Applications of random walks on complex networks: structural and dynamical insights



Erik Hormann
St Hilda's College
University of Oxford

A thesis submitted for the degree of
Doctor of Philosophy

Trinity 2024

Contents

1 Introduction **3**

1.1 Graph structure and dynamics 6

1.2 Outline and contributions 7

2 Preliminaries **9**

2.1 Graphs 10

2.1.1 Models of random graphs 13

2.2 Random walks on graphs 17

2.2.1 Return time distributions 23

2.2.2 Generating functions 24

2.3 Graph matrices 26

2.3.1 Singular value decomposition (SVD) 28

2.4 Community detection 29

2.4.1 The SBM phase transition 30

2.4.2 InfoMap 32

2.4.3 Normalised Mutual Information 33

2.5 Message passing 35

2.5.1 BP algorithm 36

2.5.2 Population dynamics algorithm 38

2.6 Graph percolation 39

3 Ergodic sets and network compression **42**

3.1 Idea and informal introduction 45

3.2 Erogdic sets: structural definition in directed networks 48

3.3 Ergodic Sets Compression Algorithm (ESCA) 57

3.3.1 Step I: ergodic sets detection 58

3.3.2 Ergodic sets coarse-graining rules 60

3.3.3 Step II: core SVD compression 64

3.4 Generalised ergodic sets in random networks . . . 67
3.5 ESCA compression of real-world networks . . . 69
3.6 Core-periphery analysis . . . 83
3.7 Discussion . . . 87

4 Return time distributions **88**
4.1 Random walks and Kac's lemma . . . 89
4.2 Approximating first return time distributions . . . 93
4.2.1 Heterogeneous mean-field approximation . . . 93
4.2.2 Tree approximation . . . 95
4.2.3 Combined approximation . . . 98
4.2.4 Approximation for networks with short cycles . . . 99
4.3 Numerical simulations . . . 103
4.3.1 Computing return time distributions . . . 103
4.3.2 Numerical results . . . 107
4.4 Implications on community detection algorithm . . . 111
4.4.1 Micro-canonical SBM model . . . 112
4.4.2 FRTD across the SBM phase transition . . . 117
4.4.3 Analytical analysis of the InfoMap function . . . 119
4.5 Discussion . . . 126

5 Random Cluster Similarity **130**
5.1 The similarity matrix . . . 133
5.2 Discussion . . . 141

6 Conclusion and outlook **143**

Bibliography **147**

List of Figures

1.1 A visualisation (artificial colours) of the network representing the complex topology of the internet. From the OPTE Project [1]. 4

2.1 Examples of instances of the four random graph models discussed. From the top left, in clockwise direction: Erdős-Rényi random graph $ER(100, 0.07)$, SBM model $SBM(100, 0.15, 0.01)$, a Watts-Strogatz model $WS(4)$, and a Barabasi-Albert Model $BA(100, 1.5)$. 17

3.1 Example of simple ergodic sets $\{A, B\}$ and $\{C, D\}$ that do not satisfy the conditions of being a sink or a source. Here GCC stands for the graph's giant connected component, a strongly connected subgraph that contains most of the nodes of the network. No individual node in the graphs would satisfy definition 5 of source or well node. 46

3.2 Ergodic sets do not account for all vertices in a graph. Node A, for example, sitting along the path from one ergodic set X (read: strongly connected component for now, see section 3.2) to the giant (strongly) connected component. 47

3.3 Graphical representation of the input and output sets in relation to the bow-tie representation of the World Wide Web. The inputs sets are represented in blue, while the output sets are represented in red. . 47

3.4 Counterexample. Both X and GCC are strongly connected components. 50

3.5 A representation of a generic network division. **Top:** the classical *bow-tie* structure [2]. **Bottom:** the division in ergodic sets. BW: backward ergodic set, FW: forward ergodic set. 74

3.6 Graphical representation of the weight assignment rules for the output sets. In dark colours the old vertices of X, in pale colours the new coarse-grained vertex n_X. Old edges before the rewiring are shown as dotted, the new edges inserted by the *ESCA* algorithm are shown in solid lines. 75

