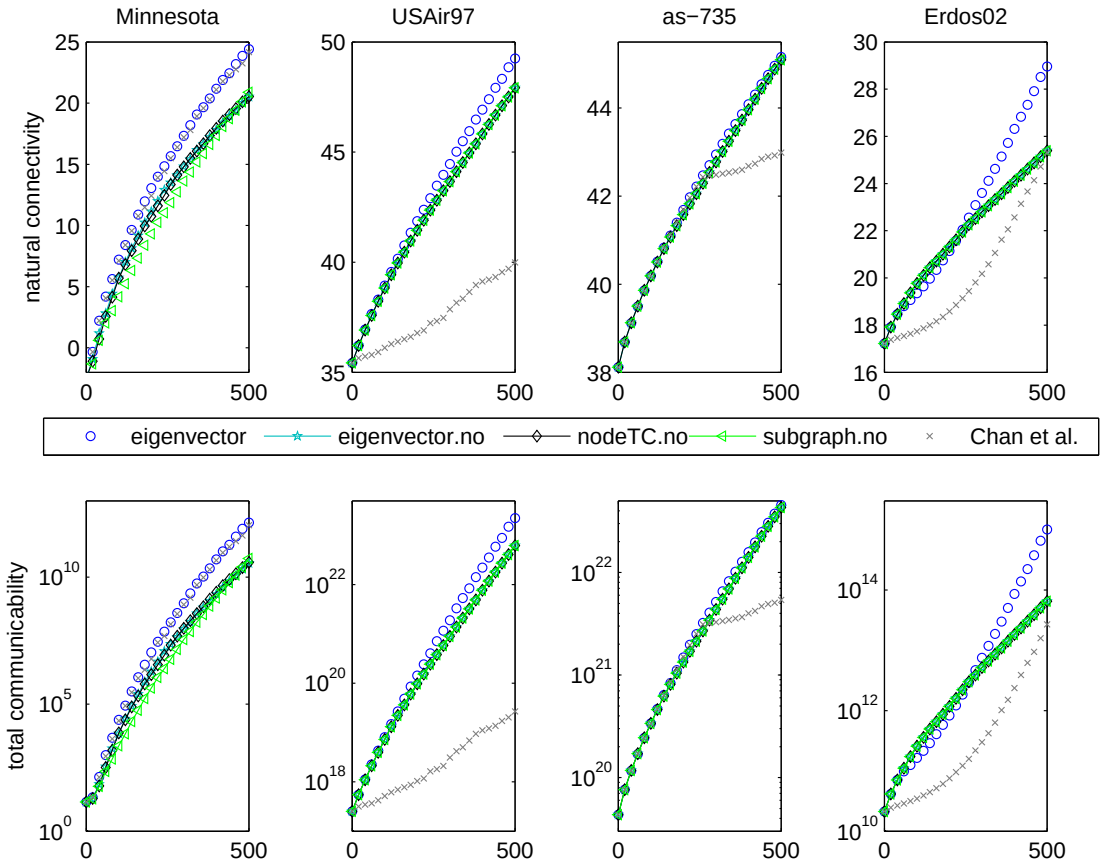
$$\begin{cases} \lambda_k = \lambda_k + 2\mathbf{q}_k(i)\mathbf{q}_k(j); \\ \mathbf{q}_k = \mathbf{q}_k + \sum_{h \neq k}\left(\frac{\mathbf{q}_h(i)\mathbf{q}_k(j) - \mathbf{q}_k(i)\mathbf{q}_h(j)}{\lambda_k - \lambda_h}\mathbf{q}_h\right) \end{cases} \qquad k = 1, 2, \ldots, t;$$

end
Return S.

We have compared our updating techniques with that described in Algorithm 4. Results for four representative networks are shown in Figure 2.11. In our tests, we used the value $t = 50$ (as in [23]), and we select $K = 500$ edges. Note that, when K is large, the authors recommend to recompute the set of t leading eigenpairs every l iterations. This operation requires an additional effort that our faster methods do not need. Since the authors in [23] show numerical experiments in which the methods with and without the recomputation return almost exactly the same results, we did not recompute the eigenpairs after any of the updates.

Figure 2.11 displays the results for both the evolution of the natural connectivity and of the normalized total communicability, where the latter is plotted in a semi-logarithmic scale. A total of 500 updates have been performed. The method labelled `Chan` selects the edges according to Algorithm 4 choosing from all the virtual edges of the graph. For our methods we used, as before, the virtual edges in the subgraph obtained selecting the top 10% or 20% of nodes ranked according to the eigenvector centrality. As one can easily see, our methods generally outperform the algorithm proposed in [23]. In particular, `nodeTC.no` and `eigenvector.no` give generally better results than `Chan` and are much faster in practice. For instance, the execution time with `Chan` on the network ca-HepTh was over 531 seconds, and much higher for the two larger networks. We recall

FIGURE 2.11: Evolution of the natural connectivity and of the normalized total communicability (in a semi–logarithmic scale plot) when up to 500 updates are performed on four real-world networks.



(see Table 2.4) that the execution times for `nodeTC.no` and `eigenvector.no` are about three orders of magnitude smaller.

It is striking to see how closely the evolution of the natural connectivity mirrors the behavior of the normalized total communicability. This is likely due to the fact that both indices depend on the eigenvalues of A (with a large contribution coming from the terms containing λ_1), and all the updating strategies used here tend to make λ_1 appreciably larger.

These findings indicate that the (normalized) total communicability is equally effective an index as the natural connectivity for the purpose of characterizing network connectivity. Since the network total communicability can be computed very fast (in $\mathcal{O}(n)$ time), we believe that the normalized total communicability should be used instead of the natural connectivity, especially for large networks. Indeed, computing the natural connectivity requires evaluating all the diagonal entries of e^A and is therefore significantly

more expensive, for large networks, than the total communicability.

2.6 The case of the resolvent

In the previous sections we have focused on the case of the total communicability defined in terms of the matrix exponential. However, as we have seen in Subsection 1.4.4, in principle one could use any other function $f : \mathbb{R} \longrightarrow \mathbb{R}$ in the definition, provided that it is defined on the spectrum of the adjacency matrix A. In this section we focus on the case of the resolvent:

$$(I - \alpha A)^{-1} = \sum_{k=0}^{\infty} (\alpha A)^k , \qquad \alpha \in \left(0, \frac{1}{\lambda_1}\right). \tag{2.2}$$

We can apply the previously introduced strategies to tune the resolvent-based total communicability defined as:

$$TC(A, f_\alpha) = \mathbf{1}^T (I - \alpha A)^{-1} \mathbf{1}, \tag{2.3}$$

where $f_\alpha(t) = (1-\alpha t)^{-1}$ and $\alpha \in (0, \lambda_1^{-1})$. The quantity we want to tune is now parameter dependent; moreover, for the matrix resolvent to be well defined, the parameter must vary in an interval which depends on the leading eigenvalue of the adjacency matrix.

To deal with the downdating, updating, and rewiring problem we then have to first set our framework, deciding whether we want to keep the parameter fixed or let it vary as we modify the edges in the network. In the following, we briefly discuss these two options.

Since (P3) requires to perform modifications of the form downdate-then-update, we will only discuss (P1) and (P2).

2.6.1 Fixed α

Let us first consider the case of $\alpha \in \left(0, \lambda_1^{-1}\right)$ fixed. This approach appears to be the most "natural", since we want to select modifications for tuning the resolvent-based total communicability of the original network and this index is associated to the original matrix via α.

As we modify the network by removing or adding links, the leading eigenvalue of the adjacency matrix evolves as well. From Corollary 1.12, we know that the reciprocal of the leading eigenvalue λ_1^{-1} cannot increase as we add links, and it cannot decrease as we remove edges. This means that the interval $\left(0, \lambda_1^{-1}\right)$ will become wider as we remove links from the network. Therefore, the originally chosen parameter α will remain in the interval when we deal with the downdating problem, not affecting the well-posedness of the Neumann series (2.3).

On the other hand, the addition of links to the network will make the interval more and more narrow; eventually, α will fall out of the interval. In such case, the global index $TC(A, f_\alpha)$ is not defined. As a further consequence of this fact, we have that only greedy approaches can be used to select updates. Indeed, we need to make sure that, after each modification, the parameter α still belongs to the interval of convergence of the series. Thus (P2) can be tackled, but the set of virtual edges among which to select the modification at each step has to be redefined after each update in order to allow α to fit in the interval of convergence of the new series at the subsequent step.

In conclusion, our techniques (as they are) can only be applied when considering (P1), if one wants to tune the resolvent-based total communicability with a fixed value for the parameter.

2.6.2 Varying α

Consider now the case in which we let the parameter α in (2.3) vary as we modify the links of the network. Assume for simplicity that the parameter evolves as a function of the leading eigenvalue of the adjacency matrix. In the numerical tests presented in this subsection we have assumed that $\alpha = \frac{1}{2\lambda_1}$.

In this framework, we are considering a different matrix function at each step:

$$TC(A, f_\alpha) = \mathbf{1}^T f_\alpha(A)\mathbf{1} = \mathbf{1}^T (I - \alpha A)^{-1}\mathbf{1}, \qquad \alpha \in \left(0, \frac{1}{\lambda_1}\right),$$

where $f_\alpha(t) = f(t, \alpha) = (1 - \alpha t)^{-1}$.

Despite the well-posedness of the problems, which is now surely verified, there are two main drawbacks in this approach. Firstly, the fact that the parameter α is varying

forces us to use only greedy approaches in this setting when tackling both (P1) and (P2). Secondly, as we will discuss shortly, it may happen that the total communicability increases as we remove edges or decreases as we add links, as a consequence of the fact that we are changing the target function at each step.

Let us first focus on the case of the downdate. Suppose that the edge $(i,j) \in \mathcal{E}$ is removed from the network. Let $\widehat{A} = A - \mathbf{e}_i\mathbf{e}_j^T - \mathbf{e}_j\mathbf{e}_i^T$ be the new adjacency matrix and let $\widehat{\alpha}$ be the new parameter associated with $\widehat{A}$.

The values of the total communicabilities before and after the downdates are thus denoted by $TC(A, f_\alpha)$ and $TC(\widehat{A}, f_{\widehat{\alpha}})$. We want to deduce under which conditions the resolvent-based total communicability increases after edge removal, i.e., when it holds that

$$TC(\widehat{A}, f_{\widehat{\alpha}}) \geq TC(A, f_\alpha). \tag{2.4}$$

Using the Sherman–Morrison–Woodbury formula [50, p. 50], after some algebraic manipulation, one gets

$$TC(\widehat{A}, f_{\widehat{\alpha}}) = TC(A, f_{\widehat{\alpha}}) - \widehat{\alpha}\ \mathfrak{d}(i,j) \tag{2.5}$$

where

$$\mathfrak{d}(i,j) = \frac{2\widehat{k}_i\widehat{k}_j(1+\widehat{\alpha}C_{ij}) - \widehat{\alpha}(\widehat{k}_i^2 C_{jj} + \widehat{k}_j^2 C_{ii})}{(1+\widehat{\alpha}C_{ij})^2 - \widehat{\alpha}^2 C_{ii}C_{jj}},$$

$C := f_{\widehat{\alpha}}(A) = (I - \widehat{\alpha}A)^{-1}$, and $\widehat{\mathbf{k}} = C\mathbf{1} = (\widehat{k}_i)$. From (2.5) it follows that relation (2.4) is satisfied when

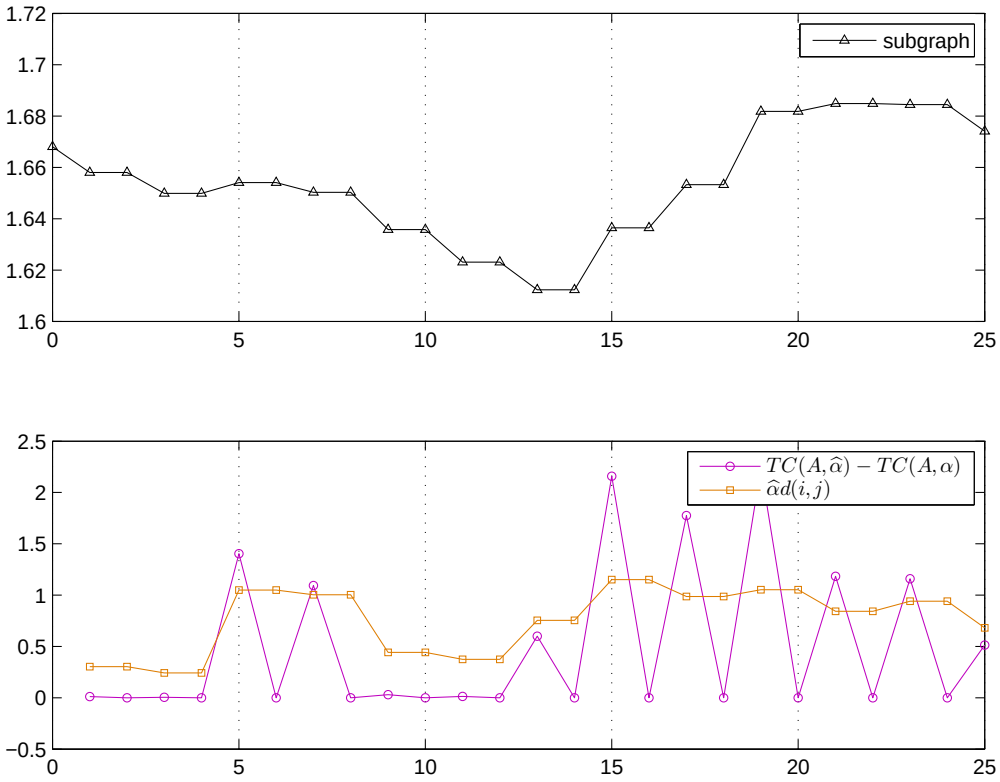
$$TC(A, f_{\widehat{\alpha}}) - TC(A, f_\alpha) \geq \widehat{\alpha}\ \mathfrak{d}(i,j). \tag{2.6}$$

Figure 2.12 displays at the top the evolution of the parametric total communicability as $K = 25$ downdates are performed on the network Sawmill [73] using the method `subgraph`, based on the f_α-subgraph centrality of nodes. More specifically, this method makes use of the edge resolvent-subgraph centrality of nodes, which is a variant of definition 2.4:

$${}^e\mathcal{RC}(i,j) = (I - \alpha A)^{-1}_{ii}(I - \alpha A)^{-1}_{jj}.$$

The bottom plot in Figure 2.12 shows the mutual behavior of the left-hand side and right-hand side of (2.6). As this figure clearly shows, the parametric total communicability increases when (2.6) is satisfied. Therefore, in general, one cannot tackle the downdating problem (P1) with a varying parameter with our heuristics.

FIGURE 2.12: Network: Sawmill. Top: evolution of the parameter dependent total communicability $TC(A, f_\alpha) = \mathbf{1}^T(I - \alpha A)^{-1}\mathbf{1}$ when $K = 25$ downdates are performed using `subgraph`. Bottom: mutual behavior of the left- and right-hand sides of Equation (2.6).



The same reasoning applied to the updating problem leads to the conclusion that, even in this case, the evolution of the parametric total communicability does not necessarily follows the behavior one would expect as edges are added to the network. Indeed, as we will show below, there are situation in which the addition of one edge may lead to the decrease in the value of the total communicability.

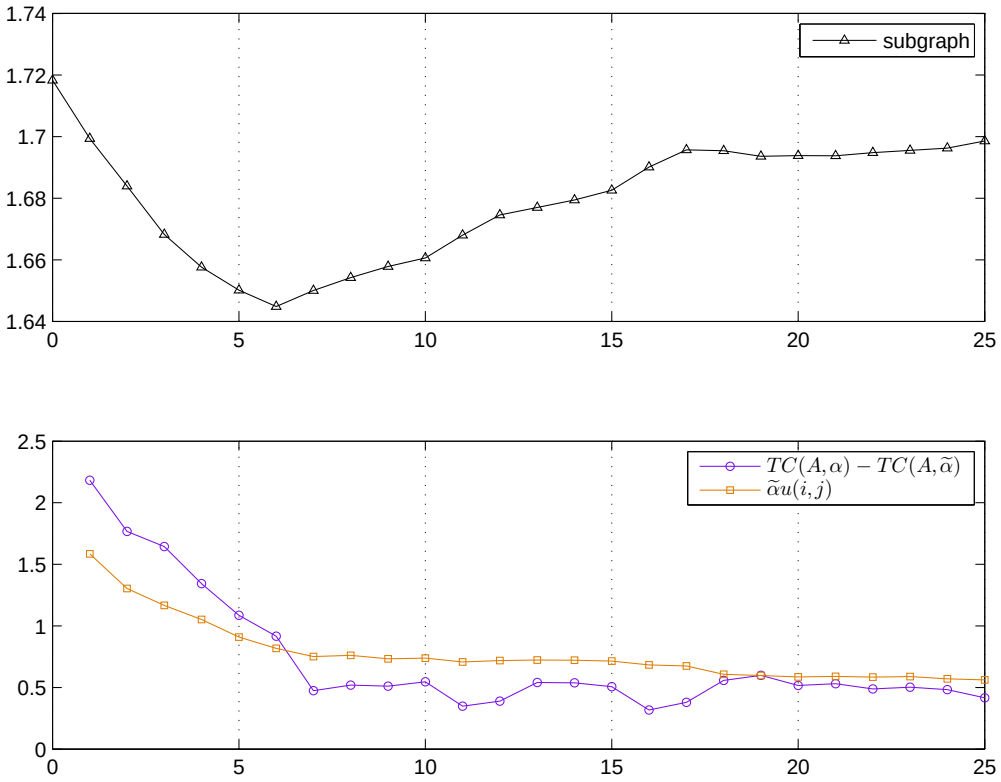
More specifically, we want to understand under which hypothesis the condition:

$$TC(A, f_\alpha) \geq TC(\widetilde{A}, f_{\widetilde{\alpha}}) \tag{2.7}$$

is satisfied, i.e., under which conditions the update of an edge leads to the decrease in the value of the resolvent-based total communicability. Using the Sherman–Morrison–Woodbury formula as before, after some algebraic manipulation, one finds that

$$TC(\widetilde{A}, f_{\widetilde{\alpha}}) = TC(A, f_{\widetilde{\alpha}}) - \widetilde{\alpha}\ \mathfrak{u}(i, j) \tag{2.8}$$

FIGURE 2.13: Network: Zachary. Top: evolution of the parameter dependent total communicability $TC(A, f_\alpha) = \mathbf{1}^T(I - \alpha A)^{-1}\mathbf{1}$ when $K = 25$ updates are performed using `subgraph`. Bottom: mutual behavior of the left- and right-hand sides of Equation (2.9).



where

$$\mathfrak{u}(i,j) = \frac{2\widetilde{k}_i\widetilde{k}_j(1-\widetilde{\alpha}D_{ij}) + \widetilde{\alpha}(\widetilde{k}_i^2 D_{jj} + \widetilde{k}_j^2 D_{ii})}{(1-\widetilde{\alpha}D_{ij})^2 - \widetilde{\alpha}^2 D_{ii}D_{jj}},$$

$D := f_{\widetilde{\alpha}}(A) = (I - \widetilde{\alpha}A)^{-1}$, and $\widetilde{\mathbf{k}} = D\mathbf{1} = (\widetilde{k}_i)$. From (2.8) it follows that relation (2.7) is satisfied when

$$TC(A, f_\alpha) - TC(A, f_{\widetilde{\alpha}}) \geq \widetilde{\alpha}\ \mathfrak{u}(i,j) \tag{2.9}$$

Figure 2.13 displays the evolution of the normalized parametric total communicability (top) and the mutual behavior of the left- and right-hand sides of relation (2.9) (bottom) when $K = 25$ updates are performed to the network Zachary [101] using the method `subgraph`.

As for the case of the downdate, where the removal of one edge could lead to an increase in the value of $TC(A, f_\alpha)$, when performing an update to the network we may as well expect the total communicability to decrease.

In conclusion, the results of this section show that (P1), (P2), and (P3) cannot always be tackled when using the resolvent-based total communicability. This is due to the fact that the function appearing in the definition is parameter dependent, with the parameter varying in an interval that depends on the leading eigenvalue of the adjacency matrix.

Conclusions

In this chapter we have introduced several techniques that can be used to modify an existing network so as to obtain networks that are highly sparse, and yet have a large total communicability.

These heuristics make use of various measures of edge centrality, a few of which have been introduced in [4]. Far from being *ad hoc*, these heuristics are widely applicable and mathematically justified. All our techniques can be implemented using well-established tools from numerical linear algebra: algorithms for eigenvector computation, Gauss-based quadrature rules for estimating quadratic forms, and Krylov subspace methods for computing the action of a matrix function on a vector. At bottom, the Lanczos algorithm is the main player. High quality, public domain software exists to perform these modifications efficiently.

Among all the methods tested for the case of the total communicability, the best results are obtained by the `nodeTC.no` and `eigenvector.no` algorithms, which are based on the edge total communicability and eigenvector centrality, respectively. These methods are extremely fast and returned excellent results in virtually all the tests we have performed. For updating networks characterized by a small spectral gap, a viable alternative is the algorithm `subgraph.no`. While more expensive than `nodeTC.no` and `eigenvector.no`, this method scales linearly with the number of nodes and yields consistently good results.

Moreover, we have shown that the total communicability can be effectively used as a measure of network connectivity, which plays an important role in designing robust networks. Indeed, the total communicability does a very good job at quantifying two related properties of networks: the ease of spreading information, and the extent to which the network is "well connected". Our results show that the total communicability behaves in a manner very similar to the natural connectivity (or free energy) under network modifications, while it can be computed much more quickly.

Finally, we have discussed the case of the resolvent-based total communicability. This case presents some difficulties, since the bilinear form considered in this case is parameter dependent. Furthermore, the parameter lies in an interval defined in terms of λ_1, the leading eigenvalue of the adjacency matrix. We have shown that, if we keep this parameter fixed as we modify the network, our techniques can be used to tackle the downdating problem only. On the other hand, if we let the parameter vary, we have shown that the resolvent-based total communicability may display an opposite behavior with respect to the desired one; namely, it can increase as we remove edges or decrease if we add connections.

Chapter 3

Heuristics for optimizing the communicability of digraphs

In Chapter 2 we considered the problem of modifying an existing sparse network so as to cause the total network communicability to change in some desired way. A serious limitation of the notion of total communicability is that it is not well suited to deal with directed networks, and indeed Chapter 2 deals exclusively with undirected networks. The main reason is that in a directed graph each node plays two roles, that of broadcaster and that of receiver of information. It is clear that a single index cannot discriminate between these two forms of communication. In this chapter, building in part on the ideas in [11], we introduce two new measures of total network communicability, which quantify how easily information is propagated on a given directed network when the two fundamental modes of communication (broadcasting and receiving) have both to be accounted for. Furthermore, we generalize the edge modification criteria in the previous chapter from the undirected to the directed case, using the newly introduced communicability indices as the objective functions.

This chapter is based on [5] and it is organized as follows. The new total communicability indices for digraphs are introduced in Section 3.1. The edge updating/downdating problem is described in Section 3.2. In Section 3.3 we introduce the proposed heuristics for edge manipulation, and in Section 3.4 we discuss the result of numerical tests (including timings) using five real-world directed networks.

3.1 Total network communicabilities for digraphs

In [13] a global measure of how easily information is diffused across an (undirected) network has been defined in terms of the matrix exponential of the adjacency matrix.

Let now A be the adjacency matrix of a directed graph. In analogy with the undirected case, we can consider the total network communicability (1.12) defined in terms of $f(t) = e^t$. In principle, this quantity (possibly normalized by n) gives us an idea of how efficient the network is, globally, at diffusing information. However, by following this approach we would be completely disregarding the twofold nature of nodes, which is one of the main features of digraphs.

To better capture the dual behavior of nodes, we introduce two new global indices of communicability defined in terms of functions of the hub and authority matrices.

Definition 3.1. Let A be the adjacency matrix of a simple digraph and let $f : \mathbb{R} \longrightarrow \mathbb{R}$ be a function defined on the spectrum of AA^T. The *total hub f-communicability* of the digraph is defined as

$$T_hC(A, f) := \mathbf{1}^T f(AA^T)\mathbf{1} = \sum_{i=1}^{n} f(\sigma_i^2)(\mathbf{1}^T \mathbf{u}_i)^2.$$

Similarly, the *total authority f-communicability* of the digraph is defined as

$$T_aC(A, f) := \mathbf{1}^T f(A^T A)\mathbf{1} = \sum_{i=1}^{n} f(\sigma_i^2)(\mathbf{1}^T \mathbf{v}_i)^2.$$

The motivation for using these quadratic forms as total communicability indices is that they exploit the recursive definition that relates hubs and authorities in a directed network. Indeed, if we assume that the function f can be expressed as a power series of the form

$$f(t) = \sum_{k=0}^{\infty} c_k t^k, \quad c_k \geq 0 \quad \forall k = 0, 1, \ldots, \tag{3.1}$$

then an easy computation shows that the total hub f-communicability can be described in terms of the in-degree vector and of the authority matrix. Analogous computations carried out on the total authority f-communicability show that this index can be completely described in terms of the out-degree vector and of the hub matrix. This highlights

the fact that the overall ability of nodes to broadcast information depends on their ability of receiving it and that the overall ability of nodes to receive information depends on how well they are able to broadcast it.

More explicitly, we can write

$$T_hC(A,f) = c_0 n + c_1\|\mathbf{d}_{in}\|_2^2 + \sum_{k=1}^{\infty} c_{k+1}\mathbf{d}_{in}^T(A^TA)^k\mathbf{d}_{in}$$

and

$$T_aC(A,f) = c_0 n + c_1\|\mathbf{d}_{out}\|_2^2 + \sum_{k=1}^{\infty} c_{k+1}\mathbf{d}_{out}^T(AA^T)^k\mathbf{d}_{out}.$$

Note that due to the nonnegativity assumption on the coefficients in (3.1), both $T_hC(A,f)$ and $T_aC(A,f)$ are an inherently nonnegative quantities.

Proposition 3.2. *The total hub and authority f-communicabilities are invariant under graph isomorphism.*

Proof. Let $\mathcal{G}_1$ and $\mathcal{G}_2$ be two isomorphic graphs with associated adjacency matrices A_1 and A_2. Then there exists a permutation matrix P such that $A_2 = PA_1P^T$. Therefore,

$$\begin{aligned} T_hC(A_2,f) &= \mathbf{1}^T f(A_2A_2^T)\mathbf{1} = \mathbf{1}^T f(PA_1P^TPA_1^TP^T)\mathbf{1} \\ &= \mathbf{1}^T Pf(A_1A_1^T)P^T\mathbf{1} = \mathbf{1}^T f(A_1A_1^T)\mathbf{1} = T_hC(A_1,f). \end{aligned}$$

Similar computations carried out on the total authority f-communicability lead to $T_aC(A_2,f) = T_aC(A_1,f)$. □

In the remaining of this chapter, we will focus on the total hub and authority f-communicabilities when the function $f(t) = \cosh(\sqrt{t})$ is used in the definition. The choice of the function $f(t)$ may seem unusual; however, we argue that this choice is the most natural one if one wants to "translate" the idea of total communicability to the case of digraphs. Indeed, in the undirected case the total communicability was defined in terms of the matrix exponential, since we were to count all the walks of any length taking place in the network, weighting walks of length k by a factor $\frac{1}{k!}$. In the case of a digraph, we want to count all the alternating walks, again penalizing walks of length k by a factor $\frac{1}{k!}$. Recall that the kth power of the hub and authority matrices contain information concerning the number of alternating walks of length $2k$ taking place in the

networks. Hence, the *total hub communicability* will be defined as

$$T_hC(A) := \mathbf{1}^T \left(\sum_{k=0}^{\infty} \frac{(AA^T)^k}{(2k)!} \right) \mathbf{1} = \mathbf{1}^T \left(\sum_{k=0}^{\infty} \frac{(\sqrt{AA^T})^{2k}}{(2k)!} \right) \mathbf{1}$$
$$= \mathbf{1}^T \cosh(\sqrt{AA^T})\mathbf{1} = T_hC(A, \cosh(\sqrt{t})),$$

and, similarly, the *total authority communicability* will be defined as:

$$T_aC(A) := \mathbf{1}^T \left(\sum_{k=0}^{\infty} \frac{(A^TA)^k}{(2k)!} \right) \mathbf{1} = \mathbf{1}^T \left(\sum_{k=0}^{\infty} \frac{(\sqrt{A^TA})^{2k}}{(2k)!} \right) \mathbf{1}$$
$$= \mathbf{1}^T \cosh(\sqrt{A^TA})\mathbf{1} = T_aC(A, \cosh(\sqrt{t})).$$

A further justification for the choice of the function $f(t)$ comes from considering the bipartite graph associated with the digraph under study (see [11]). Let $\mathscr{A}$ be the adjacency matrix of the bipartite graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ obtained from the original digraph represented by A as described in Subsection 1.1.3. Then the following result holds true.

Proposition 3.3. *Let $\mathscr{A}$ be the adjacency matrix of the undirected, bipartite graph obtained from the adjacency matrix A of a digraph. Then*

$$e^{\mathscr{A}} = \begin{pmatrix} \cosh(\sqrt{AA^T}) & \sinh^{\diamond}(A) \\ \sinh^{\diamond}(A^T) & \cosh(\sqrt{A^TA}) \end{pmatrix}, \tag{3.2}$$

where $\sinh^{\diamond}(A)$ *is the generalized hyperbolic sine of the matrix A (cf.definition 1.17).*

Proof. The result easily follows from (1.5) □

An important feature of this matrix is that its entries are nonnegative. Thus, these quantities can be used to describe the importance of nodes and how well they communicate when they are acting as broadcasters or receivers of information in the graph [11]. Indeed, the entries of the diagonal block $\cosh(\sqrt{AA^T})$ provide centrality and communicability indices for nodes and pairs of nodes when they are all seen as broadcaster of information in the network. The diagonal entries give the centralities for the nodes in the original network when they are seen as broadcasters of information (hubs), while the off-diagonal entries measure how well two nodes, both acting as broadcasters, exchange information. Similarly, when considering the entries of the diagonal block $\cosh(\sqrt{A^TA})$,

we are looking at centrality and communicability indices for nodes and pairs of nodes when they are all seen as playing the role of authorities in the digraph.

Thus, the total hub communicability and total authority communicability defined as $T_hC(A) = \mathbf{1}^T \cosh(\sqrt{AA^T})\mathbf{1}$ and $T_aC(A) = \mathbf{1}^T \cosh(\sqrt{A^TA})\mathbf{1}$, respectively, account for the overall ability of the network of exchanging information when all its nodes are playing the same role of broadcasters ($T_hC(A)$) or receivers ($T_aC(A)$).

As for the off-diagonal blocks in (3.2), they contain information concerning how nodes exchange information when one node is playing the role of broadcaster (resp., receiver) and the other is acting as a receiver (resp., broadcaster).

3.2 Edge modification strategies

Our main goal is to develop heuristics that can be used to add/remove edges from a digraph in order to tune the total hub and/or authority communicability. In particular, we will call *update* of $(i, j) \notin \mathcal{E}$ the addition of this virtual edge to the network; we want to perform this operation in such a way that this addition increases as much as possible the quantities of interest. Note that, due to the nonnegativity condition in (3.1), the addition of an edge can only increase the total communicabilities $T_hC(A)$ and $T_aC(A)$.

The operation of removing an edge from the network will be referred to as the *downdate* of an edge. Our aim is to select the edge to be removed in such a way that the target functions T_hC and T_aC are not penalized too much, i.e., their values do not drop significantly as edges are removed.[1] Both these operations can be described as rank-one modifications of the adjacency matrix A of the digraph $\mathcal{G}$ or, equivalently, as rank-two modifications of the adjacency matrix $\mathscr{A}$ of the associated bipartite graph $\mathscr{G}$.

As in the undirected case, in order to describe our updating and downdating strategies we first introduce some edge centrality measures that can be used to rank the (virtual) edges in the digraph; then, the derived rankings are used to select the modifications to be performed. More specifically, a virtual edge having a large centrality is considered important and thus its addition is expected to highly enhance the total communicabilities. On the other hand, we will remove edges that have a low ranking, since they are

[1] Clearly, our approach can be adapted so as to obtain the opposite effect if so desired. Indeed, we can adapt our algorithms to select edges whose removal heavily penalizes the target functions.

not expected to carry a lot of information; thus, their removal is not expected to heavily penalize the hub and authorities communicabilities of the network.

The resulting updating and downdating strategies will be similar in spirit to those adopted in the undirected case (cf. Chapter 2). However, as explained in more detail in the next subsection, we cannot simply apply the heuristics developed for the undirected case to the bipartite graph $\mathscr{G}$, since doing so could lead to possible loss of structure.

3.2.1 Bipartite graphs vs. digraphs

In the remaining of this section we will describe in more detail how to tackle the problem of selecting K edge modifications to be performed on the network in order to tune the total hub and authority communicability indices: $T_hC(A)$ and $T_aC(A)$.

First we describe how to rank the edges. A priori, there are two natural approaches. Indeed, given the definitions of our communicability indices in terms of the function $f(t) = \cosh(\sqrt{t})$, we can either work on the matrix (1.1) obtained from A as described in Subsection 1.1.3, or on the original adjacency matrix A. When working on $\mathscr{A}$, we would adapt to this matrix the techniques developed for the undirected case which performed best according to the results in Chapter 2, taking into account the need to preserve the zero-nonzero block structure of $\mathscr{A}$. More specifically, we will be using the edge eigenvector centrality ${}^{e}\mathcal{EC}$ (see definition 2.6) and the edge total communicability centrality ${}^{e}\mathcal{TC}$ (see definition 3.10) to rank the (virtual) edges. When working directly on the unsymmetric matrix A, on the other hand, the introduction of new edge centrality measures specially developed for the directed case is required.

We will show that the new edge centrality measures for digraphs allow us to develop heuristics that perform as well as or better than the techniques for undirected graphs applied to $\mathscr{A}$.

Remark 3.4. The set of virtual edges among which we select the updates is the same in both cases, since one wants to preserve the antidiagonal block structure of $\mathscr{A}$. Indeed, if a new edge were to destroy the structure, it could not be "translated" into a new directed edge for the original digraph.

3.2.2 Edge centralities: directed case

We now want to define two new edge centrality measures that take into account the directionality of links and that can be computed by directly working on the unsymmetric adjacency matrix A.

In Section 2.2 it has been pointed out that one of the main factors in the evolution of the total communicability is the dominant eigenvalue λ_1 of the matrix involved in its computation.

Transferring this idea to $T_hC(A)$ and $T_aC(A)$, it follows that we want to define (if possible) an edge centrality measure that allows us to control the change in the leading singular value of A, which corresponds to the square root of the leading eigenvalue of AA^T and A^TA.

Indeed, when considering the total communicability indices for the directed case, we have that, when $\sigma_1 \gg \sigma_2$, then

$$T_hC(A) \approx e^{\sigma_1}(\mathbf{u}_1^T\mathbf{1})^2, \quad T_aC(A) \approx e^{\sigma_1}(\mathbf{v}_1^T\mathbf{1})^2;$$

therefore, a major contribution to the values of the two communicability indices comes from the leading singular triplet $(\sigma_1, \mathbf{u}_1, \mathbf{v}_1)$ of A.

The following result provides guidelines for the introduction of a first edge centrality measure for the case of digraphs.

Proposition 3.5. *Let A be the adjacency matrix of a digraph. Let $\mathbf{u}_1$ and $\mathbf{v}_1$ be the hub and authority vectors, respectively. Let σ_1 be the leading singular value of A. Consider the adjacency matrix of the graph obtained after the addition of the virtual edge (i, j): $\tilde{A} = A + \mathbf{e}_i\mathbf{e}_j^T$. Then the leading eigenvalue $\tilde{\sigma}_1^2$ of the new hub and authority matrices satisfies*

$$\tilde{\sigma}_1^2 \geq \sigma_1^2 + 2\sigma_1 u_1(i)v_1(j) + \max\left\{u_1(i)^2, v_1(j)^2\right\}. \tag{3.3}$$

The inequality is strict if AA^T is irreducible.

Moreover, let $\widehat{A} = A - \mathbf{e}_i\mathbf{e}_j^T$ denote the adjacency matrix obtained after the removal of the edge $i \to j$ from the digraph. Then the leading eigenvalue $\widehat{\sigma}_1^2$ of the new hub and

authority matrices satisfies

$$\sigma_1^2 \geq \widehat{\sigma}_1^2 \geq \sigma_1^2 - 2\sigma_1 u_1(i)v_1(j) + \max\left\{u_1(i)^2, v_1(j)^2\right\}. \tag{3.4}$$

The first inequality is strict if $\widehat{A}\widehat{A}^T$ is irreducible.

Proof. Using the Rayleigh–Ritz Theorem (see, for example, [57, Theorem 4.2.2]) we get:

$$\begin{aligned}
\tilde{\sigma}_1^2 = \lambda_1(\tilde{A}\tilde{A}^T) &= \max_{\|\mathbf{z}\|_2=1} \mathbf{z}^T \left(\tilde{A}\tilde{A}^T\right)\mathbf{z} \\
&\geq \mathbf{u}_1^T \left(\tilde{A}\tilde{A}^T\right)\mathbf{u}_1 \\
&= \left\|\left(A^T + \mathbf{e}_j\mathbf{e}_i^T\right)\mathbf{u}_1\right\|_2^2 \\
&= \|\sigma_1\mathbf{v}_1 + u_1(i)\mathbf{e}_j\|_2^2 \\
&= \sigma_1^2 + 2\sigma_1 u_1(i)v_1(j) + u_1(i)^2.
\end{aligned}$$

Similarly, by working on the authority matrix one gets:

$$\tilde{\sigma}_1^2 = \lambda_1(\tilde{A}^T\tilde{A}) \geq \mathbf{v}_1^T \left(\tilde{A}^T\tilde{A}\right)\mathbf{v}_1 = \sigma_1^2 + 2\sigma_1 u_1(i)v_1(j) + v_1(j)^2.$$

From these inequalities, and from basic facts from Perron–Frobenius theory, the conclusion easily follows. Similar arguments can be used to prove (3.4). □

Relations (3.3) and (3.4) motivate the following definition.

Definition 3.6. Let A be the adjacency matrix of a directed graph. Let $\mathbf{u}_1$ and $\mathbf{v}_1$ be its hub and authority vectors, respectively. Then the *edge HITS centrality* of the existing/virtual edge (i, j) is defined as

$$^eHC(i, j) = u_1(i)v_1(j).$$

Notice that when A is symmetric this definition reduces to that of edge eigenvector centrality: $^e\mathcal{EC}(i, j) = x_1(i)x_1(j)$, where $\mathbf{x}_1$ is the eigenvector associated with the leading eigenvalue of A.

Remark 3.7. Inequalities (3.3) and (3.4) and, consequently, definition 3.6 suggest that there is a "prescribed direction" one has to follow when introducing a new edge centrality

measure. Indeed, it is required to use the centrality as broadcaster for the source node i and the centrality as receiver for the target node j when evaluating the importance of the (virtual) edge $i \to j$. This observation confirms a natural intuition and motivates the usage of this same "orientation" in all our definitions and methods (cf. Section 3.3).

The next edge centrality measure we want to define relies on the use of the total communicability of nodes. This quantity describes how well node i communicates with the whole network. As discussed in Subsection 1.4.4, this centrality measure is well defined for any adjacency matrix, in particular for the adjacency matrices of digraphs, and indeed the row and column sums of e^A do provide in some cases meaningful measures of how well nodes broadcast information (row sums of e^A) and how good they are at receiving information (column sums of e^A). However, the expressions describing these quantities do not provide information on the alternating walks taking place in the digraph and, thus, miss a crucial feature of communication in real-world directed networks.

For this reason, we introduce here new definitions for the total communicabilities of nodes which can be shown to be directly connected to their twofold nature. In order to do so, we make use of the concept of *generalized matrix function* first introduced in [55]. Recall from Chapter 1 that, given a matrix $A = U_r \Sigma_r V_r^T \in \mathbb{R}^{n\times n}$ of rank r and a scalar function $f : \mathbb{R} \longrightarrow \mathbb{R}$ such that $f(\sigma_i)$ exists for all $i = 1, 2, \ldots, r$, the induced generalized matrix function $f^\diamond : \mathbb{R}^{n\times n} \longrightarrow \mathbb{R}^{n\times n}$ is defined as

$$f^\diamond(A) = U_r f(\Sigma_r) V_r^T = \sum_{k=1}^{r} f(\sigma_k)\mathbf{u}_k\mathbf{v}_k^T.$$

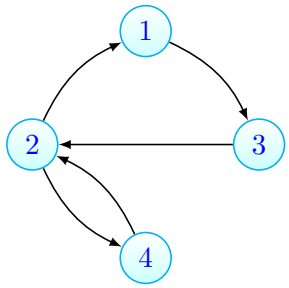
It is easy to check that[2]

$$f^\diamond(A) = \left(\sum_{k=1}^{r} \frac{f(\sigma_k)}{\sigma_k}\mathbf{u}_k\mathbf{u}_k^T\right) A = A \left(\sum_{k=1}^{r} \frac{f(\sigma_k)}{\sigma_k}\mathbf{v}_k\mathbf{v}_k^T\right). \tag{3.5}$$

These equalities show that a generalized matrix function can be expressed in terms of A and either AA^T or A^TA. Therefore, the entries of $f^\diamond(A)$ — and hence its row/column sums — can be used as meaningful measures of importance in the directed case, *provided that they are all non-negative.*

[2]We will prove this and other properties of generalized matrix functions in Chapter 4.

FIGURE 3.1: Digraph associated with the adjacency matrix described in (3.6)



It turns out that, in general, this is not the case for the generalized matrix exponential. Indeed, consider for example the generalized matrix exponential of the adjacency matrix

$$A = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \tag{3.6}$$

associated with the digraph in Figure 3.1.

It turns out that its $(3,1)$ and $(4,4)$ entries are negative, and thus these quantities cannot be interpreted as communicability/centrality measures.

If we instead consider the generalized hyperbolic sine:

$$\sinh^{\diamond}(A) = U_r \sinh(\Sigma_r) V_r^T = \sum_{k=1}^{r} \sinh(\sigma_k) \mathbf{u}_k \mathbf{v}_k^T,$$

we have that this matrix corresponds to the top right block of the matrix $e^{\mathscr{A}}$ (see (3.2)). Hence, the entries of $\sinh^{\diamond}(A)$ are all non-negative, and can be used to quantify how well nodes communicate when they are playing different roles. More precisely, reasoning in terms of alternating walks shows that the (i,j)th entry of this matrix describes how well node i exchanges information with node j when the first is playing the role of hub and the latter that of authority. Using this generalized matrix function we can introduce two new centrality measures for nodes in digraphs.

Definition 3.8. Let $A = U_r \Sigma_r V_r^T$ be the rank-r adjacency matrix of a directed network. We call *total hub communicability of node i* the quantity

$$C_h(i) = \mathbf{e}_i^T \sinh^{\diamond}(A)\, \mathbf{1} = \sum_{k=1}^{r} \sinh(\sigma_k)(\mathbf{v}_k^T \mathbf{1}) u_k(i)$$

and *total authority communicability of node j* the quantity

$$C_a(j) = \mathbf{1}^T \sinh^{\diamond}(A)\, \mathbf{e}_j = \sum_{k=1}^{r} \sinh(\sigma_k)(\mathbf{u}_k^T \mathbf{1}) v_k(j)$$

These quantities correspond to row and column sums of the off-diagonal block of $e^{\mathscr{A}}$; therefore, $C_h(i)$ quantifies the ability of node i — playing the role of hub — to communicate with all the nodes in the network, when they are all acting as receivers of information. Similarly, $C_a(j)$ accounts for the ability of node j as an authority to receive information from all the nodes in the graph, when they are acting as broadcasters of information.[3] This feature highlights the fact that these definitions are better suited than $e^A\mathbf{1}$ and $(\mathbf{1}^T e^A)^T$ when it comes to working on digraphs. The following proposition highlights how these new centrality indices for nodes can be fully described in terms of how well nodes either broadcast or receive information.

Proposition 3.9. *Let A be the adjacency matrix of a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The total hub communicability of node $i \in \mathcal{V}$ can be written as*

$$C_h(i) = \sum_{k=1}^{r} \frac{\sinh(\sigma_k)}{\sigma_k} \mathbf{e}_i^T (\mathbf{u}_k \mathbf{u}_k^T) \mathbf{d}_{out} = \sum_{k=1}^{r} \frac{\sinh(\sigma_k)}{\sigma_k} (\mathbf{v}_k^T \mathbf{1}) \sum_{\substack{\ell \in \mathcal{V} \\ i \to \ell}} v_k(\ell). \tag{3.8a}$$

Similarly, the total authority communicability of node $j \in \mathcal{V}$ can be expressed as

$$C_a(j) = \sum_{k=1}^{r} \frac{\sinh(\sigma_k)}{\sigma_k} \mathbf{d}_{in}^T (\mathbf{v}_k \mathbf{v}_k^T) \mathbf{e}_j = \sum_{k=1}^{r} \frac{\sinh(\sigma_k)}{\sigma_k} (\mathbf{1}^T \mathbf{u}_k) \sum_{\substack{\ell \in \mathcal{V} \\ \ell \to j}} u_k(\ell). \tag{3.8b}$$

Proof. Using the first equality in (3.5) one gets that:

$$C_h(i) = \mathbf{e}_i^T \left(\sum_{k=1}^{r} \frac{\sinh(\sigma_k)}{\sigma_k} \mathbf{u}_k \mathbf{u}_k^T \right) A\mathbf{1} = \sum_{k=1}^{r} \frac{\sinh(\sigma_k)}{\sigma_k} \mathbf{e}_i^T \left(\mathbf{u}_k \mathbf{u}_k^T \right) \mathbf{d}_{out},$$

[3] The reader is referred to [11] for a more detailed discussion of the interpretation of the entries in the off-diagonal blocks of $e^{\mathscr{A}}$.

TABLE 3.1: Centrality measures for the nodes in the graph represented in Figure 3.1 and described by the adjacency matrix (3.6).

NODE	$d_{out}(i)$	$d_{in}(i)$	$u_1(i)^2$	$v_1(i)^2$	$C_h(i)$	$C_a(i)$
1	1	1	.0000	.3333	1.1752	1.3683
2	2	2	.5000	.3333	2.7366	2.7366
3	1	1	.2500	.0000	1.3683	1.1752
4	1	1	.2500	.3333	1.3683	1.3683

which proves the first equality of (3.8a). To prove the second, we apply the second equality in (3.5):

$$C_h(i) = (\mathbf{e}_i^T A)\left(\sum_{k=1}^{r} \frac{\sinh(\sigma_k)}{\sigma_k}\mathbf{v}_k\mathbf{v}_k^T\right)\mathbf{1} = \sum_{k=1}^{r} \frac{\sinh(\sigma_k)}{\sigma_k}(\mathbf{v}_k^T\mathbf{1})A_{i,:}\mathbf{v}_k,$$

where $A_{i,:}$ is the ith row of the adjacency matrix A. The conclusion then follows from the fact that $A_{i,:}\mathbf{v}_k = \sum_{i\to\ell} v_k(\ell)$. The proof of (3.8b) goes along the same lines and is thus omitted. □

Before proceeding with the introduction of the associated edge centrality measure, we want to show with a small example that these measures of hub and authority centrality are indeed informative. Consider as an example the graph in Figure 3.1. It is intuitive that node 2 should be given the highest score both as hub and as authority by any reasonable centrality measure. Consequently, the authority scores for nodes 1 and 4 should be the same and higher than that of node 3 because these nodes are directly pointed to from node 2, which is the best hub in the graph. For a similar reason, nodes 3 and 4 should be ranked higher than node 1 when considering a hub score, since they directly point to node 2, which is the most important authority.

Table 3.1 contains the centrality scores for the four nodes when the in/out-degree, HITS centrality[4], and the total hub/authority communicability are considered. Clearly, the in/out-degrees of the nodes do not capture the picture we just described since they cannot discriminate between nodes 1, 3, and 4. This happens because the degree centralities take into account only local information about how nodes propagate information in the network.

[4]To compute these scores, we initialize the HITS algorithm with the constant authority vector with 2-norm equal to 1; see [11, 60].

Concerning HITS, the rankings given by the hub scores match our expectations, but those given by the authority scores do not, since they are unable to identify node 2 as the most authoritative one (it is tied with nodes 1 and 4). Another problem with HITS is that the rankings will depend in general on the initial vector, since for this example the matrices AA^T and A^TA are reducible (this also explains the occurrence of zero entries in the hub and authority vectors). Note that this is a non-issue for both $C_h(i)$ and $C_a(i)$; most importantly, however, these two measures succeed in identifying the "correct" relative rankings for the hubs and authorities in this digraph.

We now introduce the edge centrality measure associated with definition 3.8.

Definition 3.10. Let A be the adjacency matrix of a simple digraph. Then the *edge generalized total communicability centrality* of the existing/virtual edge (i, j) is defined as

$$^{e}gTC(i,j) = C_h(i)C_a(j),$$

where $C_h(i)$ and $C_a(j)$ are the total hub communicability of node i and the total authority communicability of node j, respectively.

Note that when the difference between the two largest singular values $\sigma_1 - \sigma_2$ is "large enough", the quantities $C_h(i)$ and $C_a(j)$ are essentially determined by $\sinh(\sigma_1)\|\mathbf{v}_1\|_1 u_1(i)$ and $\sinh(\sigma_1)\|\mathbf{u}_1\|_1 v_1(j)$, respectively. When this condition is satisfied we expect agreement between the rankings for the edges provided by the edge HITS and generalized total communicability centrality measures, at least when the attention is restricted to the top ranked edges.

It is natural to ask how the edge centrality measure just introduced is related to the edge total communicability centrality applied to the symmetric matrix $\mathscr{A}$. For the centrality of the (virtual) edge (i, j) we obtain

$$^{e}\mathcal{TC}(i,j') - [^{e}gTC(i,j)] = \phi(i,j) - \left(\cosh(\sqrt{AA^T})\mathbf{1}\right)_i \left(\cosh(\sqrt{A^TA})\mathbf{1}\right)_j \tag{3.9}$$

where $^{e}\mathcal{TC}(i,j')$ is the edge total communicability of (i,j') in the bipartite graph $\mathscr{G}$, $j' = j + n$, and

$$\phi(i,j) = (e^{\mathscr{A}}\mathbf{1})_i \left(\cosh(\sqrt{A^TA})\mathbf{1}\right)_j + (e^{\mathscr{A}}\mathbf{1})_{j'} \left(\cosh(\sqrt{AA^T})\mathbf{1}\right)_i.$$

The difference in (3.9) is positive and it may be so large that the edge selected when working on the digraph could well be different from that selected when working on the associated bipartite network, thus leading to different results for the two techniques. As we will see in the section on numerical experiments, the two criteria may indeed lead to different results.

Remark 3.11. Concerning the actual computation of the quantities that occur in definition 3.8, one can either exploit the relationship (3.2) between $e^{\mathscr{A}}$ and $\sinh^{\diamond}(A)$ and use standard methods for computing the matrix exponential [56] or, if the matrix A is too large to build and work with $\mathscr{A}$ explicitly, one can obtain estimates of the quantities of interest using the Golub–Kahan algorithm [49, 50]. Indeed, $\sinh^{\diamond}(A)$ can be rewritten as[5]

$$\sinh^{\diamond}(A) = \sinh(\sqrt{AA^T})(\sqrt{AA^T})^{\dagger}A = A(\sqrt{A^TA})^{\dagger}\sinh(\sqrt{A^TA}),$$

where "†" denotes the Moore–Penrose pseudoinverse, and one can obtain estimates of the desired row and column sums by applying Golub–Kahan bidiagonalization with an appropriate starting vector ($A\mathbf{1}$ or $A^T\mathbf{1}$. respectively). The test matrices used in the numerical tests are small enough that we could form and manipulate the matrix $\mathscr{A}$ explicitly. Therefore, we expect the heuristics based on the two edge centrality measures ${}^{e}gTC(i,j)$ and ${}^{e}\mathcal{TC}(i,j')$ to perform similarly in terms of timings. These computational issues are investigated in more detail in Chapter 4.