3.7 Graphical representation of the weight assignment rules for the input sets. In dark colours the old vertices of X, in pale colours the new coarse-grained vertex n_X. Old edges before the rewiring are shown as dotted, the new edges inserted by the *ESCA* algorithm are shown in solid lines. 75
3.8 Graphical representation of the weight assignment rules for the input sets. In dark colours the old vertices of X, in pale colours the new coarse-grained vertex n_X. Old edges before the rewiring are shown as dotted, the new edges inserted by the *ESCA* algorithm are shown in solid lines. 76
3.9 The resulting network at the end of the first compression step of the Ergodic Sets Compression Algorithm. Each (now proper) source is the result of the coarse-graining of an input set, while each (proper) is the result of the coarse-graining of an output set. 76
3.10 Average number of ergodic sets [inputs shown] for various choices of (N, p). Here, $N_{iter} = 1000$ for each sample. 77
3.11 Number of nodes belonging to the largest ergodic set averaged over $N_{samples} = 1000$ instances of an Erdős-Rényi model $ER(N, p)$. The dependence on p is on the x axis, while different sizes N corresponds to different data series. 77
3.12 Number of nodes belonging to *any* ergodic set averaged over $N_{samples} = 1000$ instances of an Erdős-Rényi model $ER(N, p)$. The dependence on p is on the x axis, while different sizes N corresponds to different data series. 78
3.13 Number of nodes belonging to *any* ergodic set averaged over $N_{samples} = 1000$ instances of an SBM model $SBM(N, p_{in}, p_{out})$. The dependence on $p = \frac{p_{in}+p_{out}}{2}$ is on the x axis, while different sizes N corresponds to different data series. 78
3.14 The original synthetic graph. The SBM core with its two communities is clearly visible, with vertices labelled from 0 to 39. The attachments of the inputs and outputs were randomly generated, including the possibility of multiple attachments (note vertex 46, attached to both vertex 7 and 19). 79

3.15 Number of inputs and outputs compression in the dataset of real-world networks, as function of the total number of vertices in the network. Data are arranged as input and output data and color-coded based on the application of origin. Axes in log-log scale. 80
3.16 Total number of nodes compressed by the ESCA algorithm versus the original network size. Data are arranged as input and output data and color-coded based on the application of origin. Axes in log-log scale. . 81
3.17 Compression rates for graphs in the dataset, showing both the first ESCA step compression factor C_1 and the total compression rate C_2. 82
3.18 Plots of the symmetrised input-output connection matrix $\mathcal{B} = B^T B$. To the left the plots for the actual networks, to the right the plots for the same networks with rewired cores. By row, from top to bottom, the networks represent the websites of the states of California, Oregon, and Utah. 86

4.1 Geodesic neighbourhoods $N_i^{(r)}$ versus *message passing* neighbourhoods $\mathcal{N}_i^{(r)}$. On the left, geodesic neighbourhoods around i (in red) of order one (in blue) and two (in green). On the right, message passing neighbourhood of order $r = 3$ (in purple), with the generating cycle highlighted (in red). 100
4.2 First return time distribution, $Y(t)$, for random walks on a 6-regular random graph with $n = 2^{15}$ nodes. The ground truth is computed for an arbitrary node with matrix iteration. The approximation is computed from derivatives of Eq. (4.59) at $z = 0$. Note, the tail of the distribution is straight on a semi-log plot, indicating exponential decay. The slope of this tail on a logarithmic scale is the inverse of the quantity we call the tail-mean, approximate by Eq. (4.30). 108
4.3 First return time distribution approximations against ground truth, for random regular graph of various sizes. The exponential tail of randomly chosen nodes in networks of differing sizes was compared to approximations. As the number of nodes increases, the tree approximation ($r = 0$) moves towards the line $y = x$, showing that the approximation is converging to the truth. For the $r = 3$ approximation, the points are close to the line $y = x$ even for small N. 109

4.4 First return time distribution approximations against ground truth, for regular stochastic block models of varying size but with a fixed community structure. The exponential tail of randomly chosen nodes in networks of differing sizes was compared to approximations. As the number of nodes increases, the tree approximation ($r = 0$) moves towards the line $y = x$, showing that the approximation is converging to the truth. For the $r = 3$ approximation, the points are close to the line $y = x$ even for small n. 110
4.5 First return time distribution approximation against truth for regular stochastic block models of various sizes variable number of communities $N_C = N/20$. The exponential tail of randomly chosen nodes in networks of differing sizes was compared to approximations. The predictions for the tree approximation are not very close to the line $y = x$, even as n increases. Conversely, the predictions for $r = 3$ remain close to $y = x$ for all n. 112
4.6 Variance of the return time distribution, for different values of the α parameter, as described in section 4.4.1. Other parameters are the graph size (n, m) and the number of communities. The baseline is the Erdős-Rényi graph, for which the variance decays exponentially as a function of the node degree. The results for the SBM deviate more strongly when the number of communities is higher, even for equal values of the r parameter. 118
4.7 Normalised mutual information between the planted clusters of a micro-canonical SBM model and the clusters returned by the InfoMap algorithm, as a function of the network size N. Each data point has been averaged over 200 iterations of the micro-canonical SMB model with the same parameter, and the error bar indicate one standard deviation. 120
4.8 Plots of $\Delta L(\tau, \alpha, C)$ as a function of τ, α and with $C = 4$. Note how the correction term is negative but independent on the network size N. 126