3.3 Heuristics

In this section we describe the methods we will use to perform the numerical tests presented in Section 3.4. For both the updating and downdating problem, we will first rank the (virtual) edges using a variety of edge centrality measures; for large graphs we may consider only a subset of all possible candidate edges, as discussed below. For the updating problem, we will then select the top ranked virtual edges, while for the the downdating problem we will select the edges having the lowest centrality rankings. This is because we want to increase as much as possible the total communicability indices when adding links to the digraph, and to decrease them only slightly when removing connections. As we have done in Chapter 2 for the undirected case, given a budget of K

[5]The reader is referred once again to Chapter 4 for a more detailed discussion of the properties of generalized matrix functions.

modifications to be performed, we can proceed in one of two ways. We can either perform one edge modification at a time and then recalculate all the necessary centrality scores right afterwards, or we can perform all the modifications at once, without recalculation. This latter approach will correspond to the `.no` variants of the algorithms. Recall that, in the undirected case, the latter approach was found to be essentially as effective as the former (even for relatively large K) while being dramatically less expensive in terms of computational effort.

As we already mention in Subsection 3.2.1, we can either work on the bipartite network associated with the digraph or directly on the original network. When working on the original graph, addition/deletion of an edge corresponds to rank-one updates/downdates to the corresponding adjacency matrix A.

The methods used in this framework are labeled as follows:

- `eig(.no)`. Let $\mathbf{x}_1$ be the right eigenvector associated with the leading eigenvalue of A (assumed to be simple) and $\mathbf{y}_1$ be the left eigenvector associated with the same eigenvalue. Generalizing definition 2.6 of edge eigenvector centrality to the directed case, we obtain:

$$^{e}EC(i,j) := x_1(i)y_1(j).$$

  This quantity has been recently used in [89] to devise algorithms aimed at increasing as much as possible the leading eigenvalue of A when edges are added to the network.

- `TC(.no)`. Here we use the total communicability for nodes as defined in Subsection 1.4.4. The score assigned to a (virtual) edge (i,j) is:

$$^{e}TC(i,j) := \left(e^{A}\mathbf{1}\right)_i \left(\mathbf{1}^T e^{A}\right)_j.$$

  This heuristic generalizes to the case of digraphs the technique labeled `nodeTC(.no)` in Chapter 2.

- `HITS(.no)`. Each (virtual) edge is given a score in terms of the quantities introduced in definition 3.6:

$$^{e}HC(i,j) = u_1(i)v_1(j)$$

- `gTC(.no)`. This heuristic is based on the edge generalized total communicability defined in terms of the generalized hyperbolic sine (see definition 3.10). The (virtual) edge (i, j) is assigned the score:

$$^{e}gTC(i,j) = C_h(i)C_a(j),$$

where $C_h(i) = (\sinh^{\diamond}(A)\mathbf{1})_i$ and $C_a(j) = (\sinh^{\diamond}(A^T)\mathbf{1})_j$.

The first two methods (with their variants) generalize to the case of digraphs the techniques which performed best in the undirected case. Notice that, following what observed in Remark 3.7, we have used the broadcaster score for the source node and the receiver score for the target node.

Next, we consider the bipartite network associated to the matrix $\mathscr{A}$ defined in (1.1). The criteria we use to select the modifications are based on the edge centrality measures described in Chapter 2. We will label the methods as follows:

- `b:eig(.no)`. We use the eigenvector centrality of edges; the edge eigenvector centrality of the (virtual) edge (i, j') is defined as

$$^{e}\mathcal{EC}(i,j') = q_1(i)q_1(j'),$$

where $\mathbf{q}_1$ is the Perron vector of $\mathscr{A}$.

- `b:TC(.no)`. This is based on the total communicability centrality of edges: each (virtual) edge (i, j') is assigned the score:

$$^{e}\mathcal{TC}(i,j') = \left(e^{\mathscr{A}}\mathbf{1}\right)_i \left(e^{\mathscr{A}}\mathbf{1}\right)_{j'}.$$

- `b:deg`. This simple heuristic is equivalent to the `degree` method in the undirected case. Each (virtual) edge is assigned a score of the form:

$$d(i) + d(j'), \quad i \in \mathcal{V} \text{ and } j' \in \mathcal{V}',$$

where $d(i) = (\mathscr{A}\mathbf{1})_i$ is the degree of node i in the network represented by $\mathscr{A}$.

Remark 3.12. We do not provide a method that generalizes `degree` in Chapter 2 to the case of digraphs since it would coincide with the heuristic `b:deg` just introduced. Indeed,

the straightforward generalization would require to assign to the (virtual) edge $i \to j$ the score $d_{out}(i) + d_{in}(j)$, since we need to use the degree centrality as a broadcaster for the source node i and that as a receiver for the target node j (cf. Remark 3.7). However, it is easy to see that $d_{out}(i) = d(i)$ where $i \in \mathcal{V}$ and $d_{in}(j) = d(j')$ where $j' \in \mathcal{V}'$, and thus this technique would be indistinguishable from `b:deg`.

Remark 3.13. The technique based on the degree centralities of nodes is the optimal one if we want to optimize the sum $T_h C(A) + T_a C(A)$ and we use the second order Maclaurin approximations $\cosh(\sqrt{X}) \approx I + \frac{X}{2}$, with $X = AA^T, A^T A$ to compute the the total hub and authority communicabilities.

When working on the matrix associated with the bipartite graph, each edge modification of the corresponding network will cause a rank-two change in $\mathscr{A}$. We want to stress once again that the set of virtual edges among which to select the updates is the same whether we work on A or on $\mathscr{A}$ and corresponds to the set of virtual edges of the graph $\mathcal{G}$, or a subset of it. For large networks, the set of virtual edges among which to select the updates may be too large to be exhaustively searched. In the numerical experiments we used the whole set $\overline{\mathcal{E}}$ for all networks except for the largest one, namely cit-HepTh (see Table 3.2). For this test case, we restricted our search to a subset of the set of all virtual edges constructed as follows. We first rank in descending order the nodes of $\mathscr{G}$ using the eigenvector centrality. This results in a ranking of $2n$ elements: the nodes in $\mathcal{V}$ and their copies. Next, for each $i = 1, \ldots, n$ we remove from the list the one element between i and its copy i' which has the lowest rank. We now have a list of length n which includes either one element (element of $\mathcal{V}$) or its copy (element of $\mathcal{V}'$). We thus relabel all the copies, if present, with the label of the corresponding node in $\mathcal{V}$. The resulting list contains all the n nodes in the original graph. It has been obtained considering, for each node, its best performance between its role as hub and its role as authority in the network. Finally, we take the induced subgraph corresponding to the top 10% of the nodes in this list. The set of virtual edges in this subgraph is the set we exhaustively search.

3.3.1 Rank-two modifications

Before discussing the results obtained by applying our techniques to select rank-one updates of the matrix A, we want to briefly discuss how these techniques may be modified in order to make them suitable to select symmetric rank-two modifications of the unsymmetric adjacency matrix. This approach goes beyond our scope, but it is worth some discussion. Indeed, in real world applications one may conceivably want to add (or delete) two-directional edges between nodes in a digraph in order to tune the total communicability indices. In this setting, the downdating and updating problems aim at the same goals as before, but the sets in which one searches for modifications are different from those used in our original problems. Indeed, the updates will be selected in the set $\{(i,j) \in \mathcal{V} \times \mathcal{V} | (i,j),(j,i) \notin \mathcal{E}\}$, while the downdates will be selected among the edges in $\{(i,j) \in \mathcal{V} \times \mathcal{V} | (i,j),(j,i) \in \mathcal{E}\}$.

We start by discussing the case of the degree and of the edge HITS centrality. The results obtained for these two approaches will motivate the generalization of the other techniques. As we have observed in Remark 3.13, the degree strategy works as the optimal strategy when we consider a second order approximation of the terms in the sum $T_h C(A) + T_a C(A)$. By carrying out the same computation, replacing a rank-one update of the adjacency matrix with a rank-two update, one finds that the most natural generalization requires that the quantities used to rank the (virtual) edges by the method based on the degree of nodes are

$$[d_{in}(i) + d_{out}(j)] + [d_{out}(i) + d_{in}(j)].$$

A similar results can be obtained if we want to adapt `HITS` to handle rank-two updates. Indeed, to rank the undirected (virtual) edges one may use the quantities

$$^{e}HITS(i,j) + {}^{e}HITS(j,i).$$

This follows from the application to the matrices $(A + \mathbf{e}_i\mathbf{e}_j^T + \mathbf{e}_j\mathbf{e}_i^T)(A + \mathbf{e}_i\mathbf{e}_j^T + \mathbf{e}_j\mathbf{e}_i^T)^T$ and $(A + \mathbf{e}_i\mathbf{e}_j^T + \mathbf{e}_j\mathbf{e}_i^T)^T(A + \mathbf{e}_i\mathbf{e}_j^T + \mathbf{e}_j\mathbf{e}_i^T)$ of the same technique used in the proof of Proposition 3.5.

TABLE 3.2: Description of the dataset.

NETWORK	n	m	τ	σ_1	σ_2	$\sigma_1 - \sigma_2$
GD95b	73	96	5160	4.79	4.37	0.428
Comp. Complexity	857	1596	731996	10.93	9.87	1.05
Abortion	2262	9624	5104728	31.91	20.04	5.87
Twitter	3656	188712	13176871	189.15	120.54	68.71
cit-HepTh	27400	352547	3730367	85.16	69.31	15.85

From these simple results, it follows that the quantities used by the other heuristics to handle rank-two modifications of the adjacency matrix of a digraph have the form

$$^{e}C(i,j) + {}^{e}C(j,i),$$

where ${}^{e}C$ is one among the edges centralities used in the previous section to work in the directed case.

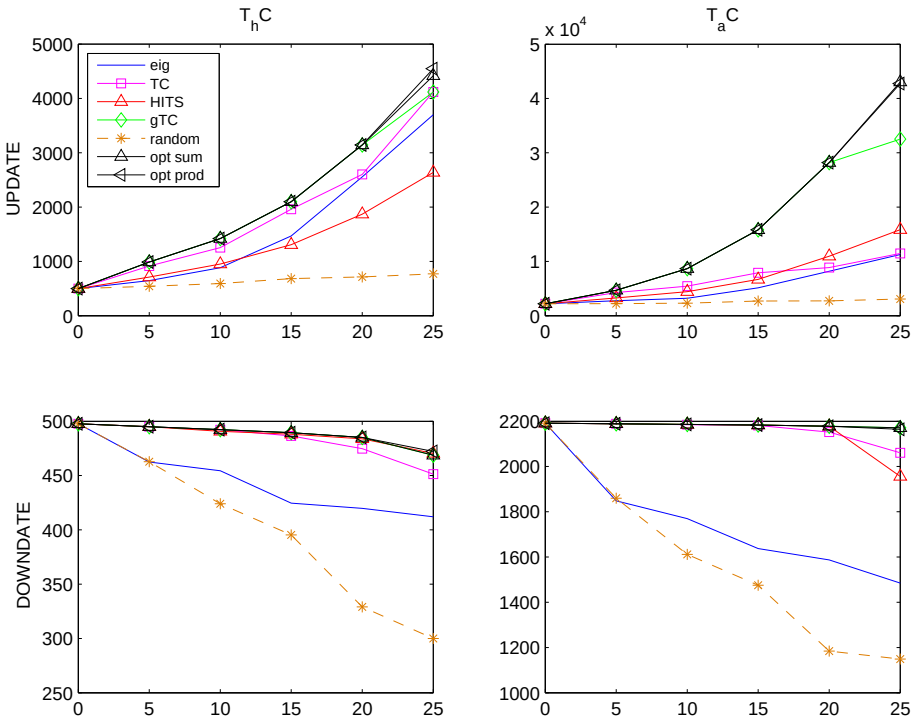
3.4 Numerical tests

In this section we comment on the results obtained from the numerical experiments performed in order to assess the valuability of our heuristics for the directed case.

The numerical tests have been performed on five networks, whose description can be found in Appendix A. Table 3.2 summarizes some properties of the networks in our dataset; namely, it contains the number of nodes n and edges m, the number of virtual edges τ, the two largest singular values of the adjacency matrix σ_1, σ_2, and their difference $\sigma_1 - \sigma_2$. An exception is the network cit-HepTh, for which τ is the number of virtual edges contained in the subgraph of the network constructed as described at the end of Section 3.3.

The small network GD95b is used to compare the effectiveness of the proposed heuristics with a "brute force" approach where each virtual edge is added in turn and the change in total communicability is monitored in order to find the "optimal" choice. Since we are tracking not one but two quantities, $T_hC(A)$ and $T_aC(A)$, we monitor both $T_hC(A) + T_aC(A)$ and $T_hC(A) \cdot T_aC(A)$ and choose the optimal edge for either one of them. These methods are labeled as `opt sum` and `opt prod`, respectively. We perform a similar set of experiments for the downdating. As a baseline method, we also report results for a random selection of the edges in all our tests. The random methods are

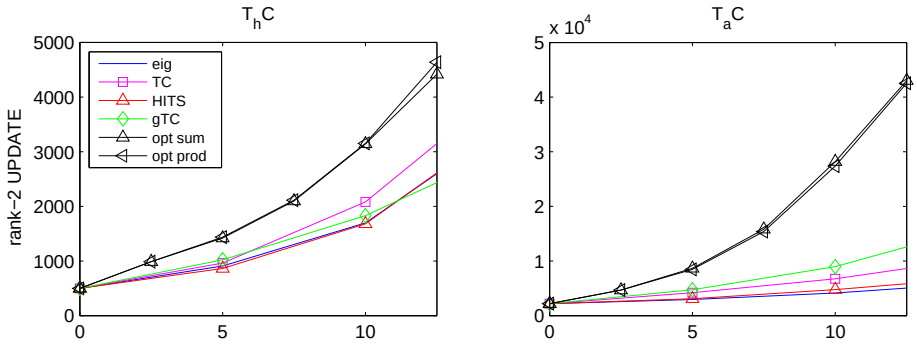
FIGURE 3.2: Evolution of T_hC and T_aC for the network GD95b when 25 edge modifications are performed working on the matrix A associated with the digraph: updates (top) and downdates (bottom).



labeled as `random` or `b:random`, depending on whether we work on the matrix A or on $\mathscr{A}$.

In Figure 3.2 we show plots of the total communicabilities T_hC and T_aC when up to $K = 25$ edge modifications are performed. We limit ourselves to the results for the heuristics based on the original digraph (matrix A). The results show that the heuristic `gTC` performs as well as the "optimal" choice based on brute force, while of course being much less expensive, in tackling both the updating and downdating problem. Note moreover that the performance of the methods `HITS` and `gTC` is different for this network. This result agrees with what one would expect, in view of the small gap $\sigma_1 - \sigma_2$ of the adjacency matrix under study. When considering the problem of downdating, on the other hand, all the methods perform well. In particular we want to stress again the excellent performance of the method `gTC`. The only exception is perhaps the heuristic `eig`, whose performance for the first 5 steps is comparable with the random choice. This result confirms our claim that this heuristic, which was shown to work very well for undirected networks, is not a good approach in the directed case.

FIGURE 3.3: Evolution of T_hC and T_aC for the network GD95b when 25 symmetric edge modifications are performed working on the matrix A associated with the digraph. The optimal methods refer to the rank-one selection of the modifications.



In Figure 3.3 we display the evolution of the total communicability indices under rank-two updates. In this plot we retain the same names for the techniques as used in case of the rank-one modifications; however, the quantities used to derive the rankings are defined as in Subsection 3.3.1. In this figure, each step corresponds to a rank-two symmetric modification, for the heuristic based on the edge centrality measures, and to two rank-one modifications, for the optimal methods. Thus, the plots for the optimal methods coincide with those in Figure 3.2. The results displayed in Figure 3.3 tell us that the symmetric rank-two modifications of the matrix may not lead to results as good as those obtained with the rank-one updates. Indeed, for both the total hub and authority communicabilities we have at least three methods in Figure 3.2 that outperform all the methods used in Figure 3.3. For this reason, we have not further investigate this approach.

The results on the small network give us confidence that at least some of our proposed heuristic for the selection of rank-one modifications of A do a very good job at enhancing the communicability properties of digraphs. In the remaining tests we concentrate on the larger networks, for which the "optimal," brute force approaches are not practical. All experiments were performed using MATLAB Version 8.0.0.783 (R2012b) on an IBM ThinkPad running Ubuntu 14.04 LTS, a 2.5 GHZ Intel Core i5 processor, and 3.7 GiB of RAM.

Figures 3.4-3.7 display the evolution of the total hub and communicability centrality (rescaled by the number of edges in the network) when $K = 200$ updates are selected using the criteria previously introduced. The plots at the top of each figure display the evolution of the total hub communicability (left) and total authority communicability

FIGURE 3.4: Evolution of T_hC and T_aC for the network Computational Complexity when 200 updates are selected working on the matrix A associated with the digraph (top) and on its bipartite version $\mathscr{A}$ (bottom).

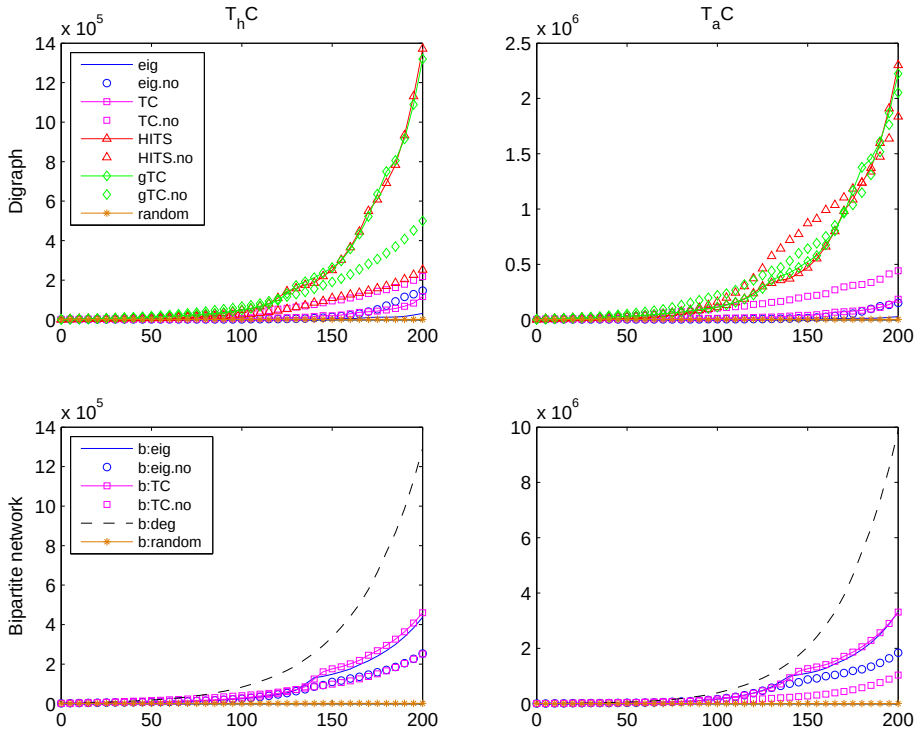


FIGURE 3.5: Evolution of T_hC and T_aC for the network Abortion when 200 updates are selected working on the matrix A associated with the digraph (top) or on its bipartite version $\mathscr{A}$ (bottom).

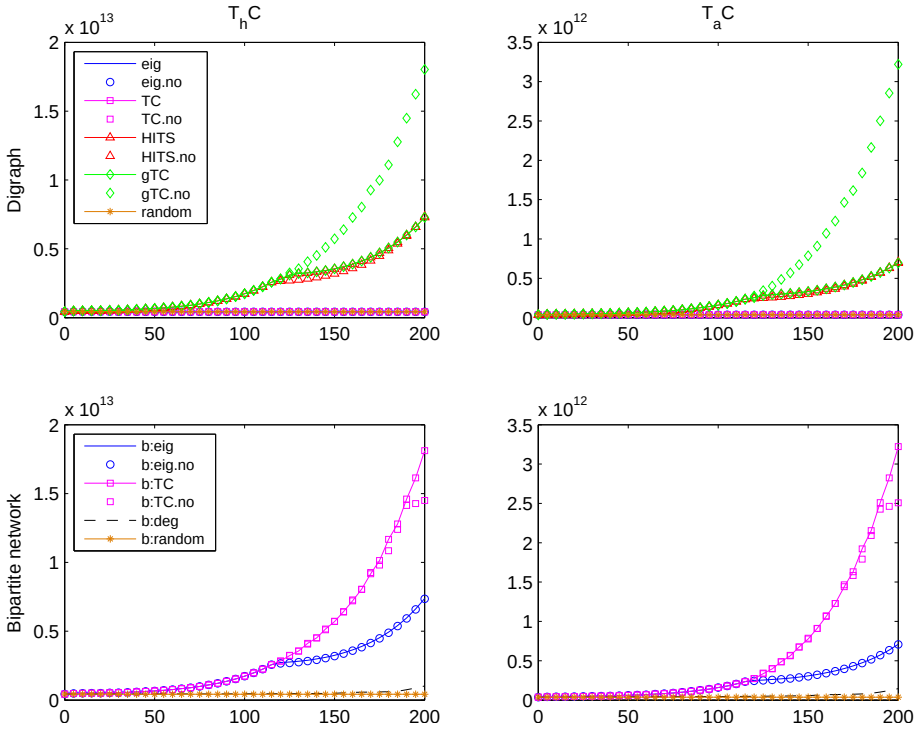


FIGURE 3.6: Evolution of T_hC and T_aC for the network Twitter when 200 updates are selected working on the matrix A associated with the digraph (top) and on its bipartite version $\mathscr{A}$ (bottom).

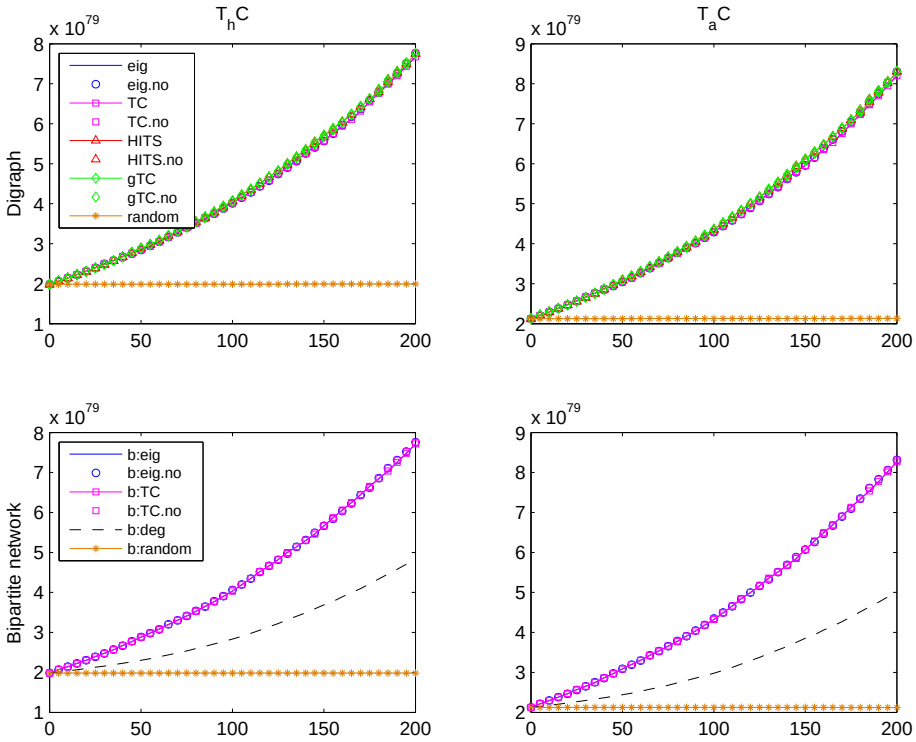


FIGURE 3.7: Evolution of T_hC and T_aC for the network cit-HepTh when 200 updates are selected working on the matrix A associated with the digraph (top) or on its bipartite version $\mathscr{A}$ (bottom).

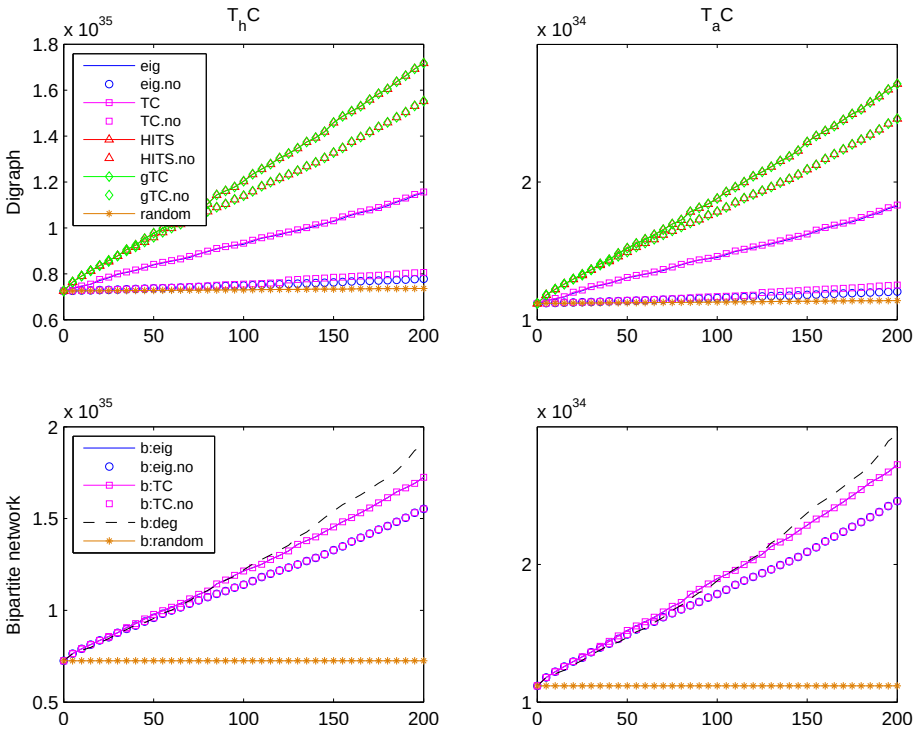


TABLE 3.3: Timings in seconds when $K = 200$ updates are selected for the networks in our Dataset using the methods described.

	Computational Complexity	Abortion	Twitter	cit-HepTh
`eig`	12.51	53.27	139.82	217.12
`eig.no`	0.13	0.73	1.75	1.33
`TC`	114.67	62.22	187.22	163.55
`TC.no`	0.61	0.76	2.19	1.02
`HITS`	8.35	50.69	133.50	88.82
`HITS.no`	0.09	0.63	1.69	0.67
`gTC`	10.63	59.31	183.48	205.17
`gTC.no`	0.12	0.68	1.77	1.28
`b:eig`	9.35	52.43	134.03	99.70
`b:eig.no`	0.21	0.69	1.66	0.88
`b:deg`	11.00	85.06	256.66	84.95
`b:TC`	11.39	59.31	154.97	139.50
`b:TC.no`	0.11	0.72	1.67	0.82

(right) when the digraph is modified using the techniques developed for the directed case. The bottom plots show the evolution of the two indices obtained when the modifications are selected by working on $\mathscr{A}$. As expected, the proposed heuristics are dramatically better than the random choice.

The results show that the heuristics `b:eig(.no)` and `b:TC(.no)` perform similarly to `HITS(.no)` and `gTC(.no)`. The methods `eig(.no)` and `TC(.no)` display erratic behavior and often perform very poorly, as shown in Figures 3.4, 3.5, and 3.7. The method `eig(.no)` also suffers from the restriction that the dominant eigenvalue must be simple, which is not always true in practice. Likewise, the performance of `b:deg` is generally unsatisfactory, with the exception of $T_aC(A)$ for the network Computational Complexity where it outperforms the other techniques (see Figure 3.4). Overall, considering also the timings (see Table 3.3), the best performance is displayed by the heuristics `gTC(.no)` and `HITS(.no)` and by their undirected counterparts `b:TC(.no)` and `b:eig(.no)`. The only possible exception is the Computational Complexity network, for which the heuristics for the directed case outperform those for the undirected, bipartite counterpart.

The disagreement between the results for the heuristics labeled `HITS` and `b:eig` for the network Computational Complexity is at first sight puzzling. The two criteria should lead to the same edge selection and therefore to the same results, since the principal eigenvector of $\mathscr{A}$ is $\mathbf{q}_1 = (\mathbf{u}_1^T, \mathbf{v}_1^T)^T$ and thus $q_1(i) = u_1(i)$ and $q_1(j') = v_1(j)$ in the definition of the heuristic `b:eig`. However, if at least two edges have the same centrality score when working with `b:eig` and `HITS`, then the two methods may select different edges. In this case, after the edge modification has been performed, the adjacency

TABLE 3.4: Timings in seconds when $K = 200$ downdates are selected for the networks in our Dataset using the methods described.

	Computational Complexity	Abortion	Twitter	cit-HepTh
`eig`	5.83	7.77	16.65	201.29
`eig.no`	0.04	0.04	0.05	1.03
`TC`	104.12	15.05	75.35	126.18
`TC.no`	0.92	0.07	0.39	0.73
`HITS`	2.72	4.71	13.32	63.40
`HITS.no`	0.02	0.02	0.08	0.34
`gTC`	5.49	12.06	60.77	175.02
`gTC.no`	0.04	0.08	0.29	0.80
`b:eig`	4.31	6.63	15.10	85.87
`b:eig.no`	0.03	0.04	0.05	1.89
`b:deg`	0.06	0.15	3.94	8.37
`b:TC`	5.51	11.66	39.02	126.44
`b:TC.no`	0.02	0.05	0.16	0.42

matrices manipulated by the two methods are different, thus causing the difference we observe in Figure 3.4. The difference will be more pronounced if the tie between edges occur at the beginning of the modification process.

Table 3.3 contains the timings (in seconds) employed for the selection of the $K = 200$ virtual edges to be updated. The heuristics used were implemented using mostly built-in MATLAB functions, such as the function `eigs` used for computing the largest eigenvalue. For the heuristics requiring the computation of a matrix function times a vector we used the code `funm_kryl` by S. Güttel [52]. To implement the degree-based heuristic we wrote our own code, which is far from optimal when compared to the other ones. The relatively high timings reported for this heuristic can likely be reduced with a more careful implementation. When interpreting the results, it has to be kept in mind that the size τ of the set of virtual edges can be pretty large (cf. Table 3.2). We have observed that, for all the methods, roughly half of the reported computing time is spent in the computation of the products used in the definitions of the edge centrality measures. Nevertheless, the timings range from very small to moderate in all cases, showing the feasibility of the proposed heuristics.

Among all the methods we tested on directed networks for the updating problem, the best performance is displayed by `HITS(.no)`, `gTC(.no)`, `b:eig(.no)`, and `b:TC(.no)` with the methods that manipulate A having the edge when $\sigma_1 - \sigma_2$ is small. Due to its erratic behavior, we cannot recommend the use of `b:deg` in general.

FIGURE 3.8: Evolution of T_hC and T_aC for the network Computational Complexity when 200 downdates are selected working on the matrix A associated with the digraph (top) and on its bipartite version $\mathscr{A}$ (bottom).

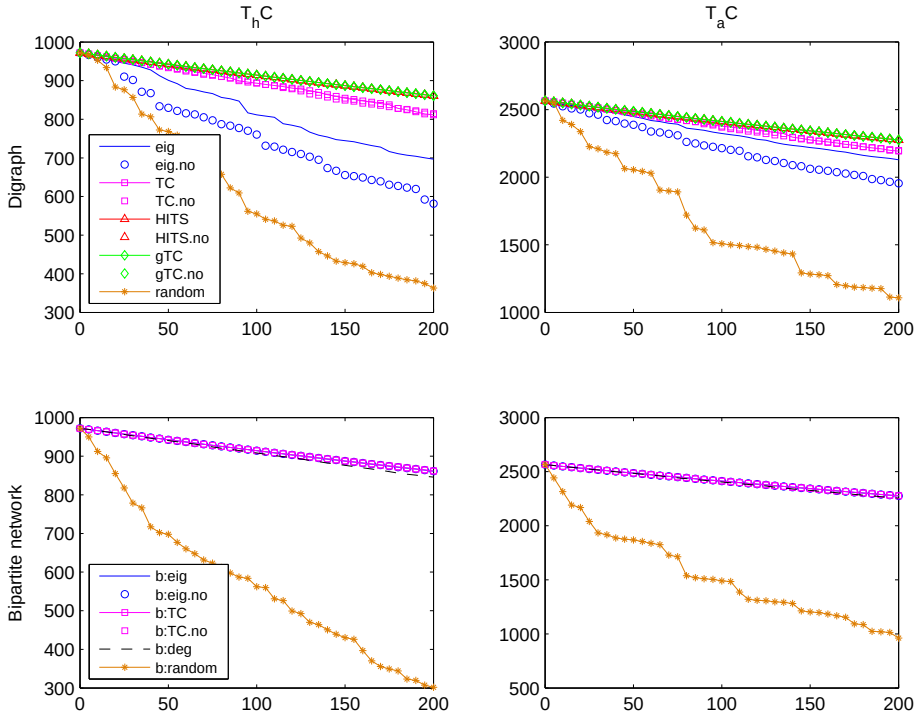


FIGURE 3.9: Evolution of T_hC and T_aC for the network Abortion when 200 downdates are selected working on the matrix A associated with the digraph (top) and on its bipartite version $\mathscr{A}$ (bottom).

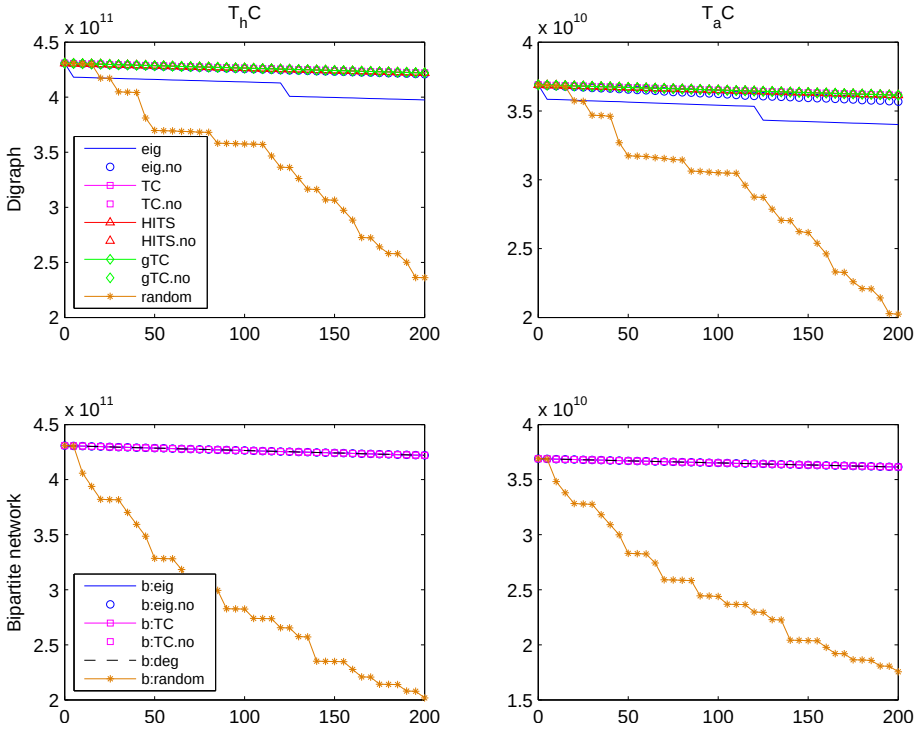


FIGURE 3.10: Evolution of T_hC and T_aC for the network Twitter when 200 downdates are selected working on the matrix A associated with the digraph (top) and on its bipartite version $\mathscr{A}$ (bottom).

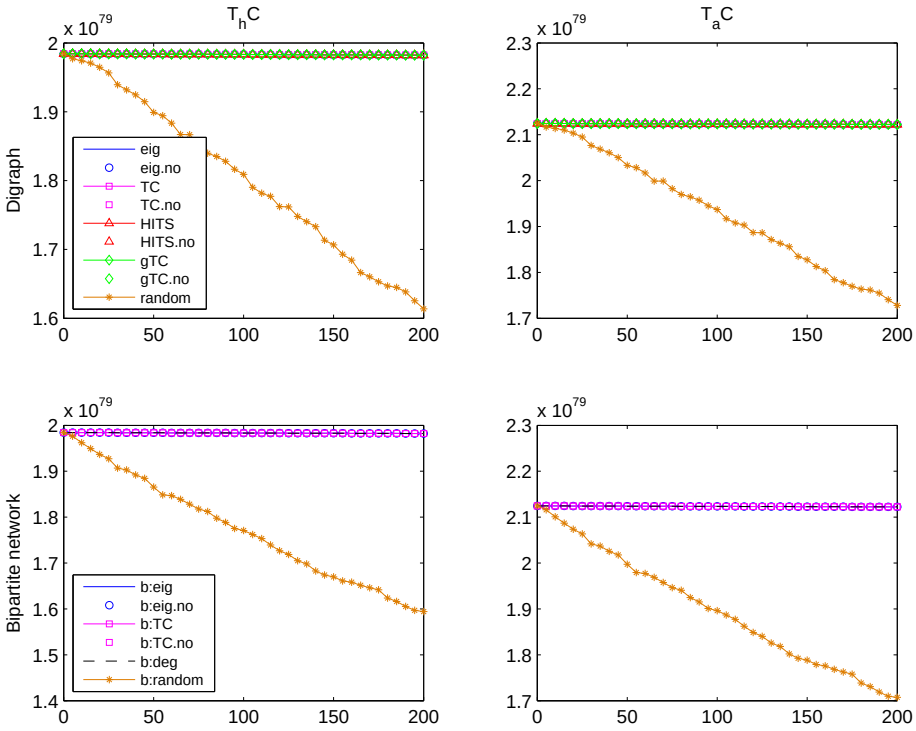
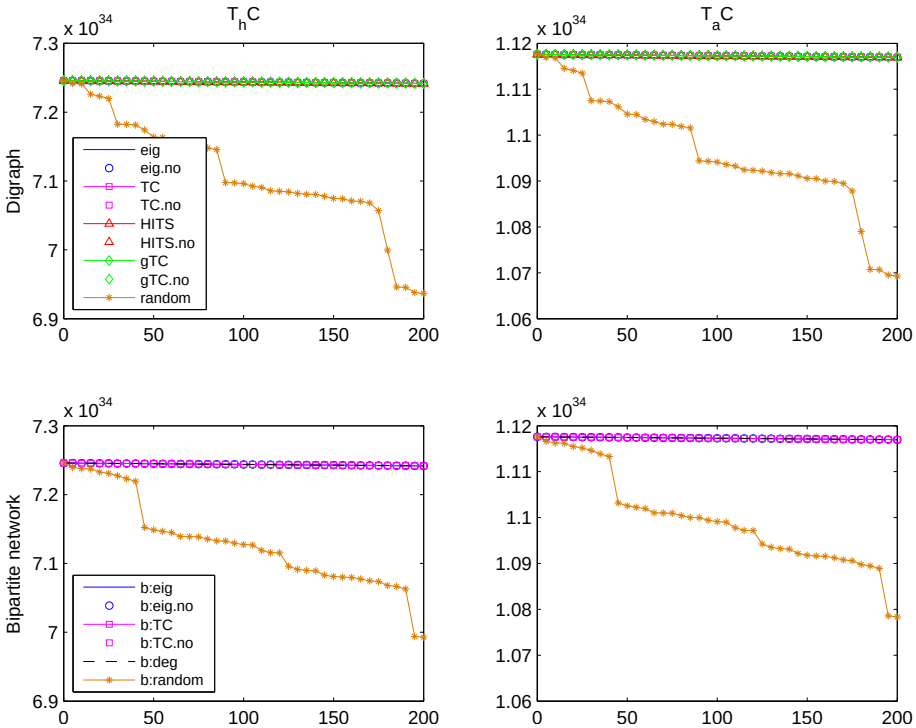


FIGURE 3.11: Evolution of T_hC and T_aC for the network cit-HepTh when 200 downdates are selected working on the matrix A associated with the digraph.



Similar conclusions can be drawn when considering the results for the downdating problem, although the differences among the techniques are less pronounced (Figures 3.8-3.11 and Table 3.4). Indeed, the results shown confirm the effectiveness of the techniques based on the edge HITS and generalized total communicability centralities and of their variants which do not require the recomputation of the rankings. As in the case of the updating problem, the results returned by these two methods essentially reproduce those obtained when working on $\mathscr{A}$ using the heuristics `b:eig(.no)` and `b:TC(.no)`.

The methods `eig(.no)` and `TC(.no)` perform no better (and in some cases worse) than `gTC(.no)` and `HITS(.no)`, while `b:deg` is usually outperformed by `b:eig(.no)` or `b:TC(.no)`.

Concerning the timings, if we compare the results in Tables 3.3 and 3.4 we can see that the values in Table 3.3 are in general higher that those in Table 3.4. This is easily understood in view of what we observed before, if one compares the number of virtual edges τ with the number of edges m in each network in the dataset (see Table 3.2).

While we do not provide a formal assessment of the computational cost of the various heuristics, arguments similar to those found in Chapter 2 indicate that the cost of the more efficient heuristics can be expected to scale approximately like $\mathcal{O}(n)$ or $\mathcal{O}(n \ln n)$ with the number of nodes n.

In conclusion, by considering the overall performance of the methods and their cost (in terms of timings), we find that the best criteria for our updating/downdating goals are the methods `HITS(.no)` and `gTC(.no)`. Besides these, satisfactory results may also be obtained using `b:eig(.no)` or `b:TC(.no)`. From the timings in Tables 3.3 and 3.4 we can deduce that the heuristics `HITS(.no)` are in general slightly faster than `b:eig(.no)` and may thus be preferred. Concerning whether it is better to use `gTC(.no)` or `b:TC(.no)`, we anticipate that the first will be preferable when used in conjunction with fast algorithms for the approximation of bilinear forms involving generalized matrix functions.

Conclusions

In this chapter we have extended the notion of total network f-communicability to the case of directed graphs. Moreover, we have developed heuristics for manipulating an existing directed network so as to tune its communicability properties. In doing so we made use of the concept of alternating walks, which allows us to take into account the dual role played by each node in a digraph, namely, receiver and broadcaster of information.

Our computational results indicate that the heuristics which take into account the dual role of nodes in directed networks tend to be preferable to those that do not. We also showed that these heuristics are very fast in practice.

Chapter 4

Computation of generalized matrix functions

Generalized matrix functions were first introduced by Hawkins and Ben-Israel in [55] in order to extend the notion of a matrix function to rectangular matrices. Essentially, the definition is based on replacing the spectral decomposition of A (or the Jordan canonical form, if A is not diagonalizable) with the singular value decomposition, and evaluating the function at the singular values of A, if defined. The paper appears to have gone largely unnoticed, despite increasing interest in matrix functions in the numerical linear algebra community over the past several years. While it is likely that the perceived scarcity of applications is to blame (at least in part) for this lack of attention, it turns out that generalized matrix functions do have interesting applications and have actually occurred in the literature without being recognized as such; see Chapter 3 and Section 4.2 for some examples.

In this chapter we revisit the topic of generalized matrix functions, with an emphasis on numerical aspects. After reviewing some properties of generalized matrix functions and proving some new results, we develop several computational approaches based on variants of Golub–Kahan bidiagonalization to compute or estimate bilinear forms involving generalized matrix functions, including entries of the generalized matrix function itself and the action of a generalized matrix function on a vector.

The content of this chapter is based on the results presented in [6].

4.1 Properties

In this section we review some properties of generalized matrix functions and we summarize a few new results. Recall that, given a rank-r matrix $A \in \mathbb{C}^{m\times n}$ and given a CSVD of the matrix $A = U_r\Sigma_r V_r^*$, then the generalized matrix function induced by the scalar function $f : \mathbb{R} \to \mathbb{R}$ is defined as

$$f^\diamond(A) = U_r f(\Sigma_r)V_r^* = U_r \mathrm{diag}(f(\sigma_1), f(\sigma_2), \ldots, f(\sigma_r))V_r^*.$$

Letting $E_i = \mathbf{u}_i\mathbf{v}_i^*$ and $E = \sum_{i=1}^r E_i$, we can write

$$A = \sum_{i=1}^r \sigma_i \mathbf{u}_i\mathbf{v}_i^* = \sum_{i=1}^r \sigma_i E_i,$$

and thus it follows that

$$f^\diamond(A) = \sum_{i=1}^r f(\sigma_i)\mathbf{u}_i\mathbf{v}_i^* = \sum_{i=1}^r f(\sigma_i)E_i. \tag{4.1}$$

Proposition 4.1. (Sums and products of functions [55]). *Let $f, g, h : \mathbb{R} \to \mathbb{R}$ be scalar functions and let $f^\diamond, g^\diamond, h^\diamond : \mathbb{C}^{m\times n} \to \mathbb{C}^{m\times n}$ be the corresponding generalized matrix functions. Then:*

(i) if $f(z) = k$, then $f^\diamond(A) = kE$;

(ii) if $f(z) = z$, then $f^\diamond(A) = A$;

(iii) if $f(z) = g(z) + h(z)$, then $f^\diamond(A) = g^\diamond(A) + h^\diamond(A)$;

(iv) if $f(z) = g(z)h(z)$, then $f^\diamond(A) = g^\diamond(A)E^*h^\diamond(A)$.

In the following we prove a few properties of generalized matrix functions.

Proposition 4.2. *Let $A = U_r\Sigma_r V_r^* \in \mathbb{C}^{m\times n}$ be a matrix of rank r. Let $f : \mathbb{R} \to \mathbb{R}$ be a scalar function and let $f^\diamond : \mathbb{C}^{m\times n} \to \mathbb{C}^{m\times n}$ be the induced generalized matrix function, assumed to be defined at A. Then the following properties hold true.*

(i) $[f^\diamond(A)]^* = f^\diamond(A^*)$;

(ii) *let* $X \in \mathbb{C}^{m\times m}$ *and* $Y \in \mathbb{C}^{n\times n}$ *be two unitary matrices, then* $f^\diamond(XAY) = X[f^\diamond(A)]Y$*;*

(iii) *Let* $P \in \mathbb{R}^{r\times r}$ *be a permutation matrix and let* $U_r = \mathscr{U}_r P$*,* $V_r = \mathscr{V}_r P$*, and* $\Sigma_r = P^T \mathscr{D}_r P$*. Then*

$$f^\diamond(A) = \mathscr{U}_r f(\mathscr{D}_r)\mathscr{V}_r^*.$$

(iv) *if* $A = \operatorname{diag}(A_{11}, A_{22}, \ldots, A_{kk})$*, then*

$$f^\diamond(A) = \operatorname{diag}(f^\diamond(A_{11}), f^\diamond(A_{22}), \ldots, f^\diamond(A_{kk}));$$

(v) $f^\diamond(I_k \otimes A) = I_k \otimes f^\diamond(A)$*, where* I_k *is the* $k \times k$ *identity matrix and* $\otimes$ *is the Kronecker product;*

(vi) $f^\diamond(A \otimes I_k) = f^\diamond(A) \otimes I_k$*.*

Proof. (i) From (1.3) it follows that $A^* = V_r \Sigma_r U_r^*$, and thus

$$f^\diamond(A^*) = V_r f(\Sigma_r) U_r^* = [U_r f(\Sigma_r) V_r^*]^* = [f^\diamond(A)]^*.$$

(ii) The result follows from the fact that unitary matrices form a group under multiplication and that the rank of a matrix does not change under left or right multiplication by a nonsingular matrix [57, p. 13]. Indeed, the matrix $B := XAY$ has rank r and thus

$$\begin{aligned} f^\diamond(B) &= f^\diamond(XU\Sigma V^* Y) = (XU)_r f(\Sigma_r)[(Y^*V)_r]^* \\ &= XU_r f(\Sigma_r) V_r^* Y = X f^\diamond(A) Y. \end{aligned}$$

where $(XU)_r$ and $(V^*Y)_r$ are the matrices containing the first r columns of (XU) and (V^*Y), respectively.

(iii) From a basic property of matrix functions [56, Theorem 1.13] it follows that:

$$\mathscr{U}_r f(\mathscr{D}_r)\mathscr{V}_r^* = U_r P^T P f(\Sigma_r) P^T P V_r^* = U_r f(\Sigma_r) V_r^* = f^\diamond(A).$$

(iv) Let $A_{ii} = U_{i,r_i}\Sigma_{i,r_i}V_{i,r_i}^*$ be the CSVD of the rank-r_i matrix A_{ii} for $i = 1, 2, \ldots, k$. Then $A = \mathscr{U}_r\mathscr{D}_r\mathscr{V}_r^*$, where

$$\begin{aligned}\mathscr{D}_r &:= \operatorname{diag}\left(\Sigma_{1,r_1},\ \Sigma_{2,r_2}, \ldots,\ \Sigma_{k,r_k}\right),\\ \mathscr{U}_r &:= \operatorname{diag}\left(U_{1,r_1},\ U_{2,r_2}, \ldots,\ U_{k,r_k}\right),\\ \mathscr{V}_r &:= \operatorname{diag}\left(V_{1,r_1},\ V_{2,r_2}, \ldots,\ V_{k,r_k}\right).\end{aligned}$$

From (iii) it follows that $f^\diamond(A) = \mathscr{U}_r f(\mathscr{D}_r)\mathscr{V}_r^*$, and thus

$$\begin{aligned}f^\diamond(A) &= \mathscr{U}_r f(\mathscr{D}_r)\mathscr{V}_r^*\\ &= \operatorname{diag}\left(U_{1,r_1}f(\Sigma_{1,r_1})V_{1,r_1}^*,\ U_{2,r_2}f(\Sigma_{2,r_2})V_{2,r_2}^*, \ldots,\ U_{k,r_k}f(\Sigma_{k,r_k})V_{k,r_k}^*\right)\\ &= \operatorname{diag}\left(f^\diamond(A_{11}), f^\diamond(A_{22}), \ldots, f^\diamond(A_{kk})\right).\end{aligned}$$

(v) The result follows from (iv) and the fact that $I_k \otimes A = \operatorname{diag}(A, A, \ldots, A)$ is a $km \times kn$ diagonal block matrix with k copies of A on the main diagonal.

(vi) It follows from (v) and from the fact that for two general matrices $A \in \mathbb{C}^{m\times n}$ and $B \in \mathbb{C}^{p\times q}$, there exist two permutation matrices $K^{(p,m)}$ and $K^{(n,q)}$ called *commutation matrices* such that $K^{(p,m)}\left(A \otimes B\right)K^{(n,q)} = B \otimes A$ (see [68, Chapter 3]).

□

The following theorem provides a result for the composition of two functions.

Proposition 4.3. (Composite functions) *Let $A = U_r\Sigma_r V_r^* \in \mathbb{C}^{m\times n}$ be a rank-r matrix and let $\{\sigma_i : 1 \le i \le r\}$ be its singular values. Assume that $h : \mathbb{R} \to \mathbb{R}$ and $g : \mathbb{R} \to \mathbb{R}$ are two scalar functions such that $h(\sigma_i) \neq 0$ and $g(h(\sigma_i))$ exist for all $i = 1, 2, \ldots, r$. Let $g^\diamond : \mathbb{C}^{m\times n} \to \mathbb{C}^{m\times n}$ and $h^\diamond : \mathbb{C}^{m\times n} \to \mathbb{C}^{m\times n}$ be the induced generalized matrix functions. Moreover, let $f : \mathbb{R} \to \mathbb{R}$ be the composite function $f = g \circ h$. Then the induced matrix function $f^\diamond : \mathbb{C}^{m\times n} \to \mathbb{C}^{m\times n}$ satisfies*

$$f^\diamond(A) = g^\diamond(h^\diamond(A)).$$

Proof. Let $B := h^\diamond(A) = U_r h(\Sigma_r) V_r^* = U_r \Theta_r V_r^*$. Since $h(\sigma_i) \neq 0$ for all $i = 1, 2, \ldots, r$, this matrix has rank r. Using (iii) in Proposition 4.2 it thus follows that

$$g^\diamond(h^\diamond(A)) = g^\diamond(B) = U_r g(\Theta_r) V_r^* = U_r g(h(\Sigma_r)) V_r^* = U_r f(\Sigma_r) V_r^* = f^\diamond(A).$$

□

The following result describes the relationship between standard matrix functions and generalized matrix functions.