5.1 Similarity matrix study for a degree-corrected SBM with power law degree distribution. The graph has been generated the parameters $N = 200$ and $\gamma = 3$, and has two communities of equal size. We report (by column) the value of $A(p, q)$, the share of pairs in the upper $\mathcal{P}$ percentile having the degree and community property (that is, $\frac{|\mathcal{C}|}{|\mathcal{P}|}$. Different values for the percentile $\mathcal{P}$ are reported (by row). 137

5.2 Similarity matrix study for a degree-corrected SBM with power law degree distribution. The graph has been generated using the parameters $N = 300$ and $\gamma = 3$, and has two communities of equal size. We report (by column) the value of $A(p, q)$, the share of pairs in the upper $\mathcal{P}$ percentile having the degree and community property (that is, $\frac{|\mathcal{C}|}{|\mathcal{P}|}$. Different values for the percentile $\mathcal{P}$ are reported (by row). 138

5.3 Similarity matrix study for a degree-corrected SBM with power law degree distribution. The graph has been generated using the parameters $N = 400$ and $\gamma = 3$, and has two communities of equal size. We report (by column) the value of $A(p, q)$, the share of pairs in the upper $\mathcal{P}$ percentile having the degree and community property (that is, $\frac{|\mathcal{C}|}{|\mathcal{P}|}$. Different values for the percentile $\mathcal{P}$ are reported (by row). 139

5.4 Analysis of the three graphs studied in figures 5.1 to 5.3, By row, adjacency matrix, degree distribution, and three samples of the similarity matrix from different values of decreasing p at fixed q. 140

Statement of authorship and originality

Chapter 2 contains expository introductions, unified from several referenced sources for the reader. All results are either lifted from the literature or minimally adapted to suit successive developments.

The work in chapter 3 is original and has been developed in collaboration with Renaud Lambiotte. Some results are adaptations of previously known work and have been clearly marked in the text. The algorithm, the analyses, and the numerical simulation results don't appear, to the best of my knowledge, in any previously published scientific publication.

Chapter 4 contains original work developed in collaboration with Renaud Lambiotte and George T. Cantwell. The original research question was formulated by George T. Cantwell, but it received substantial modification during the study. The theoretical results are a collaboration with George T. Cantwell, while I was solely responsible for the numerical simulations.

Chapter 5 contains original results obtained in collaboration with Renaud Lambiotte, Nicola Pedreschi, Gaogao Dong, and Xiangyi Meng. My contributions pertain to the results of the numerical simulations, and a contribution to the result discussion. The discussion in section 5.2, including the limit in equation (5.12) is my own original work.

Papers related to this thesis

Much of the original materials in this thesis are either published as journal articles and conference presentations or are in the process of being converted into journal publications. More specifically, the published work related to this thesis include:

- much of chapter 3 is also included in a paper under preparation, provisionally titled *Ergodic Sets Compression Algorithm (ESCA): a dynamics-preserving coarse-graining for complex directed networks*, authored by me and R. Lambiotte.

- part of the materials in 4 (particularly sections 4.1 to 4.3 and 4.5) also appear in a paper authored by me, R. Lambiotte, G. Cantwell, which is titled *An approximation for return time distributions of random walks on sparse networks* and has been submitted to *Physical Review E*.

The following publication on opinion dynamics and the voter model contains contributions deriving from work undertaken during my DPhil but such topic is not included in this thesis:

- Papanikolaou, N., Vaccario, G., Hormann, E., Lambiotte, R., & Schweitzer, F. (2022). "Consensus from group interactions: An adaptive voter model on hypergraphs." *Physical Review E*, 105(5), 054307. [3]

Chapter 1

Introduction

Humanity has always been interested in – and fascinated by – the study of the world and the discovery of its inner workings, from the motion of the stars in the night sky in ancient Babylon all the way to the interconnected communication networks that span our globe since the rise of the information era. Our ability to craft manufacts of increasing complexity has represented an inflection point in this progression: while science used to be confined to explaining objects and complexities of natural origin, nowadays scientists increasingly turn their attention to explaining the behaviour of man-made phenomena.