Theorem 4.4. *Let $A \in \mathbb{C}^{m\times n}$ be a rank-r matrix and let $f : \mathbb{R} \to \mathbb{R}$ be a scalar function. Let $f^\diamond : \mathbb{C}^{m\times n} \to \mathbb{C}^{m\times n}$ be the induced generalized matrix function. Then*

$$f^\diamond(A) = \left(\sum_{i=1}^{r} \frac{f(\sigma_i)}{\sigma_i} \mathbf{u}_i \mathbf{u}_i^* \right) A = A \left(\sum_{i=1}^{r} \frac{f(\sigma_i)}{\sigma_i} \mathbf{v}_i \mathbf{v}_i^* \right), \tag{4.2a}$$

or, equivalently,

$$f^\diamond(A) = f(\sqrt{AA^*})(\sqrt{AA^*})^\dagger A = A(\sqrt{A^*A})^\dagger f(\sqrt{A^*A}). \tag{4.2b}$$

Proof. The two identities are an easy consequence of the fact that $\mathbf{u}_i = \frac{1}{\sigma_i} A\mathbf{v}_i$ and $\mathbf{v}_i = \frac{1}{\sigma_i} A^*\mathbf{u}_i$ for $i = 1, 2, \ldots, r$. □

Remark 4.5. The identities $(\sqrt{AA^*})^\dagger A = A(\sqrt{A^*A})^\dagger = E = U_r V_r^*$ hold, and these prove the necessity of condition $f(0) = 0$ in Remark 1.18.

Proposition 4.6. *Let $A \in \mathbb{C}^{m\times n}$ be a rank-r matrix and let $f : \mathbb{R} \to \mathbb{R}$ and $g : \mathbb{R} \to \mathbb{R}$ be two scalar functions such that $f^\diamond(A)$ and $g(AA^*)$ are defined. Then*

$$g(AA^*) f^\diamond(A) = f^\diamond(A) g(A^*A).$$

Proof. From $A = U_r \Sigma_r V_r^*$ it follows $AA^* = U_r \Sigma_r^2 U_r^*$ and $A^*A = V_r \Sigma_r^2 V_r^*$; thus

$$\begin{aligned} g(AA^*) f^\diamond(A) &= U_r g(\Sigma_r^2) U_r^* U_r f(\Sigma_r) V_r^* = U_r g(\Sigma_r^2) f(\Sigma_r) V_r^* \\ &= U_r f(\Sigma_r) g(\Sigma_r^2) V_r^* = U_r f(\Sigma_r) V_r^* V_r g(\Sigma_r^2) V_r^* \\ &= f^\diamond(A) g(A^*A). \end{aligned}$$

□

4.2 Manifestations of generalized matrix functions

As mentioned in the introduction to this chapter, generalized matrix functions (in the sense of Hawkins and Ben-Israel) have appeared in the literature without being recognized as such. Here we discuss a few examples that we are aware of. No doubt there have been other such instances.

In [28], the authors address the problem of computing functions of real skew-symmetric matrices, in particular the evaluation of the product $e^A b$ for a given skew-symmetric matrix A and vector b using the Lanczos algorithm. The authors observe that any $A \in \mathbb{R}^{2n\times 2n}$ with $A^T = -A$ is orthogonally similar to a matrix of the form

$$\begin{pmatrix} 0 & -B \\ B^T & 0 \end{pmatrix},$$

where B is lower bidiagonal of order n. As a consequence, if $B = U\Sigma V^T$ is an SVD of B, the matrix exponential e^A is orthogonally similar to the matrix

$$\begin{pmatrix} U\cos(\Sigma)U^T & -U\sin(\Sigma)V^T \\ V\sin(\Sigma)U^T & V\cos(\Sigma)V^T \end{pmatrix}, \tag{4.3}$$

where the matrix in the upper right block is precisely $-\sin^\diamond(B)$. The authors of [28] develop computational techniques for the matrix exponential based on (4.3). We also mention that in the same paper the authors derive a similar expression, also found in [11], for the exponential of the symmetric matrix $\mathscr{A}$ given in (1.1). These expressions are extended to more general matrix functions in [64], where they are used to investigate the off-diagonal decay of analytic functions of large, sparse, skew-symmetric matrices. Furthermore, in [29] it is shown how these ideas can be used to develop efficient geometrical integrators for the numerical solution of certain Hamiltonian differential systems.

In [25], the authors consider the problem of detecting (approximate) directed bipartite communities in directed graphs. Consideration of alternating walks in the underlying graph leads them to introducing a "non-standard matrix function" of the form

$$f(A) = I - A + \frac{AA^T}{2!} - \frac{AA^TA}{3!} + \frac{AA^TAA^T}{4!} - \cdots,$$

where A is the adjacency matrix of the graph. Using $A = U\Sigma V^T$ this expression is readily recognized to be equivalent to

$$f(A) = U\cosh(\Sigma)U^T - U\sinh(\Sigma)V^T,$$

which is a "mixture" of the standard matrix function $\cosh(\sqrt{AA^T})$ and the generalized matrix function $\sinh^\diamond(A)$.

As mentioned, generalized hyperbolic matrix functions were also considered in [11] in the context of directed networks, also based on the notion of alternating walks in directed graphs. In Chapter 3, the action of generalized matrix functions on a vector of all ones was used to define certain centrality measures for nodes in directed graphs; Chapter 3 is based on [5], where the connection with the work of Hawkins and Ben-Israel was explicitly made.

Finally, we mention that generalized matrix functions arise when *filter factors* are used to regularize discrete ill-posed problems; see, e.g., [53].

4.3 Computational methods

The computation of the generalized matrix functions defined as in Definition 1.17 requires the knowledge of the singular value decomposition of A. When m and n are large, computing the SVD may be unfeasible. Moreover, in most applications it is not required to compute the whole matrix $f^\diamond(A)$; rather, the goal is often to estimate quantities of the form

$$Z^T f^\diamond(A)W, \qquad Z \in \mathbb{R}^{m\times k},\ W \in \mathbb{R}^{n\times k}, \tag{4.4}$$

or to compute the action of the generalized matrix function on a set of k vectors, i.e., to evaluate $f^\diamond(A)W$, usually with $k \ll \min\{m, n\}$. For example, computing selected columns of $f^\diamond(A)$ reduces to the evaluation of $f^\diamond(A)W$ where W consists of the corresponding columns of the identity matrix I_n, and computing selected entries of $f^\diamond(A)$ requires evaluating $Z^T f^\diamond(A)W$ where Z contains selected columns of the identity matrix I_m.

The problem of estimating or giving bounds on such quantities can be tackled, following [49], by using Gauss-type quadrature rules. In this work we will analyze the case $k = 1$; the case of $k > 1$ is discussed in [6].

It is known that in certain cases Gauss-type quadrature rules can be used to obtain lower and upper bounds on bilinear forms like $\mathbf{z}^T f(A)\mathbf{w}$, where $f(A)$ is a (standard) matrix function and $A = A^T$ (cf. Section 1.3). This is the case when f enjoys certain monotonicity properties. If f is completely monotonic (c.m.) on an interval containing the spectrum of $A = A^T$, then one can obtain lower and upper bounds on quadratic forms of the type $\mathbf{u}^T f(A)\mathbf{u}$ and from these lower and upper bounds on bilinear forms like $\mathbf{z}^T f(A)\mathbf{w}$ with $\mathbf{z} \neq \mathbf{w}$. For a general f, on the other hand, Gaussian quadrature can only provide estimates of these quantities.

Similarly, in order to obtain bounds (rather than mere estimates) for bilinear expressions involving generalized matrix functions, we need the scalar functions involved in the computations to be completely monotonic.

Remark 4.7. We will be applying our functions to diagonal matrices that contain the singular values of the matrix of interest. Thus, in our framework, the interval on which we want to study the complete monotonicity of the functions is $I = (0, \infty)$.

We briefly recall here a few properties of c. m. functions; see, e.g., [74, 98] and references therein for systematic treatments of complete monotonicity.

Lemma 4.8. *If f_1 and f_2 are completely monotonic functions on I, then*

(i) $\alpha f_1(t) + \beta f_2(t)$ with $\alpha, \beta \geq 0$ is completely monotonic on I;

(ii) $f_1(t) f_2(t)$ is completely monotonic on I.

Lemma 4.9. [74, Theorem 2] *Let f_1 be completely monotonic and let f_2 be a nonnegative function such that f_2' is completely monotonic. Then $f_1 \circ f_2$ is completely monotonic.*

Using these lemmas, we can prove the following useful result.

Theorem 4.10. *If f is completely monotonic on $(0, \infty)$, then $g(t) := \frac{f(\sqrt{t})}{\sqrt{t}}$ is completely monotonic on $(0, \infty)$.*

Proof. Let $h(t) = t^{-1}$; then by Lemma 4.8 (ii) we know that $g(t)$ is completely monotonic on $I = (0, \infty)$ if both $f(\sqrt{t})$ and $h(\sqrt{t})$ are completely monotonic on I. The function $\sqrt{t}$ is positive on the interval $(0, \infty)$; moreover, it is such that its first derivative $\frac{1}{2}t^{-1/2}$ is completely monotonic on I. Therefore, from Lemma 4.9 it follows that if f is c.m. , then $f(\sqrt{t})$ is. Similarly, since $h(t) = t^{-1}$ is completely monotonic, $h(\sqrt{t})$ is completely monotonic. This concludes the proof. □

In the following, we propose three different approaches to approximate the bilinear forms of interest. The first approach exploits the results of Theorem 4.4 to describe $\mathbf{z}^T f^\diamond(A)\mathbf{w}$ as a bilinear form that involves standard matrix functions of a tridiagonal matrix. The second approach works directly with the generalized matrix function and the Moore–Penrose pseudo-inverse of a bidiagonal matrix. The third approach first approximates the action of a generalized matrix function on a vector and then derives the approximation for the bilinear form of interest.

4.3.1 First approach

When the function $f : \mathbb{R} \longrightarrow \mathbb{R}$ that defines $f^\diamond$ is c.m., then Gauss-type quadrature rules can be used to derive upper and lower bounds for the quantities of interest. It is straightforward to see by using (4.2a) that a bilinear form involving a generalized matrix function can be written as

$$\mathbf{z}^T f^\diamond(A)\mathbf{w} = \mathbf{z}^T \left(\sum_{i=1}^{r} \frac{f(\sigma_i)}{\sigma_i} \mathbf{u}_i \mathbf{u}_i^T \right) \widetilde{\mathbf{w}} = \widetilde{\mathbf{z}}^T \left(\sum_{i=1}^{r} \frac{f(\sigma_i)}{\sigma_i} \mathbf{v}_i \mathbf{v}_i^T \right) \mathbf{w},$$

where $\widetilde{\mathbf{w}} = A\mathbf{w}$ and $\widetilde{\mathbf{z}} = A^T\mathbf{z}$. Using the equalities in (4.2b) one can see that these quantities can also be expressed as bilinear forms involving functions of the matrices AA^T and A^TA, respectively. More specifically, one obtains

$$\mathbf{z}^T f^\diamond(A)\mathbf{w} = \widetilde{\mathbf{z}}^T \left(\sum_{i=1}^{r} \frac{f(\sigma_i)}{\sigma_i} \mathbf{v}_i \mathbf{v}_i^T \right) \mathbf{w} = \widetilde{\mathbf{z}}^T g(A^TA)\mathbf{w}, \tag{4.5a}$$

$$\mathbf{z}^T f^\diamond(A)\mathbf{w} = \mathbf{z}^T \left(\sum_{i=1}^{r} \frac{f(\sigma_i)}{\sigma_i} \mathbf{u}_i \mathbf{u}_i^T \right) \widetilde{\mathbf{w}} = \mathbf{z}^T g(AA^T)\widetilde{\mathbf{w}}, \tag{4.5b}$$

where in both cases $g(t) = (\sqrt{t})^{-1} f(\sqrt{t})$.

In the following we focus on the case described by (4.5a). The discussion for the case described by (4.5b) follows the same lines.

Remark 4.11. Note that if $\mathbf{z}, \mathbf{w}$ are vectors such that $\widetilde{\mathbf{z}} \neq \mathbf{w}$, then we can use the *polarization identity* [49, p. 114]:

$$\widetilde{\mathbf{z}}^T g(A^T A)\mathbf{w} = \frac{1}{4}\left[(\widetilde{\mathbf{z}}+\mathbf{w})^T g(A^T A)(\widetilde{\mathbf{z}}+\mathbf{w}) - (\widetilde{\mathbf{z}}-\mathbf{w})^T g(A^T A)(\widetilde{\mathbf{z}}-\mathbf{w})\right]$$

to reduce the evaluation of the bilinear form of interest to the evaluation of two symmetric bilinear forms. For this reason, the theoretical description of the procedure to follow will be carried out only for the case $\widetilde{\mathbf{z}} = \mathbf{w}$.

Let $\widetilde{\mathbf{z}} = \mathbf{w}$ be a unit vector (i.e., $\|\mathbf{w}\|_2 = 1$). We can rewrite the quantity (4.5a) as a Riemann–Stieltjes integral by substituting the spectral factorization of $A^T A$:

$$\mathbf{w}^T g(A^T A)\mathbf{w} = \mathbf{w}^T V_r g(\Sigma_r^2) V_r^T \mathbf{w} = \sum_{i=1}^{r} \frac{f(\sigma_i)}{\sigma_i}(\mathbf{v}_i^T \mathbf{w})^2 = \int_{\sigma_r^2}^{\sigma_1^2} g(t)\,\mathrm{d}\alpha(t), \tag{4.6}$$

where $\alpha(t)$ is a piecewise constant step function with jumps at the positive eigenvalues $\{\sigma_i^2\}_{i=1}^r$ of $A^T A$ defined as follows:

$$\alpha(t) = \begin{cases} 0, & \text{if } t < \sigma_r^2 \\ \sum_{i=j+1}^{r} (\mathbf{v}_i^T \mathbf{w})^2, & \text{if } \sigma_{j+1}^2 \leq t < \sigma_j^2 \\ \sum_{i=1}^{r} (\mathbf{v}_i^T \mathbf{w})^2, & \text{if } t \geq \sigma_r^2. \end{cases}$$

We use partial Golub–Kahan bidiagonalization [48, 50] of the matrix A to find upper and lower bounds for the bilinear form described in (4.6). After ℓ steps, the Golub–Kahan bidiagonalization of the matrix A with initial vector $\mathbf{w}$ yields the decompositions

$$AQ_\ell = P_\ell B_\ell, \qquad A^T P_\ell = Q_\ell B_\ell^T + \gamma_\ell \mathbf{q}_\ell \mathbf{e}_\ell^T, \tag{4.7}$$

where the matrices $Q_\ell = [\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_{\ell-1}] \in \mathbb{R}^{n\times\ell}$ and $P_\ell = [\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_{\ell-1}] \in \mathbb{R}^{m\times\ell}$ have orthonormal columns, the matrix

$$B_\ell = \begin{pmatrix} \omega_1 & \gamma_1 & & \\ & \ddots & \ddots & \\ & & \omega_{\ell-1} & \gamma_{\ell-1} \\ & & & \omega_\ell \end{pmatrix} \in \mathbb{R}^{\ell\times\ell}$$

is upper bidiagonal, and the first column of Q_ℓ is $\mathbf{w}$.

Remark 4.12. All the $\{\gamma_j\}_{j=1}^{\ell-1}$ and $\{\omega_j\}_{j=1}^{\ell}$ can be assumed to be nonzero [48]. With this assumption, the CSVD of the bidiagonal matrix B_ℓ coincides with its SVD:

$$B_\ell = \mathcal{U}_\ell \Theta_\ell \mathcal{V}_\ell^T,$$

where $\mathcal{U}_\ell = [\boldsymbol{\upsilon}_1, \boldsymbol{\upsilon}_2, \ldots, \boldsymbol{\upsilon}_\ell] \in \mathbb{R}^{\ell\times\ell}$ and $\mathcal{V}_\ell = [\boldsymbol{\nu}_1, \boldsymbol{\nu}_2, \ldots, \boldsymbol{\nu}_\ell] \in \mathbb{R}^{\ell\times\ell}$ are orthogonal, and $\Theta_\ell = \mathrm{diag}(\theta_1, \theta_2, \ldots, \theta_\ell) \in \mathbb{R}^{\ell\times\ell}$.

Combining the equations in (4.7) leads to

$$A^T A Q_\ell = Q_\ell B_\ell^T B_\ell + \gamma_\ell \omega_\ell \mathbf{q}_\ell \mathbf{e}_\ell^T,$$

where $\mathbf{q}_\ell$ denotes the Lanczos vector computed at iteration $\ell + 1$. The matrix

$$T_\ell = B_\ell^T B_\ell$$

is thus symmetric and tridiagonal and coincides (in exact arithmetic) with the matrix obtained when the Lanczos algorithm is applied to $A^T A$.

The quadratic form in (4.6) can then be approximated by using an ℓ-point Gauss quadrature rule [49, Theorem 6.6]:

$$\mathcal{G}_\ell := \mathbf{e}_1^T g(T_\ell)\mathbf{e}_1 = \mathbf{e}_1^T (\sqrt{T_\ell})^\dagger f(\sqrt{T_\ell})\mathbf{e}_1. \tag{4.8}$$

If the function $f(t)$ is c.m., then the Gauss rule provides a lower bound for (4.6), which

can be shown to be strictly increasing with ℓ. If the recursion formulas for the Golub–Kahan bidiagonalization break down, that is, if $\gamma_\ell = 0$ at step ℓ, then the Gauss quadrature rule gives the exact value (see [50, p. 490ff]).

The following result can be easily derived from (4.8).

Proposition 4.13. *Let $A \in \mathbb{R}^{m\times n}$ and let $B_\ell \in \mathbb{R}^{\ell\times\ell}$ be the bidiagonal matrix computed after ℓ steps of the Golub–Kahan bidiagonalization algorithm. Let $(\theta_i, \boldsymbol{\upsilon}_i, \boldsymbol{\nu}_i)$ for $i = 1, 2, \ldots, \ell$ be the singular triplets of $B_\ell = \mathcal{U}_\ell \Theta_\ell \mathcal{V}_\ell^T$. Then the nodes of the ℓ-point Gauss quadrature rule $\mathcal{G}_\ell$ are the singular values $\{\theta_i\}_{i=1}^{\ell}$. Furthermore, if $\widetilde{\mathbf{z}} = \mathbf{w}$, then the weights of the rule are given by $(\mathbf{e}_1^T \boldsymbol{\nu}_i)^2 \theta_i^{-1}$.*

Similarly, if $\mathbf{z} = \widetilde{\mathbf{w}}$, the weights of $\mathcal{G}_\ell$ are $(\mathbf{e}_1^T \boldsymbol{\upsilon}_i)^2 \theta_i^{-1}$ for $i = 1, 2, \ldots, \ell$.

To provide an upper bound for (4.6) when f is c. m., one can use a $(\ell+1)$-point Gauss–Radau quadrature rule with a fixed node $\tau = \sigma_1^2$; this can be expressed in terms of the entries of the symmetric tridiagonal matrix

$$\widehat{T}_{\ell+1} = \begin{pmatrix} T_\ell & \rho_\ell \mathbf{e}_\ell \\ \rho_\ell \mathbf{e}_\ell^T & \widehat{\omega}_{\ell+1} \end{pmatrix} \in \mathbb{R}^{(\ell+1)\times(\ell+1)}$$

as $\widehat{\mathcal{G}}_{\ell+1} := \mathbf{e}_1^T g(\widehat{T}_{\ell+1})\mathbf{e}_1$, where $g(t) = (\sqrt{t})^{-1} f(\sqrt{t})$. The entries of this matrix, except for the last diagonal entry, are those of $B_{\ell+1}^T B_{\ell+1}$. To compute the last diagonal entry so that $\widehat{T}_{\ell+1}$ has $\tau = \sigma_1^2$ among its eigenvalues, we proceeds as follows [49, p. 89]. First, we compute ρ_ℓ; then we set $\widehat{\omega}_{\ell+1} = \tau + \mathbf{e}_\ell^T \mathbf{x}$, where $\mathbf{x}$ is the solution of the tridiagonal linear system $(T_\ell - \tau I)\mathbf{x} = \rho_\ell^2 \mathbf{e}_\ell$. The arithmetic mean between the ℓ-point Gauss rule $\mathcal{G}_\ell$ and the $(\ell+1)$-point Gauss–Radau rule $\widehat{\mathcal{G}}_{\ell+1}$ is then used as an approximation of the quadratic form $\mathbf{w}^T g(A^T A)\mathbf{w}$.

4.3.2 Second approach

In this section we provide a second approach to the approximation of bilinear forms expressed in terms of generalized matrix functions.

The following result shows how to compute the ℓ-point Gauss quadrature rule in terms of the generalized matrix function of the bidiagonal matrix B_ℓ. Two expressions are derived, depending on the starting (unit) vector given as input to the Golub–Kahan

algorithm. Recall that, unless $\mathbf{z} = A\mathbf{w}$ or $\mathbf{w} = A^T\mathbf{z}$, one has to use the polarization identity to estimate the bilinear forms of interest.

Proposition 4.14. [6, Proposition 5.5] *Let be $A \in \mathbb{R}^{m\times n}$ and let $B_\ell \in \mathbb{R}^{\ell\times\ell}$ be the bidiagonal matrix computed at step ℓ of the Golub–Kahan bidiagonalization algorithm. Then, the ℓ-point Gauss quadrature rule $\mathcal{G}_\ell$ is given by*

$$\mathcal{G}_\ell = \mathbf{e}_1^T B_\ell^\dagger f^\diamond(B_\ell)\mathbf{e}_1, \quad \text{if } \widetilde{\mathbf{z}} = \mathbf{w},$$

or

$$\mathcal{G}_\ell = \mathbf{e}_1^T f^\diamond(B_\ell) B_\ell^\dagger \mathbf{e}_1, \quad \text{if } \mathbf{z} = \widetilde{\mathbf{w}}.$$

The $(\ell+1)$-point Gauss-Radau quadrature rule $\widehat{\mathcal{G}}_{\ell+1}$ with a fixed node σ_1 can be expressed in terms of the entries of the bidiagonal matrix

$$\widehat{B}_{\ell+1} = \begin{pmatrix} B_\ell & \gamma_\ell \mathbf{e}_\ell \\ \mathbf{0}^T & \widehat{\omega}_{\ell+1} \end{pmatrix} \in \mathbb{R}^{(\ell+1)\times(\ell+1)}$$

as $\widehat{\mathcal{G}}_{\ell+1} = \mathbf{e}_1^T \widehat{B}_{\ell+1}^\dagger f^\diamond(\widehat{B}_{\ell+1})\mathbf{e}_1$ if $\widetilde{\mathbf{z}} = \mathbf{w}$ or as $\widehat{\mathcal{G}}_{\ell+1} = \mathbf{e}_1^T f^\diamond(\widehat{B}_{\ell+1})\widehat{B}_{\ell+1}^\dagger \mathbf{e}_1$ when $\mathbf{z} = \widetilde{\mathbf{w}}$.

The entries of $\widehat{B}_{\ell+1}$, except for the last diagonal entry, are those of $B_{\ell+1}$. To compute the last diagonal entry, one has to ensure that σ_1^2 is an eigenvalue of $\widehat{T}_{\ell+1} = \widehat{B}_{\ell+1}^T \widehat{B}_{\ell+1}$. It can be easily shown that

$$\widehat{\omega}_{\ell+1} = \sqrt{\sigma_1^2 + \mathbf{e}_\ell^T \mathbf{x} - \gamma_\ell^2},$$

where $\mathbf{x}$ is the solution of the tridiagonal linear system $(B_\ell^T B_\ell - \sigma_1^2 I)\mathbf{x} = (\omega_\ell \gamma_\ell)^2 \mathbf{e}_\ell$.

4.3.3 Third approach

Assume that we have used $\ell = r = \text{rank}(A)$ steps of the Golub–Kahan bidiagonalization algorithm with starting vector $\mathbf{w}$ (normalized so as to have unit norm) to derive the matrices P_r, B_r, and Q_r such that $A = P_r B_r Q_r^T$. The CSVD of the bidiagonal matrix is $B_r = \mathcal{U}_r \Sigma_r \mathcal{V}_r^T$, where Σ_r is the same diagonal matrix appearing in the CSVD of A. Since P_r and Q_r have full column rank, we know that $\text{rank}(P_r B_r Q_r^T) = \text{rank}(B_r) = r$,

and thus we can write

$$\begin{aligned}\mathbf{z}^T f^\diamond(A)\mathbf{w} &= \mathbf{z}^T f^\diamond(P_r B_r Q_r^T)\mathbf{w} = \mathbf{z}^T f^\diamond(P_r \mathcal{U}_r \Sigma_r \mathcal{V}_r^T Q_r^T)\mathbf{w} \\ &= \mathbf{z}^T (P_r \mathcal{U}_r) f(\Sigma_r)(Q_r \mathcal{V}_r)^T \mathbf{w} = \widehat{\mathbf{z}}^T f^\diamond(B_r)\mathbf{e}_1,\end{aligned}$$

where $\widehat{\mathbf{z}} = P_r^T \mathbf{z}$ and $Q_r^T \mathbf{w} = \mathbf{e}_1$.

Assume now that $\ell < r$. We can then truncate the bidiagonalization process and approximate $f^\diamond(A)\mathbf{w}$ as

$$f^\diamond(A)\mathbf{w} \approx P_\ell f^\diamond(B_\ell)\mathbf{e}_1$$

and then obtain the approximation to the bilinear form of interest as

$$\mathbf{z}^T f^\diamond(A)\mathbf{w} \approx \mathbf{z}^T P_\ell f^\diamond(B_\ell)\mathbf{e}_1.$$

The quality of the approximation will depend in general on the distribution of the singular values of A and on the particular choice of f. Generally speaking, if $f(\sigma_i)$ is much larger on the first few singular values of A than for the remaining ones, then a small number of steps result in approximations with small relative errors.

Remark 4.15. It is worth mentioning that these two latter approaches provide the same results in exact arithmetic.

4.4 Numerical results

In this section we present some numerical results concerning the application of the previously introduced techniques to the computation of centrality and communicability indices in directed networks. The first set of experiments concerns the computation of the *total hub communicability* of nodes, which, for a node i, is defined as the following bilinear form:

$$C_h(i) := [h^\diamond(A)\mathbf{1}]_i = \mathbf{e}_i^T h^\diamond(A)\mathbf{1}\,, \tag{4.9}$$

where $h(t) = \sinh(t)$ and A is the adjacency matrix of the digraph. As shown in Chapter 3, this quantity can be used to rank how important node i is when regarded as a "hub", i.e., as a broadcaster of information (analogous quantities rank the nodes in order of their importance as "authorities", i.e., receivers of information). The second set

of experiments concerns the computation of the resolvent-based communicability [11] between node i, playing the role of broadcaster of information, and node j, acting as a receiver. The quantities of interest here have the form $[h^{\diamond}(A)]_{ij}$, where $h(t) = \alpha t(1-(\alpha t)^2)^{-1}$ and $\alpha \in (0, \sigma_1^{-1})$. In all the tests we apply the approaches previously described and we use as stopping criterion

$$\mathcal{R}_\ell = \left| \frac{x^{(\ell+1)} - x^{(\ell)}}{x^{(\ell)}} \right| \leq \text{tol}, \tag{4.10}$$

where tol is a fixed tolerance and $x^{(\ell)}$ represents the approximation to the bilinear form of interest computed at step ℓ by the method under study. The relative error, used to assess the accuracy of approximation, is denoted by

$$\mathcal{E}_\ell = \frac{|x^{(\ell)} - \mathbf{z}^T h^{\diamond}(A)\mathbf{w}|}{|\mathbf{z}^T h^{\diamond}(A)\mathbf{w}|}.$$

Our dataset contains the adjacency matrices associated with three real world unweighted and directed networks: Roget, SLASHDOT, and ITwiki [9, 26, 77]. The adjacency matrix associated with Roget is 994×994 and has 7281 nonzeros. The graph contains information concerning the cross-references in Roget's Thesaurus. The adjacency matrix associated with SLASHDOT is an 82168×82168 matrix with 948464 nonzeros. For this network, there is a connection from node i to node j if user i indicated user j as a friend or a foe. The last network used in the tests, ITwiki, represents the Italian Wikipedia. Its adjacency matrix is 49728×49728 and has 941425 nonzeros, and there is a link from node i to node j in the graph if page i refers to page j.

Node centralities

In this section we want to investigate how the three approaches perform when we want to approximate (4.9), the total communicability of nodes in the network. For each of the three networks in the dataset, we computed the centralities of ten nodes chosen uniformly at random among all the nodes in the graph.

The results for the tests are presented in Tables 4.1–4.3. The tolerance used in the stopping criterion (4.10) is set to $\text{tol} = 10^{-6}$. The tables display the number of iterations required to satisfy the above criterion and the relative error of the computed solution

TABLE 4.1: Network: Roget, $h(t) = \sinh(t)$ (tol $= 10^{-6}$).

	First approach		Second approach		Third approach	
	ℓ	$\mathcal{E}_\ell$	ℓ	$\mathcal{E}_\ell$	ℓ	$\mathcal{E}_\ell$
1	8	1.10e-06	8	1.10e-06	9	9.45e-09
2	34	9.93e-08	34	9.74e-08	10	2.78e-09
3	5	3.20e-05	5	3.20e-05	8	5.26e-07
4	6	4.38e-06	6	4.38e-06	9	1.21e-08
5	20	6.18e-06	20	6.18e-06	9	1.21e-08
6	7	2.62e-06	7	2.62e-06	10	3.68e-10
7	8	7.08e-06	8	7.08e-06	9	1.99e-08
8	15	9.07e-07	15	9.07e-07	9	2.80e-08
9	9	8.15e-08	9	8.15e-08	9	1.72e-09
10	7	3.78e-07	7	3.78e-07	9	2.64e-08

TABLE 4.2: Network: SLASHDOT, $h(t) = \sinh(t)$ (tol $= 10^{-6}$).

	First approach		Second approach		Third approach	
	ℓ	$\mathcal{E}_\ell$	ℓ	$\mathcal{E}_\ell$	ℓ	$\mathcal{E}_\ell$
1	6	4.31e-07	6	5.61e-07	9	2.45e-08
2	9	3.24e-05	15	2.26e-06	9	1.56e-08
3	7	1.24e-06	8	1.75e-06	9	1.04e-07
4	14	2.21e-04	8	2.12e-04	10	1.74e-08
5	7	2.24e-05	7	2.35e-05	10	5.16e-09
6	10	4.84e-04	19	3.72e-04	10	1.99e-08
7	7	1.20e-06	7	1.20e-06	9	6.47e-08
8	7	7.11e-07	7	7.66e-07	9	7.68e-09
9	7	5.53e-06	7	5.98e-06	9	1.32e-09
10	6	6.98e-07	6	4.92e-07	8	8.68e-09

TABLE 4.3: Network: ITwiki, $h(t) = \sinh(t)$ (tol $= 10^{-6}$).

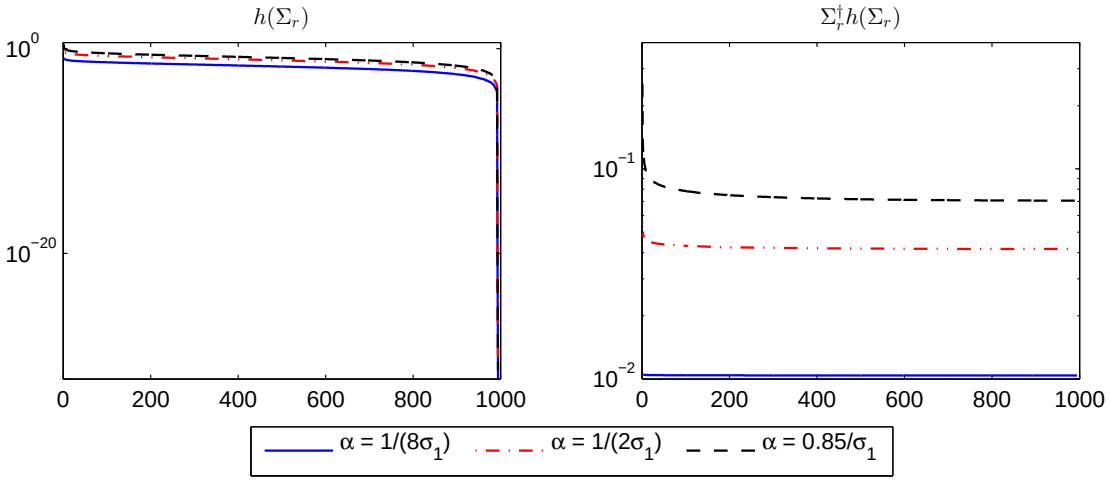
	First approach		Second approach		Third approach	
	ℓ	$\mathcal{E}_\ell$	ℓ	$\mathcal{E}_\ell$	ℓ	$\mathcal{E}_\ell$
1	5	3.88e-08	5	2.90e-08	6	8.02e-09
2	10	4.72e-05	9	4.68e-05	7	1.27e-08
3	5	3.20e-08	5	3.17e-08	6	7.01e-09
4	7	2.31e-05	9	2.33e-05	8	4.31e-09
5	8	4.20e-05	20	5.77e-05	8	5.91e-09
6	9	2.19e-04	24	2.13e-04	8	2.70e-08
7	6	4.26e-07	6	5.85e-07	7	3.15e-09
8	14	1.91e-04	29	2.24e-04	8	3.38e-09
9	5	8.57e-08	5	9.31e-08	6	5.07e-09
10	9	9.36e-06	8	1.12e-05	8	3.22e-10

with respect to the "exact" value of the bilinear form. The latter has been computed using the full SVD for the smallest network, and using a partial SVD with a sufficiently large number of terms ($\gg \ell$) for the two larger ones.

Concerning the first approach, since $g(t) = (\sqrt{t})^{-1}\sinh(\sqrt{t})$ is not completely monotonic, we have used the Gauss quadrature rule as an approximation for the quantities of interest, rather than as a lower bound.

As one can see from the tables, only a small number of steps is required for all the three approaches. The third approach appears to be the best one for computing these

FIGURE 4.1: Network: Roget. Diagonal entries of $h(\Sigma_r)$ and $\Sigma_r^\dagger h(\Sigma_r)$ for $h(t) = \frac{\alpha t}{1-(\alpha t)^2}$, when $\alpha = \frac{1}{8\sigma_1}, \frac{1}{2\sigma_1}, 0.85\ \sigma_1^{-1}$



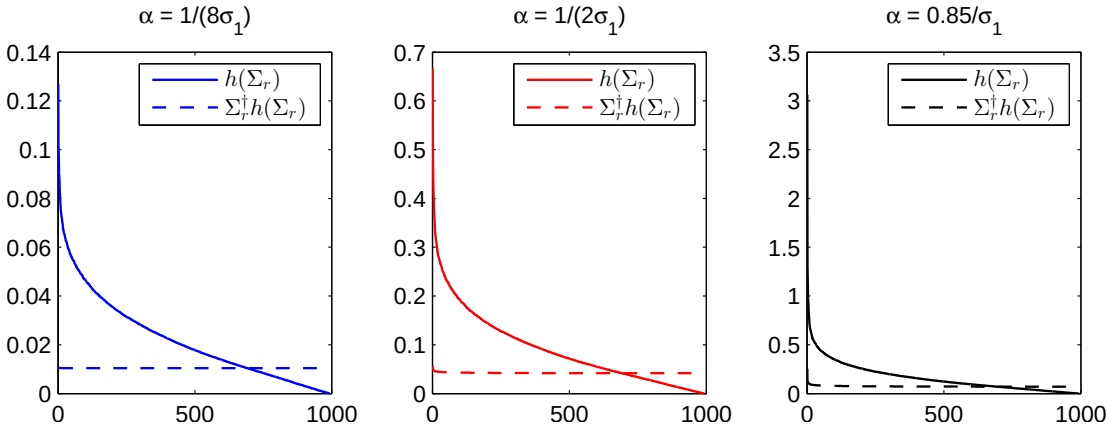
quantities since it requires almost always the same number of steps for all the nodes in each network, while attaining higher accuracy. Somewhat inferior results (in terms of both the number of iterations performed and the accuracy of the computed solution) are obtained with the other two approaches, which however return very good results as well. This can be explained by observing that the function $\sinh(t)$ being applied to the larger (approximate) singular values of A takes much larger values than the function $t^{-1}\sinh(t)$ used by the other two approaches, therefore a small relative error can be attained in fewer steps (since the largest singular values are the first to converge).

Resolvent-based communicability between nodes

Our second set of numerical experiments concerns the computation of the resolvent-based communicability between two nodes i and j. The concerned function is now $h(t) = \frac{\alpha t}{1-(\alpha t)^2}$, where $\alpha \in (0, \sigma_1^{-1})$ is a user-defined parameter. The generalized matrix function $h^\diamond(A)$ arises as the top right square block of the matrix resolvent $(I - \alpha\mathscr{A})^{-1}$, where the matrix $\mathscr{A}$ is defined as in (1.1). This resolvent function is similar to one first used by Katz to assign centrality indices to nodes in a network, see [59]. In [11] the authors showed that when $\mathscr{A}$ is as in (1.1), the resolvent can be written as

$$(I - \alpha\mathscr{A})^{-1} = \begin{pmatrix} (I - \alpha^2 AA^T)^{-1} & h^\diamond(A) \\ h^\diamond(A^T) & (I - \alpha^2 A^T A)^{-1} \end{pmatrix}, \qquad \alpha \in (0, \sigma_1^{-1}).$$

FIGURE 4.2: Network: Roget. Diagonal entries of $h(\Sigma_r)$ and $\Sigma_r^\dagger h(\Sigma_r)$ for $h(t) = \frac{\alpha t}{1-(\alpha t)^2}$, when $\alpha = \frac{1}{8\sigma_1}, \frac{1}{2\sigma_1}, 0.85\ \sigma_1^{-1}$



Furthermore, the entries of its top right block can be used to account for the communicability between node i (playing the role of spreader of information, or hub) and node j (playing the role of receiver, or authority). As before, the function $g(t) = (\sqrt{t})^{-1}h(\sqrt{t})$ is not completely monotonic. Thus the Gauss rule can only be expected to provide an approximation to the quantity of interest.

We have performed three different tests on the network Roget for three different values of α. More specifically, we have tested $\alpha = \frac{1}{8\sigma_1}, \frac{1}{2\sigma_1}$, and $\frac{17}{20\sigma_1} = \frac{0.85}{\sigma_1}$. Figure 4.1 shows the values of the diagonal entries of $h(\Sigma_r)$ and $\Sigma_r^\dagger h(\Sigma_r)$ for the three different values of α used in the tests. Figure 4.2 plots the respective behavior of the diagonal entries of $\Sigma_r^\dagger h(\Sigma_r)$ and $h(\Sigma_r)$ for the three values of the parameter α. From these plots, one can expect the first and second approach to require a higher number of steps than that required by the third one; this is because the leading singular values are mapped to appreciably larger values when applying the function $h(t)$ than when applying the function $t^{-1}h(t)$. The results for this set of experiments are contained in Tables 4.4–4.6, when the tolerance for the stopping criterion (4.10) is set to tol $= 10^{-4}$. The pairs of nodes whose communicability we want to approximate are chosen uniformly at random among all the possible pairs of distinct nodes in the graph. We kept the same set of pairs in all the three experiments. We want to point out, however, that the value being computed for each pair varies with α, and thus the results (in terms of number of iterations and accuracy) cannot be compared among the three tables.

As can be clearly seen from the tables, the fastest and most accurate method (in terms of

TABLE 4.4: Network: Roget, $h(t) = \frac{\alpha t}{1-(\alpha t)^2}$, $\alpha = \frac{1}{8\sigma_1}$ (tol $= 10^{-4}$).

	First approach		Second approach		Third approach	
	ℓ	$\mathcal{E}_\ell$	ℓ	$\mathcal{E}_\ell$	ℓ	$\mathcal{E}_\ell$
1	75	2.14e+03	75	2.14e+03	5	6.61e-08
2	106	1.60e-02	106	1.60e-02	4	3.75e-08
3	906	2.99e-01	906	2.99e-01	5	1.65e-08
4	992	2.17e-08	992	7.82e-06	5	1.33e-07
5	166	7.16e+02	166	7.16e+02	5	7.52e-08
6	257	1.03e-01	257	1.03e-01	4	4.46e-08
7	874	5.41e-01	874	5.41e-01	5	2.46e-08
8	274	1.06e+00	274	1.06e+00	5	9.46e-08
9	259	8.37e-04	259	8.37e-04	5	7.31e-11
10	733	2.48e+01	733	2.48e+01	5	3.37e-07

TABLE 4.5: Network: Roget, $h(t) = \frac{\alpha t}{1-(\alpha t)^2}$, $\alpha = \frac{1}{2\sigma_1}$ (tol $= 10^{-4}$).

	First approach		Second approach		Third approach	
	ℓ	$\mathcal{E}_\ell$	ℓ	$\mathcal{E}_\ell$	ℓ	$\mathcal{E}_\ell$
1	75	6.32e+00	75	6.32e+00	7	1.90e-06
2	87	8.90e-03	87	8.90e-03	6	6.02e-07
3	308	1.10e-02	308	1.10e-02	6	7.96e-06
4	106	5.71e-01	106	5.72e-01	7	7.77e-08
5	495	1.72e-01	495	1.72e-01	7	4.43e-06
6	79	4.51e-02	79	4.51e-02	6	2.12e-06
7	118	8.64e-02	118	8.64e-02	7	4.24e-07
8	121	1.00e-01	121	1.01e-01	7	5.84e-07
9	59	1.91e-02	59	1.91e-02	6	6.99e-07
10	574	1.87e-01	574	1.87e-01	7	1.49e-06

TABLE 4.6: Network: Roget, $h(t) = \frac{\alpha t}{1-(\alpha t)^2}$, $\alpha = 0.85\ \sigma_1^{-1}$ (tol $= 10^{-4}$).

	First approach		Second approach		Third approach	
	ℓ	$\mathcal{E}_\ell$	ℓ	$\mathcal{E}_\ell$	ℓ	$\mathcal{E}_\ell$
1	74	2.72e-01	74	2.72e-01	10	4.88e-06
2	66	3.45e-03	66	3.45e-03	8	5.22e-06
3	97	6.96e-02	97	6.96e-02	8	4.79e-06
4	58	3.72e-02	58	3.72e-02	8	8.04e-05
5	147	6.23e-02	147	6.23e-02	9	1.23e-05
6	53	8.48e-03	53	8.48e-03	9	3.34e-06
7	74	1.58e-02	74	1.58e-02	7	3.20e-04
8	117	6.49e-03	117	6.49e-03	10	4.52e-06
9	23	6.02e-03	23	6.02e-03	9	4.34e-07
10	152	1.70e-01	152	1.70e-01	9	3.90e-05

relative error with respect to the exact value) is once again the third. Indeed, it requires fewer steps than the first two approaches and it achieves a higher level of accuracy (see, e.g., Table 4.5). In fact, in some cases the first two approaches stabilize at a value which is far from the quantity that needs to be computed; this kind of stagnation leads to the termination criterion to be satisfied even if convergence has not been attained. Moreover, in Table 4.4, case 4 requires $\ell = \text{rank}(A) = 992$ steps to satisfy (4.10) for the first two approaches, whereas the third only requires 5 steps.

TABLE 4.7: Network: Roget, $h(t) = \sinh(t)$ (tol $= 10^{-6}$).

	First approach		Second approach		Third approach		mmq	
	ℓ	$\mathcal{E}_\ell$	ℓ	$\mathcal{E}_\ell$	ℓ	$\mathcal{E}_\ell$	ℓ	$\mathcal{E}_\ell$
1	10	4.69e-06	10	4.69e-06	9	1.09e-08	11	2.46e-09
2	6	4.48e-06	6	4.48e-06	9	3.71e-08	10	3.38e-08
3	9	4.27e-06	9	4.27e-06	9	5.07e-10	10	3.35e-08
4	5	1.82e-06	5	1.82e-06	8	2.36e-07	10	4.34e-08
5	13	4.25e-06	13	4.25e-06	7	1.29e-08	10	3.47e-09
6	10	1.20e-05	10	1.20e-05	9	4.34e-08	10	1.44e-08
7	4	1.96e-06	4	1.96e-06	9	4.63e-09	9	5.29e-10
8	17	1.02e-05	17	1.02e-05	9	8.20e-08	11	3.53e-09
9	6	1.26e-05	6	1.26e-05	10	2.13e-09	10	4.86e-08
10	13	3.62e-06	13	3.62e-06	9	7.45e-08	11	2.29e-09

Comparison with standard Lanczos-based approach

In the case of generalized matrix functions like $\sinh^\diamond(A)$, which occur as submatrices of "standard" matrix functions applied to the symmetric matrix $\mathscr{A}$, it is natural to compare the previously proposed approaches with the use of Gauss quadrature-based bounds and estimates based on the Lanczos process. This was the approach used, for example, in [11]. Henceforth, we refer to this approach as "the `mmq` approach," since it is implemented on the basis of the `mmq` toolkit [72] originally developed by Gérard Meurant; see also [49].

We have computed the hub centrality of ten nodes chosen uniformly at random among all the nodes in the network Roget using our three approaches and the `mmq` approach. The results when the tolerance in (4.10) is set to tol $= 10^{-6}$ are summarized in Table 4.7 experiments. These indicate that on average, the `mmq` approach requires a slightly higher number of iterations than our third approach to deliver comparable accuracy. Note that the cost per step is comparable for the two methods. An advantage of the `mmq` approach is that it can provide lower and upper bounds on the quantities being computed, but only if bounds on the singular values of A are available. A disadvantage is that it requires working with vectors of length $2n$ instead of n.

Of course, the Lanczos-based approach is not applicable to generalized matrix functions that do not arise as submatrices of standard matrix functions.

4.5 Conclusions

In this chapter we have proposed several algorithms for the computation of certain quantities associated with generalized matrix functions. These techniques are based on Gaussian quadrature rules and different variants of the Lanczos and Golub–Kahan algorithms. In particular, we have investigated three distinct approaches for estimating scalar quantities like $\mathbf{z}^{\mathbf{T}} f^{\diamond}(A)\mathbf{w}$. The performance of the various approaches has been tested in the context of computations arising in network theory. We have observed that the quality of approximation and the number of iterations required to satisfy the stopping criterion depend in general on the choice of the function and on its action on the diagonal entries of Σ_r.

Chapter 5

Predicting triadic closure in complex networks

It is well-documented that real-world networks have a clustering coefficient (1.9) that is larger than what one would expect from the uniform model [78]. The high degree of transitivity is a common characteristic of many types of networks, such as social, biomolecular, cellular, ecological, infrastructural, and technological (see [34] and references therein). In 1922 Simmel [86] theorized that people with common friends are more likely to create friendships. This "*friendship transitivity*" definitively implies a social mechanism for triadic closure in social networks which may then be applied to explain the evolution of triangle closures [63]. It assumes that individuals can benefit from cooperative relations, and this may induce individuals to choose new acquaintances from among their friends' friends. Although intuitive, this simple idea has some fundamental drawbacks. First, it is not always true that pairs of nodes benefit from cooperative relations, and therefore the Simmelian principle is useless in such situations. Secondly, it is evident that not every pair of nodes separated by two edges participates in a triangle in a real-world network. Thus, some kind of selective process has been taking place, closing some of the triads in a network and leaving many others open.

In this chapter, which is based on [37], we describe a general mechanism to account for such selective process of triadic closure in networks. We propose a strategy for predicting triadic closure based on the idea that triadic closure is a communication-driven process.

5.1 Communicability distances and triad closure

The communicability (function) [39–41], defined $\forall i, j \in \mathcal{V}$ as $(e^A)_{ij}$, can be used to quantify the quality of communication between nodes i and j in a network, which depends on two factors: how much information departing from a source node reaches its target $(e^A)_{ij}$, and how much information departing from the node returns to it without ending at its destination $(e^A)_{ii}$. That is, two nodes communicate better when there is a large amount of information that departs from the originator and arrives at its destination. On the other hand, the quality of communication between two nodes decreases when there is a large amount of information that is wasted due to the fact that the information returns to its source without being delivered to the target. In [35] the *communicability distance* is defined as

$$\xi_{ij} = \sqrt{(e^A)_{ii} + (e^A)_{jj} - 2\,(e^A)_{ij}}. \tag{5.1}$$

This is an Euclidean distance between the nodes i and j in $\mathcal{G}$ (see [35, 36]). From its definition, it is clear that ξ_{ij} characterizes the quality of the communication taking place between nodes i and j.

We start by considering the square of the communicability distance defined in (5.1) for a pair of nodes i and j in a connected graph. This distance characterizes how effectively nodes i and j communicate when we assume that the information departing from node i travels to node j (and vice versa) by taking a series of one-hop steps between the nodes in any of the walks connecting them. From (5.1), it is clear that the smaller the value of ξ_{ij}^2, the better nodes i and j are at exchanging information.

The communicability distance (5.1) depends on e^A, where A is the adjacency matrix of a simple graph. If we consider i and j such that $a_{ij} = 1$, then we are assuming that these two nodes are attracted to each other. If instead we were to consider that these nodes repel each other, we would use e^{-A}.

If the (squared) communicability distance between two pairs of nodes i and j and p and q satisfies $\xi_{ij}^2 < \xi_{pq}^2$ then we say that the attraction between the pair i and j is *stronger* than that of the pair p and q in the corresponding network.

Now, consider a triad i, l, j, where $(i, l) \in \mathcal{E}$, $(l, j) \in \mathcal{E}$ but $(i, j) \notin \mathcal{E}$. Because $a_{il} = 1$ and $a_{lj} = 1$ we can infer that there are attractive "forces" between i and l and between l and

j. A simple metaphoric way to represent such attractive forces between pairs of nodes is to suppose that they have opposite charges which attract to each other. For instance, we can consider either of the following schemes for the previous example: $i^+ - l^- - j^+$ or $i^- - l^+ - j^-$. Notice that considering a particle spin, as is usually done in sociophysical models of opinion dynamics, also works here as an appropriate metaphor (see, e.g., [84]). Observe that there are two types of interactions between the nodes i and j. First, due to the attractions between i and l, and l and j, node j "feels" an attractive force from i, which is transmitted through the edges of the network. On the other hand, due to the fact that both i and j have the same charge, they experience some repulsion from each other, which takes place in a "through-space" fashion, as we will clarify later. We can thus expect the link (i, j) to be created if the through-edge attractive force between the nodes i and j is larger than the through-space repulsive force between them.

In order to understand the nature of the interactions described in the previous paragraph we consider a molecular system as a model example. In this case there is a communication between pairs of atoms which occurs through the covalent bonds of the molecule. This kind of interaction takes place through the edges (covalent bonds) and is analogous to the attractive forces we have previously described. Hereafter we will refer to this interaction as the *Through-Edges Communicability (TEC)*. If two non-covalently bonded atoms are close in space, they can interact with each other through non-covalent interactions, for example, by hydrophobic, polarity, or electrostatic forces. These interactions are analogous to our through-space repulsion and we will refer to them as *direct Long-Range Communicability (LRC)*. In a social network, TEC is present when information is transmitted from one individual to another in the network by using the social ties that define the edges of the graph. On the other hand, LRC is realized by the direct influence of an individual to another through any source of social signalling.

Note that although the shortest path distance between every pair of nodes in a triangle equals one, every pair of nodes in it is connected by a pair of adjacent edges through the third vertex. A natural way to account for all the pairs of nodes connected by pairs of adjacent edges is to consider the number of walks of length two between the pairs of nodes. We can then transform a graph accordingly. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a simple and connected graph and let $\mathcal{W}_2(\mathcal{G}) = (\mathcal{V}, \mathcal{E}')$ be the graph with the same set of nodes as $\mathcal{G}$, but whose edges are weighted by the number of walks of length two between every pair of (not necessarily distinct) nodes in $\mathcal{G}$. More precisely, if $\mu_2(i, j)$ is the number of

walks of length two between nodes i and j, then the adjacency matrix $A^{(2)} = (a_{ij}^{(2)})$ of $\mathcal{W}_2(\mathcal{G})$ is

$$a_{ij}^{(2)} = \begin{cases} \mu_2(i,j) & i \neq j \\ \mu_2(i,i) = d_i & i = j, \end{cases}$$

where d_i is the degree of node i.

Remark 5.1. Clearly $A^{(2)} = A^2$ and so we do not need to explicitly construct the graph $\mathcal{W}_2(\mathcal{G})$, since we can simply work with the square of the adjacency matrix of the original graph $\mathcal{G}$.

Note that two nodes are connected in $\mathcal{W}_2(\mathcal{G})$ if they have the same charge. Thus, connected nodes in $\mathcal{W}_2(\mathcal{G})$ repel each other. Consequently the *repulsive communicability* between a given pair of nodes in $\mathcal{G}$ is given by $(e^{-A^{(2)}})_{ij} = (e^{-A^2})_{ij}$.

Using this idea of repulsive communicability, we can define a communicability distance based on $(e^{-A^2})_{ij}$ which accounts for the quality of LRC between pairs of nodes separated by two adjacent edges, i.e., pairs of nodes feeling mutual repulsion in $\mathcal{G}$. This new communicability distance function will be defined as

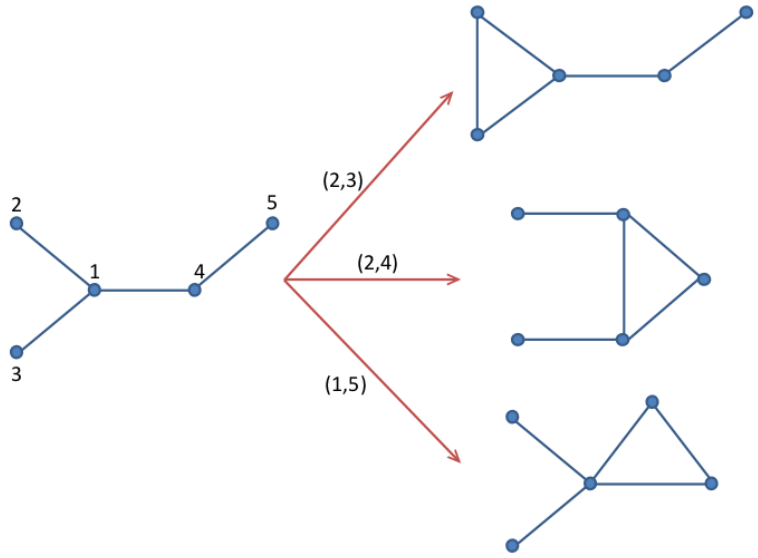
$$\eta_{ij} = \sqrt{(e^{-A^2})_{ii} + (e^{-A^2})_{jj} - 2(e^{-A^2})_{ij}}. \tag{5.2}$$

A large value of η_{ij} indicates that there is a weak repulsion between nodes i and j. This new communicability distance η_{ij} is an Euclidean distance between nodes i and j. The proof of this follows the same lines as in [35, 36] and it is hence omitted.

Remark 5.2. The graph $\mathcal{W}_2(\mathcal{G})$ is not always connected. Therefore, the function η_{ij} is defined only for pairs of nodes which sit in the same connected component of the graph. Elsewhere η_{ij} is set to infinity.

Before continuing, consider the following example. The tree illustrated on the left in Figure 5.1 can be transformed by adding an edge which closes any of the three nonequivalent existing triads of the graph, i.e., by adding the edge $(2,3)$, $(2,4)$ or $(1,5)$. The resulting unicyclic graphs are illustrated on the right of Figure 5.1. In Table 5.1 we report the values of ξ_{ij}^2 and η_{ij}^2 for each of the three triads. Now assume that we have information indicating that the process giving rise to the closure of the $1,2,3$-triad is favored over the other two. We cannot a priori know for any particular system how the attractive and repulsive forces scale. In real physical systems such terms are scaled by

FIGURE 5.1: Example of evolution of a tree after one edge is added to close a triangle.



minimizing the global energy of the system. Here we simply consider the weighted difference between the two terms: $\alpha\xi_{ij}^2 - \beta\eta_{ij}^2$, where α and β are two parameters. We will propose a method to determine the values of the empirical parameter α and β in a given network a little later. For this example it can be verified that, for instance, $\xi_{ij}^2 - 1.5\eta_{ij}^2$ produces a negative value only for the pair $(2, 3)$ (see Table 5.1). This weighted difference between ξ_{ij}^2 and η_{ij}^2 corresponds to the case in which the attractive forces between the corresponding nodes outweigh the magnitude of the repulsive ones. As noted previously, a large value of η_{ij}^2 indicates a small repulsion between the corresponding nodes, and here we have multiplied η_{ij}^2 by a coefficient $\beta > 1$, which further reduces the repulsive forces.

Now suppose instead that we have information indicating that the process giving rise to the closure of the $1, 4, 5$-triad is favored over the other two. In this case it can be verified (see Table 5.1) that the weighted difference $-0.5\xi_{ij}^2 + 1.5\eta_{ij}^2$ is negative only for the pair $(1, 5)$. Here, we have considered that the attractive forces between the nodes make a negative contribution to the creation of an edge closing the triad. This may correspond to the situation in which the links (i, l) and (l, j) are both very weak, i.e., friendship ties between the corresponding individuals are not too strong. We further weaken those relations by multiplying ξ_{ij}^2 by a coefficient $\alpha < 0$. At the same time, by multiplying η_{ij}^2 by a coefficient $\beta < 0$ we have assumed that the repulsive factor does not play a major role in determining whether the new edge is created or not. Indeed, in this way η_{ij}^2 is

TABLE 5.1: Values of weighted sum of ξ^2 and η^2 for the potential edges considered in Figure 5.1.

pair	ξ_{ij}^2	η_{ij}^2	$\xi_{ij}^2 - 1.5\eta_{ij}^2$	$-0.5\xi_{ij}^2 + 1.5\eta_{ij}^2$	$\xi_{ij}^2 + \eta_{ij}^2$
2,3	2.000	2.000	**-1.000**	2.000	4.000
1,5	3.184	0.960	1.744	**-0.152**	4.144
2,4	2.545	1.312	0.577	0.696	**3.857**

transformed into an attraction term. In the charged-particles analogy this corresponds to a situation in which the charges between the corresponding nodes are very weak and there is no repulsion between those nodes separated by two adjacent edges.

Finally, suppose that we have information indicating that the process giving rise to the closure of the $1, 2, 4$-triad is favored over the other two. In this case it can be verified (see Table 5.1) that the weighted difference $\xi_{ij}^2 + \eta_{ij}^2$ reaches the smallest value for $(2, 4)$. The values of the weighted differences for the three triad closure processes are positive, but the one corresponding to the closure of he $1, 2, 4$-triad is the lowest among the three. In this case, triadic closure is dominated by attractive forces only. The term $\alpha\xi_{ij}^2$ with $\alpha > 0$ indicates the normal attractive forces between the corresponding pair of nodes while $\beta\eta_{ij}^2$ with $\beta < 0$ is transformed into an attractive term.

A case we have not considered here is if $\alpha < 0$ and $\beta > 0$, when both terms represent repulsive forces between nodes. In this case $\alpha\xi_{ij}^2 - \beta\eta_{ij}^2 < 0$ for all i, j and the order in which the triads will be closed is determined by the magnitudes of α and β. In such a repulsive system there are no attractive forces to fuel the creation of new edges. Consequently, the creation of the new edges to close triads is controlled by factors such as their similarities or complementarity in their functions which do not depend particularly on the communicability between nodes. In this case, predictions of triad closure made on the basis of communicability distances are not expected to differ significantly from those made by a random closure of the triads.

In summary, we can use the function

$$\Delta_{ij}(\alpha, \beta) := \alpha\xi_{ij}^2 - \beta\eta_{ij}^2, \qquad \forall i, j \in \mathcal{V}, \tag{5.3}$$

to determine which triad is closed in the network.

In order to predict which triads will close in a given network it is necessary to know the values of α and β. We now propose a method that allows us to estimate these empirical

parameters and consequently to determine which triads will close in a given network.

5.2 Proposed method

In order to predict the triadic closure in a network based on $\Delta_{ij}(\alpha, \beta)$ we develop a procedure to find the values of the empirical parameters α and β which best predict the triadic closure in a network from which we have removed all the triangles. That is, we take a network $\mathcal{G}$ and we list all its existing triangles. We then transform $\mathcal{G}$ into a triangle-free graph $\mathcal{G}'$ by removing one and only one of the edges forming each triangle. The deleted edges are selected uniformly at random from the three edges forming each triangle. As this procedure is likely to be repeated a large number of times (see below for details), the chance that each of the three edges is selected at least once is very high. We keep a list of all these removed edges which we call R. It may happen that two triangles T_1 and T_2 share an edge e. If we select e when considering T_1, then, when it comes to select an edge in T_2, we pick an edge which may or may not coincide with e. If it does, we do not add it to the list. It may also happen that T_2 consists of e and two other edges, one of which has also already been removed because it was in common with a third triangle. In such cases, we do not remove the last connection remaining in T_2, since it could disconnect the network.

We also create a list P of all the pairs of nodes which form triads in $\mathcal{G}$ but are not part of any triangle. In other words, P contains the potential edges of the network $\mathcal{G}$ (cf. Chapter 1). Finally, we create the list $L = R \cup P$ which contains all the potential edges in $\mathcal{G}'$. Our task is to select appropriate values of the empirical parameters α and β that differentiate as much as possible the pairs of nodes in R from those in P. We do this by using a non-increasing ranking of all the pairs of nodes in L according to $\Delta_{ij}(\alpha, \beta)$. We have previously predicted that the triadic closure process should be controlled by the smallest values of $\Delta_{ij}(\alpha, \beta)$ (see the example described in Figure 5.1). Thus, we expect that a non-increasing ranking of the values of $\Delta_{ij}(\alpha, \beta)$ contains most of the elements of R at the top of the ranking and most of those of P at the bottom.

In order to quantify the percentage of triangles that were correctly predicted we proceed as follow. We first rank the entries of L in non-increasing order using the quantities (5.3). We select the top r entries of this ordered list, where r is the cardinality of R. Then,

we count the number r_p of entries in this top r which are elements of R. These entries correspond to those virtual edges which were originally closing triangles in $\mathcal{G}$. That is, r_p represents the number of correct predictions made by the current method. We call the (percentage) ratio of r_p to r the percentage of `detected`.

5.2.1 Datasets and computational methods

We now give some computational details on how we implemented these calculations to find the optimal values of α and β for a selection of networks.

We study 25 networks representing complex systems from a wide variety of environments, such as social (Colorado, dolphins, Drugs, Galesburg, Geom, High School, High tech, Matheoremethod, Prison, Sawmill, social3, Zachary), ecological (BridgeBrook, Grassland, ScotchBroom, StMartin, and Ythan1), biomolecular (Neurons, PIN_Ecoli, PIN_Human, PIN_Yeast, and Transc_yeast), technological (Internet and USAir97), and informational (Roget). A brief description of all these networks is given in Appendix A.

In order to find the optimal values of the empirical parameters α and β for these networks we proceed as follows. We calculate all the values of α and β in the interval $I = [-2.1, 2.1]$ with a step length of 0.1. This interval I has been determined empirically as smaller intervals led to worse results and larger ones did not improve the results. Then, for each combination of α and β in $\Delta_{ij}(\alpha, \beta)$ we rank all the elements of L in non-increasing order and find the percentage of `detected`. The optimal values of α and β for this particular network are those that produce the highest percentage of `detected`. These computations were repeated 100 times.

The effectiveness of the proposed method is tested by considering a simple null model constructed as follows. We randomly order the edges in L, select the top r pairs of nodes and count how many of them were in the set R. With this information we compute the percentage of correct predictions made by a random ordering of the pairs of nodes (`rand`). Similar values of the percentages of `detected` and `rand` indicate that the ranking produced by the function $\Delta_{ij}(\alpha, \beta)$ does not differ significantly from a random ordering of the pairs of nodes and consequently is not a good one; while larger differences between the percentages of `detected` and `rand` indicate good performance of the proposed method.

Before starting with the detailed analysis of these 25 datasets we consider the possibility of fixing one of the parameters (α or β) and letting the other varying in the bounded interval $[-2.1, 2.1]$. To do this we set $\alpha = 1$ and let β vary. This seems reasonable, since this choice allows us to tune the disturbance caused by the repulsion in the values of Δ_{ij}. However, the results obtained for 10 of the studied networks discouraged us from proceeding with this approach. On average the use of the two parameters α and β makes predictions of triadic closure which are 7% higher than those using only one parameter, with maximum differences of up to 20% for one network (results not shown here). Thus, we will use the more general approach of calibrating both empirical parameters.

5.2.2 Bounds for communicability distance functions

Although in our experiments we use the exact values of the communicability distance functions in order to obtain the values of Δ_{ij}, we now give some bounds for ξ_{ij} and η_{ij}, which can be used in the computations when working on extremely large networks. It is clear from the definitions given in (5.1), (5.2), and (5.3) that for large matrices these values may be too costly to compute. To avoid the computation of the matrix exponential, we derive bounds for ξ_{ij}^2 and η_{ij}^2 (and therefore for $\Delta_{ij}(\alpha, \beta)$) by means of a Gauss–Radau quadrature rule as described in Section 1.3.

Our results are summarized in the following theorems.

Theorem 5.3. *Let A be the adjacency matrix of an unweighted and undirected network. Then*

$$\Phi\left(b, \omega_1 + \frac{\gamma_1^2}{\omega_1 - b}\right) \leq \frac{(\xi_{ij})^2}{2} \leq \Phi\left(a, \omega_1 + \frac{\gamma_1^2}{\omega_1 - a}\right), \tag{5.4}$$

where

$$\Phi(x, y) = \frac{c\,(e^{-x} - e^{-y}) + xe^{-y} - ye^{-x}}{x - y}, \qquad c = \omega_1, \tag{5.5}$$

$$\begin{cases} \omega_1 = a_{ij}, \\ \gamma_1 = \left[\frac{d_i + d_j}{2} - \omega_1 - A_{ij}^2\right]^{\frac{1}{2}}, \end{cases}$$

and $[a, b]$ is an interval containing the spectrum of $-A$.

The proof of this result can be found in [37]. Since it goes along the same line as the proof of Theorem 2.1 it is omitted.

Remark 5.4. If $(i,j) \notin \mathcal{E}$ the bounds simplify considerably. Indeed, in this case $\omega_1 = 0$ and hence

$$\frac{b^2 e^{\frac{\gamma_1^2}{b}} + \gamma_1^2 e^{-b}}{b^2 + \gamma_1^2} \leq \frac{(\xi_{ij})^2}{2} \leq \frac{a^2 e^{\frac{\gamma_1^2}{a}} + \gamma_1^2 e^{-a}}{a^2 + \gamma_1^2}$$

Similar bounds can be computed for η_{ij}^2 and are described in the following theorem.

Theorem 5.5. *Let A be the adjacency matrix of an unweighted and undirected network. Then*

$$\Phi\left(\tilde{b}, \tilde{\omega}_1 + \frac{\tilde{\gamma}_1^2}{\tilde{\omega}_1 - \tilde{b}}\right) \leq \frac{(\eta_{ij})^2}{2} \leq \Phi\left(\tilde{a}, \tilde{\omega}_1 + \frac{\tilde{\gamma}_1^2}{\tilde{\omega}_1 - \tilde{a}}\right)$$

where Φ is defined as in (5.5) *with $c = \tilde{\omega}_1$, $[\tilde{a}, \tilde{b}]$ is an interval containing the spectrum of A^2, and*

$$\begin{cases} \tilde{\omega}_1 = \gamma_1^2 + \omega_1; \\ \tilde{\gamma}_1 = \left[\frac{1}{2}\sum_{l=1}^{n}\left(A_{il}^2 - A_{lj}^2\right)^2 - \tilde{\omega_1}^2\right]^{\frac{1}{2}} \end{cases}.$$

with ω_1 and γ_1 as in Theorem 5.3.

Remark 5.6. Since the behavior of the eigenvalues of A is known (see [91]), we may take $\tilde{a} = 0$ and $\tilde{b} = a^2$ as the square of the approximation to the largest eigenvalue of A. For these values, the bounds simplify to

$$\Phi\left(a^2, \tilde{\omega}_1 + \frac{\tilde{\gamma}_1^2}{\tilde{\omega}_1 - a^2}\right) \leq \frac{(\eta_{ij})^2}{2} \leq \Phi\left(0, \tilde{\omega}_1 + \frac{\tilde{\gamma}_1^2}{\tilde{\omega}_1}\right) = \frac{\tilde{\omega}_1^2 e^{-\frac{\tilde{\omega}_1^2 + \tilde{\gamma}_1^2}{\tilde{\omega}_1}} + \tilde{\gamma}_1^2}{\tilde{\omega}_1^2 + \tilde{\gamma}_1^2}.$$

Combining the results described in the previous theorems, one easily get bounds for the values of $\frac{\Delta_{ij}(\alpha,\beta)}{2}$. Indeed, the computation is straightforward, since the new bounds are linear combinations of the previous ones. For example, if we have $\alpha \geq 0$ and $\beta \leq 0$ we get as lower bound for $\frac{\Delta_{ij}(\alpha,\beta)}{2}$

$$\alpha\Phi\left(b, \omega_1 + \frac{\gamma_1^2}{\omega_1 - b}\right) + \beta\Phi\left(\tilde{a}, \tilde{\omega}_1 + \frac{\tilde{\gamma}_1^2}{\tilde{\omega}_1 - \tilde{a}}\right),$$

and as upper bound

$$\alpha\Phi\left(a, \omega_1 + \frac{\gamma_1^2}{\omega_1 - a}\right) + \beta\Phi\left(\tilde{b}, \tilde{\omega}_1 + \frac{\tilde{\gamma}_1^2}{\tilde{\omega}_1 - \tilde{b}}\right),$$

where ω_1, γ_1, $\tilde{\omega}_1$, and $\tilde{\gamma}_1$ depend on the choice of i and j.

5.3 Modeling results and discussion

5.3.1 Predicting and interpreting triadic closure

The first series of results refers to the finding of the optimal values of α and β for the studied networks and the comparison of the percentage of triadic closures correctly predicted by the proposed method in comparison with the random process. The results of the tests are listed in Table 5.2. The columns $\langle\alpha^*\rangle$ and $\langle\beta^*\rangle$ contain the average best values for the parameters, where the average is taken over the 100 iterations we run. The results show that on average the method based on the communicability distance functions (`detect`) correctly predicts 20% of the triad closures in the real-world networks studied. In 7 cases this percentage of correct prediction reaches values larger than 25%. In contrast, the random closure of triads identifies 7.6% of the real triangles existing in those networks.

We can now gain some insights about the processes that have governed the triad closure in the studied networks. According to our standpoint, the triadic closure process arises from a combination of two kinds of node interaction: TEC and LRC. If we refer to the nature of the two kinds of transmission in the order TEC-LRC we can have any of the following four scenarios:

- $\alpha > 0$, $\beta < 0$, the triads close by means of attractive-attractive interactions;
- $\alpha > 0$, $\beta > 0$, the triads close by means of attractive-repulsive interactions;
- $\alpha < 0$, $\beta < 0$, the triads close by means of repulsive-attractive interactions;
- $\alpha < 0$, $\beta > 0$, the triads close by means of repulsive-repulsive interactions.

In Table 5.2 we have arranged the values of $\langle\alpha^*\rangle$ and $\langle\beta^*\rangle$ to correspond to these four classes. For instance, the networks Sawmill, Social3, Matheoremethod, Galesburg, Prison, Zachary, and Colorado (all social networks), Grassland and Bridge Brook (food webs) and Transc_yeast (a gene transcription network) close their triads following a scheme of attractive-attractive interactions. The three social networks of High Tech, Drugs and Geom as well as the networks of USAir97 (air transportation network), neurons (neural network), Ythan1 (a food web) and the Internet network, all belong to the

class of networks in which triads are closed by an attraction-repulsion mechanism. The only network with a repulsion-attraction triad closure mechanism is the social network High School, while there are seven networks closing triads with a repulsion-repulsion mechanism (three protein-protein interaction (PPIs) networks, two food webs, one animal social network, and the Roget thesaurus).

The group of networks with attractive-attractive interactions consists of 63% of all the social networks studied here. Among them we find a communication network within a small enterprise: a sawmill, where all employees were asked to indicate the frequency with which they discussed work matters with each of their colleagues on a five-point scale ranging from less than once a week to several times a day. Two employees were linked in the communication network if they rated their contact as a three or higher. This is a cooperative process in which both advisers and advisees cooperate to share the information needed for improving their skills and knowledge. Thus, closing the potential triangles in order to enhance the communication between the individuals involved seems a very reasonable mechanism. The other social networks included in this class all share a common characteristic. In the networks Social3 (a network of social contacts among college students participating in a leadership course), Galesburg (a network of friendship among physicians) and Matheoremethod (a network of friendship among school superintendents) the participants in the respective studies were asked the following questions:

- Choose the three members they wished to include in a committee;
- Nominate three doctors with whom they would choose to discuss medical matters;
- Name their best friends among the chief school administrators in Allegheny County.

In the first two cases it is very clear that the participants were asked to nominate individuals with whom they would easily cooperate, e.g., members of a committee or colleagues with whom to discuss medical matters. The third resembles a general kind of friendship relation, but the question was formulated in the context of analyzing the diffusion of a new mathematical method among High Schools in the county. Thus, selecting your best friends among other chief school administrators also means selecting those with whom you would easily cooperate in technical matters. These facts may explain the kind of attraction-attraction interaction which dictates the main mechanism for closing the triads in these networks. Transmission of information through the edges

as well as a direct long-range interaction between peers both benefit the cooperative atmosphere needed for performing the tasks for which these networks are created.

In the class of networks in which triads have been closed by attraction-repulsion mechanisms we find networks of very different natures and it is difficult to extend the previous analysis to all these networks. This group includes a social network in a small high-tech computer firm which sells, installs, and maintains computer systems, where individuals were asked: "*Who do you consider to be a personal friend?*". It could be speculated that a mechanism of the type based on Simmelian principles dominates here. That is, if $i - l - j$ is a triad and the two pairs $i - l$ and $l - j$ have strong social relations, it is natural to think that there is not a strong repulsion between i and j and they can create a new social tie. The friendship network among boys in a High School, which is the only one showing repulsion-attraction mechanisms, was created by asking the pupils: "*What fellows here in school do you go around with most often?*". The triads here are formed when the relations between the pairs $i - l$ and $l - j$ are not strong enough to tie i and j together. If the pairs $i - l$ and $l - j$ have some strong relation, i.e., if they are dating, a link between i and j could be seen as offensive to the already established couples. The final class of networks, that characterized by repulsion-repulsion interactions, does not contain any human social network. The three PPIs included in this study are characterized by this type of triad closure mechanism, together with two food webs, an animal social network, and a thesaurus. The repulsion-repulsion mechanism is characterized by weak through-edge transmission of information and weak long-range interaction between pairs of nodes separated by two adjacent edges. Thus, it is expected that the triad closure is controlled by non-topological factors, such as similarities or complimentarities among the nodes. This is a plausible explanation for the case of the PPI networks, where triads of proteins may form triangles due to their functional similarities. We notice that, as expected, the percentages of correct prediction of triad closure in this group are the smallest of the four groups. That is, the difference between the predictions made by the proposed method and the random one in this group is 8.7%, in contrasts with 15.5% for the attraction-attraction, 14.1% for the attraction-repulsion and 10% for the only network with repulsion-attraction mechanisms.

TABLE 5.2: Results of the proposed method for predicting triad closure in 25 complex networks.

Network	r	`detected`	`rand`	$\langle\alpha^*\rangle$	$\langle\beta^*\rangle$
Sawmill	18	27%	10%	1.906	−1.25
social3	32	24%	11%	1.164	−1.258
Matheoremethod	19	25%	10%	1.196	−0.574
Grassland	30	25%	9%	1.833	−1.203
Galesburg	29	23%	11%	0.902	−0.648
Prison	58	30%	12%	0.294	−1.492
Zachary	45	42%	10%	1.696	−0.392
BridgeBrook	774	13%	3%	1.977	−1.046
Colorado	17	20%	1%	0.754	−0.044
Transc_yeast	72	4%	1%	0.221	−0.544
USAir97	12181	45%	18%	1.452	0.63
High tech	77	31%	16%	0.198	0.288
Drugs	3598	27%	16%	0.526	1.048
Neurons	2808	16%	8%	0.526	0.978
Geom	12325	12%	6%	0.14	1.149
Ythan1	507	10%	4%	0.248	0.492
Internet	2331	26%	0%	0.1	1.842
High School	199	28%	18%	−0.654	−0.434
dolphins	95	24%	13%	−0.364	0.586
ScotchBroom	358	31%	4%	−0.372	0.660
StMartin	278	16%	11%	−0.232	0.335
PIN_Ecoli	478	10%	5%	−1.025	0.137
PIN_Yeast	3530	13%	4%	−1.53	1.842
PIN_Human	1047	5%	2%	−0.203	0.291
Roget	1550	7%	6%	−0.305	0.008

5.3.2 Network evolution under triadic closure

Finally, in this section we use the results of the proposed method for modeling the triadic closure evolution in a given network. In the next experiment, we simulate the evolution of a real-world complex network to examine to which extent the appearance of triadic closure can be inferred from TEC-LRC interactions. Starting from an incomplete description of the real network, we compare the effect of a sequence of triadic closures generated by our prediction methods against the present structure of the network, with respect to some well established structural indices. This method allows us to contrast the predictions made by the current method with some control parameters obtained for the real-world network in its current state. For this experiment we selected the network of injecting drug users (Drugs) for which we consider the clustering coefficient (1.9), the average path length (1.10), and the average communicability of the actual network, defined as

$$\overline{e} = \frac{TC(A) - EE(A)}{n^2 - n} = \frac{1}{n(n-1)} \sum_{i \neq j} (e^A)_{ij}.$$

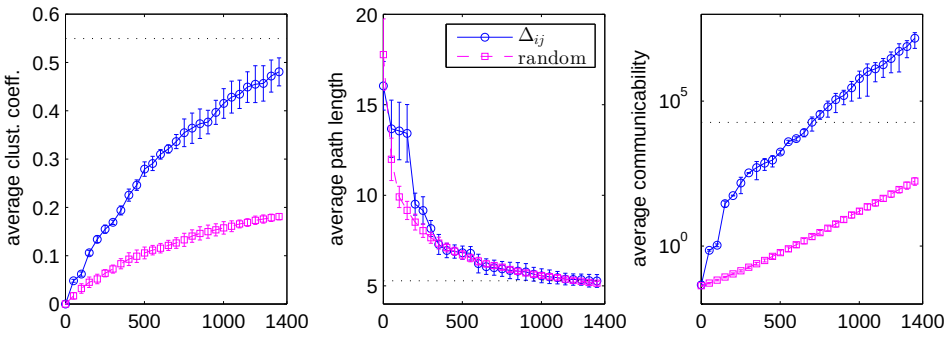
In order to perform these experiments we select 50% of the total number of triangles existing in the network and we remove one edge from each of them. Edges are selected uniformly at random among those present in the corresponding triangle. As before, let L be the list of edges obtained from the union of the potential edges and of those we removed. The values for α^* and β^* are those determined empirically using the calibration method already described (cf. Table 5.2).

The iteration process goes as follows. We select the potential edge realizing the smallest value for $\Delta_{ij}(\alpha^*, \beta^*)$ and we add this edge to the network. Then we compute the values of the parameters of interest: the average clustering coefficient, the average path length, and the average communicability. Finally, the values for $\Delta_{ij}(\alpha^*, \beta^*)$ are recomputed using the new adjacency matrix, obtained by the addition of the selected potential edge. Here every addition of an edge is considered as a time step.

This iteration is run as many times as the number of edges we have removed. That is, if we removed r edges, we consider a discrete time evolution for $0 \leq t \leq r$. We then repeat this experiment 10 times, taking the average and standard deviations of the corresponding parameter. In order to compare the results we simulate the same process by adding such edges uniformly at random.

The results of this experiment are illustrated in Figure 5.2, where we plot the values for the parameters of interest (with the corresponding error bar) versus time. The horizontal dotted line represents the actual value of the property for the original real-world network. As can be seen in Figure 5.2, the proposed method outperforms the random one for predicting the clustering coefficient of the network. The current value of $\overline{C}$ for this network is 0.549, while the one predicted by Δ_{ij} is 0.486, which contrasts with that of 0.183 obtained by the random method. We remark here that this is the most important parameter to be considered in this experiment as it is the one which accounts more directly for the ratio of triangles to paths of length two in the network. Both methods predict the average path length of the network very well, returning values very close to the actual one ($\overline{\ell} = 5.287$). In addition, the proposed method increases the average communicability of the network more significantly than the random triadic closure. This feature is important when one is interested in maximizing the total average communicability of a network, which is equivalent to increasing the quality of communication among the nodes in the network.

FIGURE 5.2: Illustration of the evolution of the clustering coefficient, average path length, and average communicability for the network of injecting drug users (Drugs) versus the number of links added using the function (Δ_{ij}) and at random (see the text for explanations).



5.4 Conclusions

The prediction of triadic closure is a very important and far from trivial problem in network theory. The fact that most real-world complex networks have more triangles than random counterparts makes the problem interesting *per se*. In addition, there is a large amount of evidence that shows that triadic closure in (social) networks is an important driver for other important structural characteristics of networks, such as degree distributions, clustering, and community structure. In this chapter, we introduced a triad closure mechanism based on communicability distances among pairs of nodes in a network. Our results show acceptable levels of predictability and interpretability of the potential mechanisms controlling triad closure in real-world networks.

Conclusions and outlook

In this thesis we have studied two different problems:

- how to make a small number of modifications to the edges in a network in order to tune its total communicability;
- how to predict triadic closure in networks.

We have developed efficient and effective techniques to select which virtual edges to add in an undirected network in order to increase as much as possible its total communicability or to select which edges to remove in order to reduce this index as little as possible. These heuristics are based on some newly introduced edge centrality measures that are used to rank the (virtual) edges in the graph. Moreover, we have shown that the total communicability is equally effective an index of network connectivity as the Helmholtz free energy, while being much cheaper to compute. We have further investigated the potentiality of our methods by studying what happens when one tries to tune the resolvent-based total communicability. We proved that, depending on whether the parameter used in the definition of this index is kept fixed or it is allowed to vary as the network is modified, different results are obtained. In neither of these two cases the results are completely satisfying. For example, we showed that when the parameter is allowed to vary, the resolvent-based total communicability can decrease under link addition and increase when one removes connections.

We have then generalized the idea of total network communicability to the case of digraphs, introducing two indices of total communicability which serve to describe the overall capability of nodes of playing the roles of broadcasters and receivers of information, respectively. We have introduced a few new edge centrality measures based on the notion of alternating walks and we have used these centrality measures to describe

edge modification techniques to tune the two indices. The modifications were aimed at the same goals as in the undirected case. When defining the generalized edge total communicability centrality, we made use of the concept of generalized matrix functions, introduced in [55]. We have developed three different approaches to approximate this edge centrality measure, which is expressed as a bilinear form involving the generalized hyperbolic sine of the adjacency matrix.

Future work in this area includes the generalization to the case of weighted networks of the edge modification techniques introduced. Moreover, object of future study on this project will be the investigation of whether these strategies can also be applied to tune the epidemic threshold of a susceptible-infected-susceptible (SIS) epidemic model [79, 92] and the synchronizability of networks [3, 17]. Indeed, concerning the former, we know that the total communicability of a network evolves similarly to a quantity which is strictly related to the epidemic threshold. Since this threshold is a critical value which separates the absorbing phase (in which the infection dies out exponentially fast) from the endemic phase (in which there is an outbreak of the disease) there is a strong interest in understanding how to best modify a few connections in the network so as to reduce the risk of outbreaks. Concerning the network synchronizability, it is well known that synchronization is an emerging phenomenon of a population of dynamically interacting entities which adjust a given property of their motion due to a coupling configuration or to external forces. In all settings, the purpose of synchrony is to strengthen the group bond. The synchronizability of a network is measured by looking at some spectral properties of the graph Laplacian. Preliminary studies suggested that our techniques can be adapted to tune this index.

Concerning the problem of triadic closure in complex networks, we proposed a strategy for predicting triadic closure based on the usage of communicability distance functions. We studied 25 real world networks and showed that our technique outperforms the random mechanism, which was previously used in the literature. In the future, we plan to extend this result to the case of multiplexes. A multiplex is a structure formed by several layers, each of which represents different types of interactions among the same nodes. As an example of this type of structure one may consider a set of users that have profiles on Facebook, Twitter, and Instagram. Then each layer of this multiplex contains the same set of nodes, but the interactions among them describe the connections among the users in the three different social networks: one layer describes the interactions on

Facebook, one that on Twitter, and one the relationships on Instagram. The idea of communicability has already been extended by other authors to this type of networks, and thus we aim to extend the methodology developed to the detection of triadic closure in multiplexes.

Appendix A

Datasets

In this Appendix we shall describe the networks used in the numerical tests.

Brain networks

- Neurons: Neuronal synaptic network of the nematode *C. elegans*. Includes all data except muscle cells and uses all synaptic connections [76, 97].

Ecological networks

- BridgeBrook: pelagic species from the largest of a set of 50 New York Adirondack lake food webs [80];
- Grassland: all vascular plants and all insects and trophic interactions found inside stems of plants collected from 24 sites distributed within England and Wales [70];
- ScotchBroom: trophic interactions among herbivores, parasitoids, predators, and pathogens associated with broom, *Cytisus scoparius*, collected in Silwood Park, Berkshire, England, UK [71];
- StMartin: birds and predators and arthropod prey of Anolis lizards on the island of St. Martin, located in the northern Lesser Antilles [69];
- Ythan1: mostly birds, fishes, invertebrates, and metazoan parasites in a Scottish Estuary [58].

Informational networks

- Roget: (symmetrized) vocabulary network of words related by their definitions in Roget Thesaurus of English. Two words are connected if one is used in the definition of the other [51].

PPI networks

- PIN_Ecoli: protein-protein interaction network in *Escherichia coli* [21];
- PIN_Human: protein-protein interaction network in human [83];
- PIN_Yeast: protein-protein interaction network in *S. cerevisiae* (yeast) [20, 95].

Social and economic networks

- ca-HepTh: the collaboration network in the arXiv group of High Energy Physics Theory [26];
- Colorado: the risk network of persons with HIV infection during its early epidemic phase in Colorado Spring, USA, using analysis of community wide HIV/AIDS contact tracing records (sexual and injecting drugs partners) from 1985-1999 [81];
- Dolphins: social network of frequent association between 62 bottlenose dolphins living in the waters off New Zealand [66];
- Drugs: social network of injecting drug users (IDUs) that have shared a needle in the last six months [1].
- Erdös02: Erdös collaboration network, Erdös included [26];
- Galesburg: friendship ties among 31 physicians [24, 27, 61];
- Geom: collaboration network of scientists in the field of Computational Geometry [9];
- High School: network of relations in a high school. The students choose the three members they wanted to have in a committee [102];
- High tech: friendship ties among the employees in a small high-tech computer firm which sells, installs, and maintain computer systems [27, 62];

- Matheoremethod: network that concerns the diffusion of a new mathematics method in the 1950s. It traces the diffusion of the modern mathematical method among school systems that combine elementary and secondary programs in Allegheny County (Pennsylvania, U.S.) [22, 27];
- Prison: social network of inmates in prison who chose "What fellows on the tier are you closest friends with?" [67];
- Sawmill: social communication network within a sawmill, where employees were asked to indicate the frequency with which they discussed work matters with each of their colleagues [27, 73];
- social3: social network among college students in a course about leadership. The students choose which three members they wanted to have in a committee [102];
- Zachary: social network of friendship among the members of a karate club [101].

Technological networks

- Internet: the Internet at the Autonomous System (AS) level as of September 1997 and of April 1998 [43];
- USAir97: airport transportation network between airports in US in 1997 [9];
- as-735: communication network of a group of autonomous system (AS) measured over 735 days between November 8, 1997 and January 2, 2000. Communication occurs when routers from two ASs exchange information [26];
- as-22july06: (symmetrized) structure of Internet routers as of July 22, 2006 [26].

Road networks

- Minnesota: the road network of Minnesota [26];
- usroad-48: the continental US road network [26].

Transcription networks

- Transc_yeast: direct transcriptional regulation between genes in *Saccaromyces cerevisiae* [75, 76].

Directed networks

- GD95b: direct network representing entries in a graph drawing context [26].
- cit-HepTh: citation network between papers in the arXiv group of High Energy Physics Theory [26].
- Abortion: web graph consisting of web sites on the topic of abortion [90].
- Computational Complexity: web graph consisting of web sites on the topic of computational complexity [90].
- Twitter: network of mentions and retweets of some part of the social network Twitter [2].
- Roget: Roget's Thesaurus, 1879 [9, 26].
- SLASHDOT: Slashdot social network from February 2009 [26].
- ITwiki: Wikipedia structure in Italian as of May 2005 [77].

Bibliography

[1] *Data for this project were provided, in part, by nih grants da12831 and hd41877, and copies can be obtained from james moody.* `moody.77@sociology.osu.edu`.

[2] *Gephi sample dataset.* `http://wiki.gephi.org/index.php/Datasets`.

[3] A. Arenas, A. Díaz-Guilera, J. Kurths, Y. Moreno, and C. Zhou, *Synchronization in complex networks*, Physics Reports, 469 (2008), pp. 93–153.

[4] F. Arrigo and M. Benzi, *Updating and downdating techniques for optimizing network communicability*, SIAM Journal on Scientific Computing, 38 (2016), pp. B25–B49.

[5] ——, *Edge modification criteria for enhancing the communicability of digraphs*, SIAM Journal on Matrix Analysis and Applications, (in press).

[6] F. Arrigo, M. Benzi, and C. Fenu, *Computation of generalized matrix functions*, arXiv preprint arXiv:1512.01446v2, (2015).

[7] R. B. Bapat, *Graphs and matrices*, Springer, 2010.

[8] A.-L. Barabási and R. Albert, *Emergence of scaling in random networks*, Science, 286 (1999), pp. 509–512.

[9] V. Batagelj and A. Mrvar, *Pajek datasets.* `http://vlado.fmf.uni-lj.si/pub/networks/data/`.

[10] M. Benzi and P. Boito, *Quadrature rule-based bounds for functions of adjacency matrices*, Linear Algebra and its Applications, 433 (2010), pp. 637–652.

[11] M. Benzi, E. Estrada, and C. Klymko, *Ranking hubs and authorities using matrix functions*, Linear Algebra and its Applications, 438 (2013), pp. 2447–2474.

[12] M. Benzi and G. H. Golub, *Bounds for the entries of matrix functions with applications to preconditioning*, BIT Numerical Mathematics, 39 (1999), pp. 417–438.

[13] M. Benzi and C. Klymko, *Total communicability as a centrality measure*, Journal of Complex Networks, 1 (2013), pp. 124–149.

[14] M. Benzi and C. Klymko, *On the limiting behavior of parameter-dependent network centrality measures*, SIAM Journal on Matrix Analysis and Applications, 36 (2015), pp. 686–706.

[15] M. W. Berry, T. P. Chartier, K. R. Hutson, and A. N. Langville, *Identifying influential edges in a directed network: big events, upsets and non-transitivity*, Journal of Complex Networks, 2 (2013), pp. 87–109.

[16] A. Beygelzimer, G. Grinstein, R. Linsker, and I. Rish, *Improving network robustness by edge modification*, Physica A: Statistical Mechanics and its Applications, 357 (2005), pp. 593–612.

[17] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D.-U. Hwang, *Complex networks: Structure and dynamics*, Physics reports, 424 (2006), pp. 175–308.

[18] P. Bonacich, *Power and centrality: A family of measures*, American Journal of Sociology, 92 (1987), pp. 1170–1182.

[19] R. A. Brualdi, F. Harary, and Z. Miller, *Bigraphs versus digraphs via matrices*, Journal of Graph Theory, 4 (1980), pp. 51–73.

[20] D. Bu, Y. Zhao, L. Cai, H. Xue, X. Zhu, H. Lu, J. Zhang, S. Sun, L. Ling, N. Zhang, et al., *Topological structure analysis of the protein–protein interaction network in budding yeast*, Nucleic acids research, 31 (2003), pp. 2443–2450.

[21] G. Butland, J. M. Peregrín-Alvarez, J. Li, W. Yang, X. Yang, V. Canadien, A. Starostine, D. Richards, B. Beattie, N. Krogan, et al., *Interaction network containing conserved and essential protein complexes in escherichia coli*, Nature, 433 (2005), pp. 531–537.

[22] R. O. Carlson, *Adoption of educational innovations*, Center for the Advanced Study of Educational Administration, University of Oregon, (1965), p. 19.

[23] H. Chan, L. Akoglu, and H. Tong, *Make it or break it: Manipulating robustness in large networks*, Proceedings of the 2014 SIAM Data Mining Conference, (2014), pp. 325–333.

[24] J. S. Coleman, E. Katz, H. Menzel, et al., *Medical innovation: A Diffusion Study*, Bobbs-Merrill Indianapolis, 1966.

[25] J. J. Crofts, E. Estrada, D. J. Higham, and A. Taylor, *Mapping directed networks*, Electronic Transactions on Numerical Analysis, 37 (2010), pp. 337–350.

[26] T. Davis and Y. Hu, *The University of Florida Sparse Matrix Collection.* `http://www.cise.ufl.edu/research/sparse/matrices/.`

[27] W. De Nooy, A. Mrvar, and V. Batagelj, *Exploratory social network analysis with Pajek*, Cambridge University Press, 2004.

[28] N. Del Buono, L. Lopez, and R. Peluso, *Computation of the exponential of large sparse skew-symmetric matrices*, SIAM Journal on Scientific Computing, 27 (2005), pp. 278–293.

[29] N. Del Buono, L. Lopez, and T. Politi, *Computation of functions of Hamiltonian and skew-symmetric matrices*, Mathematics and Computers in Simulation, 79 (2008), pp. 1284–1297.

[30] A. L. Dulmage and N. S. Mendelsohn, *Coverings of bipartite graphs*, Canadian Journal of Mathematics, 10 (1958), pp. 516–534.

[31] A. L. Dulmage and N. S. Mendelsohn, *Graphs and matrices*, Graph Theory and Theoretical Physics, (1967), pp. 167–227.

[32] E. Estrada, *Characterization of 3d molecular structure*, Chemical Physics Letters, 319 (2000), pp. 713–718.

[33] ——, *Spectral scaling and good expansion properties in complex networks*, Europhysics Letters, 73 (2006), pp. 649–655.

[34] ——, *The Structure of Complex Networks: Theory and Applications*, Oxford University Press, 2011.

[35] ——, *The communicability distance in graphs*, Linear Algebra and its Applications, 436 (2012), pp. 4317–4328.

[36] ——, *Complex networks in the Euclidean space of communicability distances*, Physical Review E, 85 (2012), p. 066122.

[37] E. Estrada and F. Arrigo, *Predicting triadic closure in networks using communicability distance functions*, SIAM Journal on Applied Mathematics, 75 (2015), pp. 1725–1744.

[38] E. Estrada and N. Hatano, *Statistical-mechanical approach to subgraph centrality in complex networks*, Chemical Physics Letters, 439 (2007), pp. 247–251.

[39] ——, *Communicability in complex networks*, Physical Review E, 77 (2008), p. 036111.

[40] E. Estrada, N. Hatano, and M. Benzi, *The physics of communicability in complex networks*, Physics Reports, 514 (2012), pp. 89–119.

[41] E. Estrada and D. J. Higham, *Network properties revealed through matrix functions*, SIAM Review, 52 (2010), pp. 696–714.

[42] E. Estrada and J. A. Rodríguez-Velázquez, *Subgraph centrality in complex networks*, Physical Review E, 71 (2005), p. 056103.

[43] M. Faloutsos, P. Faloutsos, and C. Faloutsos, *On power-law relationships of the internet topology*, 29 (1999), pp. 251–262.

[44] C. Fenu, D. Martin, L. Reichel, and G. Rodriguez, *Network analysis via partial spectral factorization and Gauss quadrature*, SIAM Journal on Scientific Computing, 35 (2012), pp. A2046–A2068.

[45] H. Frank and I. T. Frisch, *Analysis and design of survivable networks*, IEEE Transactions on Communication Technology, 18 (1970), pp. 501–519.

[46] L. C. Freeman, *A set of measures of centrality based on betweenness*, Sociometry, (1977), pp. 35–41.

[47] C. Godsil and G. F. Royle, *Algebraic graph theory*, vol. 207, Springer Science & Business Media, 2013.

[48] G. Golub and W. Kahan, *Calculating the singular values and pseudo-inverse of a matrix*, SIAM Journal on Numerical Analysis, 2 (1965), pp. 205–224.

[49] G. H. Golub and G. Meurant, *Matrices, Moments and Quadrature with Applications*, Princeton University Press, 2010.

[50] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Third ed., 2012.

[51] P. Gutenberg, *Roget's thesaurus of english words and phrases.* `http://www.gutenberg.org/etxt/22`.

[52] S. Güttel, *funm_kryl toolbox for MATLAB.* `www.mathe.tu-freiberg.de/~guttels/funm_kryl/`.

[53] M. Hanke, J. Nagy, and R. Plemmons, *Preconditioned iterative regularization*, Numerical linear algebra. Proceedings of the Conference in Numerical Linear Algebra and Scientific Computation, (1993), pp. 141–163.

[54] P. E. Hart, N. J. Nilsson, and B. Raphael, *A formal basis for the heuristic determination of minimum cost paths*, IEEE Transactions on Systems Science and Cybernetics, 4 (1968), pp. 100–107.

[55] J. B. Hawkins and A. Ben-Israel, *On generalized matrix functions*, Linear and Multilinear Algebra, 1 (1973), pp. 163–171.

[56] N. J. Higham, *Functions of Matrices: Theory and Computation*, Society for Industrial and Applied Mathematics, 2008.

[57] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge University Press, Second ed., 2012.

[58] M. Huxham, S. Beaney, and D. Raffaelli, *Do parasites reduce the chances of triangulation in a real food web?*, Oikos, (1996), pp. 284–300.

[59] L. Katz, *A new status index derived from sociometric analysis*, Psychometrika, 18 (1953), pp. 39–43.

[60] J. M. Kleinberg, *Authoritative sources in a hyperlinked environment*, Journal of the ACM (JACM), 46 (1999), pp. 604–632.

[61] D. Knoke and R. S. Burt, *Prominence*, Applied Network Analysis, (1983), pp. 195–222.

[62] D. Krackhardt, *The ties that torture: Simmelian tie analysis in organizations*, Research in the Sociology of Organizations, 16 (1999), pp. 183–210.

[63] D. Krackhardt and M. S. Handcock, *Heider vs simmel: Emergent features in dynamic structures*, Statistical Network Analysis: Models, Issues, and New Directions, (2007), pp. 14–27.

[64] L. Lopez and A. Pugliese, *Decay behaviour of functions of skew-symmetric matrices*, Proceedings of the 7th Hellenic-European Conference on Computer Mathematics and Applications, (2005), pp. 22–24.

[65] V. H. Louzada, F. Daolio, H. J. Herrmann, and M. Tomassini, *Smart rewiring for network robustness*, Journal of Complex Networks, 1 (2013), pp. 150–159.

[66] D. Lusseau, K. Schneider, O. J. Boisseau, P. Haase, E. Slooten, and S. M. Dawson, *The bottlenose dolphin community of Doubtful Sound features a large proportion of long-lasting associations*, Behavioral Ecology and Sociobiology, 54 (2003), pp. 396–405.

[67] D. MacRae, *Direct factor analysis of sociometric data*, Sociometry, 23 (1960), pp. 360–371.

[68] J. R. Magnus and H. Neudecker, *Matrix Differential Calculus with Applications in Statistics and Econometrics*, John Wiley & Sons, 1988.

[69] N. D. Martinez, *Artifacts or attributes? Effects of resolution on the Little Rock Lake food web*, Ecological Monographs, 61 (1991), pp. 367–392.

[70] N. D. Martinez, B. A. Hawkins, H. A. Dawah, and B. P. Feifarek, *Effects of sampling effort on characterization of food-web structure*, Ecology, 80 (1999), pp. 1044–1055.

[71] J. Memmott, N. Martinez, and J. Cohen, *Predators, parasitoids and pathogens: species richness, trophic generality and body sizes in a natural food web*, Journal of Animal Ecology, 69 (2000), pp. 1–15.

[72] G. Meurant, *MMQ toolbox for MATLAB.* `http://pagesperso-orange.fr/gerard.meurant/.`

[73] J. H. Michael and J. G. Massey, *Modeling the communication network in a sawmill*, Forest Products Journal, 47 (1997), pp. 25–30.

[74] K. S. Miller and S. G. Samko, *Completely monotonic functions*, Integral Transforms and Special Functions, 12 (2001), pp. 389–402.

[75] R. Milo, S. Itzkovitz, N. Kashtan, R. Levitt, S. Shen-Orr, I. Ayzenshtat, M. Sheffer, and U. Alon, *Superfamilies of evolved and designed networks*, Science, 303 (2004), pp. 1538–1542.

[76] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon, *Network motifs: Simple building blocks of complex networks*, Science, 298 (2002), pp. 824–827.

[77] L. Muchnik, *Lev Muchnik's data sets web page.* `http://www.levmuchnik.net/Content/Networks/NetworkData.html`.

[78] M. E. J. Newman, *The structure and function of complex networks*, SIAM Review, 45 (2003), pp. 167–256.

[79] ——, *Networks: an Introduction*, Oxford University Press, 2010.

[80] G. A. Polis and D. R. Strong, *Food web complexity and community dynamics*, American Naturalist, (1996), pp. 813–846.

[81] J. Potterat, L. Phillips-Plummer, S. Muth, R. Rothenberg, D. Woodhouse, T. Maldonado-Long, H. Zimmerman, and J. Muth, *Risk network structure in the early epidemic phase of hiv transmission in colorado springs*, Sexually transmitted infections, 78 (2002), pp. i159–i163.

[82] D. Puder, *Expansion of random graphs: new proofs, new results*, Inventiones mathematicae, (2014), pp. 1–64.

[83] J.-F. Rual, K. Venkatesan, T. Hao, T. Hirozane-Kishikawa, A. Dricot, N. Li, G. F. Berriz, F. D. Gibbons, M. Dreze, N. Ayivi-Guedehoussou, et al., *Towards a proteome-scale map of the human protein–protein interaction network*, Nature, 437 (2005), pp. 1059–1069.

[84] P. Sen and B. K. Chakrabarti, *Sociophysics: an Introduction*, Oxford University Press, 2014.

[85] B. Shargel, H. Sayama, I. R. Epstein, and Y. Bar-Yam, *Optimization of robustness and connectivity in complex networks*, Physical review letters, 90 (2003), p. 068701.

[86] G. Simmel, *Conflict and the Web of Group Affiliations*, Free Press, 1992.

[87] A. Taylor and D. J. Higham, *CONTEST: Toolbox files and documentation.* `http://www.mathstat.strath.ac.uk/research/groups/numerical_analysis/-contest/toolbox.`

[88] A. Taylor and D. J. Higham, *CONTEST: A controllable test matrix toolbox for matlab*, ACM Transactions on Mathematical Software, 35 (2009), pp. 26:1–26:17.

[89] H. Tong, B. A. Prakash, T. Eliassi-Rad, M. Faloutsos, and C. Faloutsos, *Gelling, and melting, large graphs by edge manipulation*, Proceedings of the 21st ACM international conference on information and knowledge management, (2012), pp. 245–254.

[90] P. Tsapras, *Dataset for experiments on link analysis ranking algorithms.* `http://www.cs.toronto.edy/≈tsap/experiments/datasets/index.html/.`

[91] P. Van Mieghem, *Graph Spectra for Complex Networks*, Cambridge University Press, 2011.

[92] P. Van Mieghem, D. Stevanović, F. Kuipers, C. Li, R. Van De Bovenkamp, D. Liu, and H. Wang, *Decreasing the spectral radius of a graph by link removals*, Physical Review E, 84 (2011), p. 016101.

[93] R. S. Varga, *p-cyclic matrices: A generalization of the Young–Frankel successive overrelaxation scheme.*, Pacific Journal of Mathematics, 9 (1959), pp. 617–628.

[94] ——, *Matrix iterative analysis*, vol. 27, Springer Science & Business Media, 2009.

[95] C. Von Mering, R. Krause, B. Snel, M. Cornell, S. G. Oliver, S. Fields, and P. Bork, *Comparative assessment of large-scale data sets of protein–protein interactions*, Nature, 417 (2002), pp. 399–403.

[96] D. J. Watts and S. H. Strogatz, *Collective dynamics of small-world networks*, Nature, 393 (1998), pp. 440–442.

[97] J. White, E. Southgate, J. Thomson, and S. Brenner, *The structure of the nervous system of the nematode caenorhabditis elegans: the mind of a worm*, Philosophical Transactions of the Royal Society B, 314 (1986), pp. 1–340.

[98] D. V. Widder, *The Laplace Transform.*, Princeton University Press, 1946.

[99] J. Wu, M. Barahona, Y.-j. Tan, and H.-z. Deng, *Robustness of random graphs based on graph spectra*, Chaos: An Interdisciplinary Journal of Nonlinear Science, 22 (2012), p. 043101.

[100] J. Wu, M. Barahona, T. Yue-Jin, and D. Hong-Zhong, *Natural connectivity of complex networks*, Chinese Physics Letters, 27 (2010), p. 078902.

[101] W. W. Zachary, *An information flow model for conflict and fission in small groups*, Journal of Anthropological Research, 33 (1977), pp. 452–473.

[102] L. D. Zeleny, *Adaptation of research findings in social leadership to college classroom procedures*, Sociometry, 13 (1950), pp. 314–328.