The driving force behind this paradigm shift is the human ability to products to interface, communicate, and influence each other, all without the need of direct human intervention. For example, both the steam engine and the computer have been transformative, paradigm-shifting inventions that defined their respective eras and propelled humanity into a new stage of civilisation. However, despite both being equally defining for their times, they are fundamentally different when considered in a group. The production output of a number of steam engines is merely the sum of the outputs of the single machines. Computers, on the other hand, can connect, exchange information, and provide a much more useful and complex experience than

a set of computers taken individually. A single computer, taken in isolation, provide limited usefulness; in contrast, when connected to the internet, it becomes part of a worldwide, interconnected system capable of performing tasks and providing experiences far beyond what any set of disconnected machines could achieve. This is the key idea behind the rise of complex systems, sets of homogeneous (or mostly homogeneous) individual elements exhibiting collective behaviours characterised by phenomena broader and more faceted than the sum of the behaviours of each individual taking part in the system.

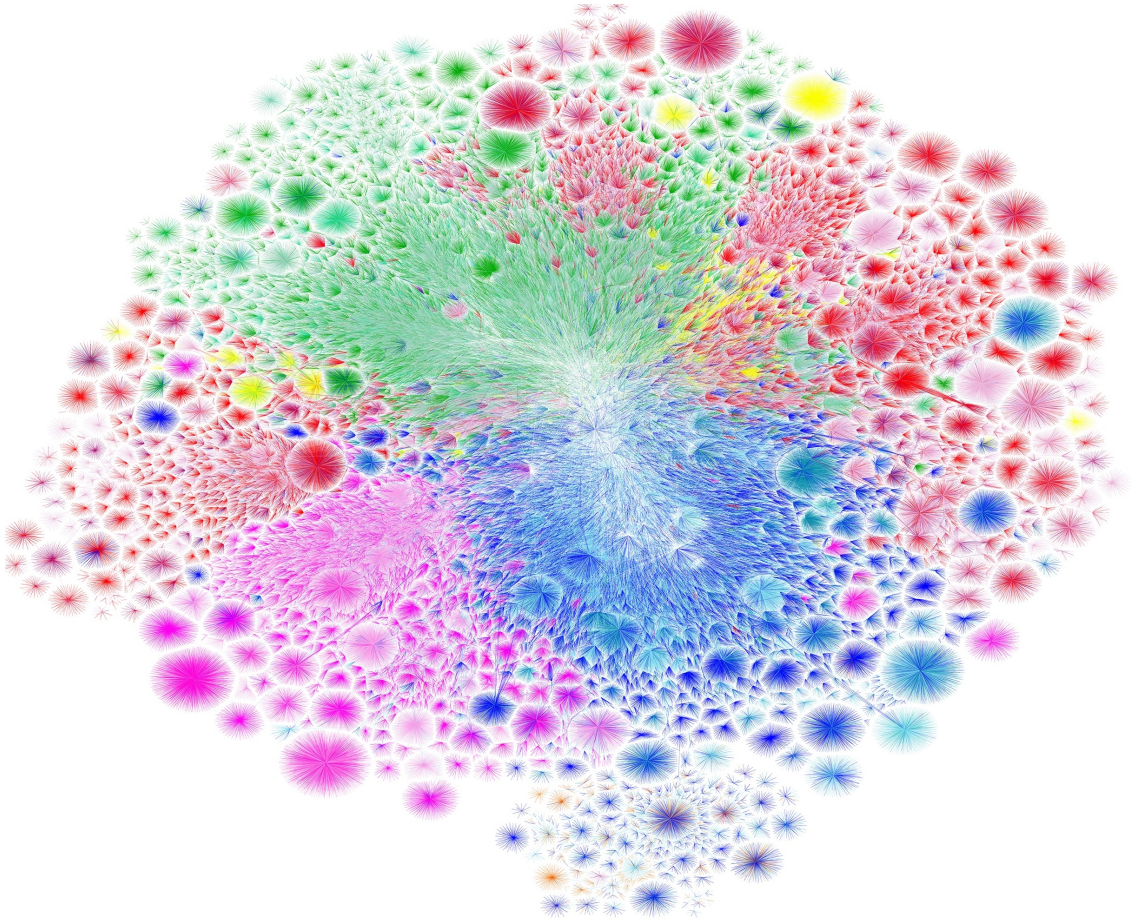


Figure 1.1: A visualisation (artificial colours) of the network representing the complex topology of the internet. From the OPTE Project [1].

Complex systems that can be modelled as elements interacting in pairs are often modelled with a mathematical structure called *graph*. The corresponding system

is often called a *network*[1]. Network science is the branch of applied mathematics that studies these systems. It is useful to think of network science as *applied graph theory* or *pairwise complex systems*, depending on our focus. If we seek to extend the mathematical core of graph theory to new applications, it aligns with applied graph theory. Conversely, if we are interested in specialising the wide range of models and methods from the many disciplines within complex system theory to a specific class of problems, it can be thought as the study of pairwise complex systems.

One of the most significant applications of network science – and also one of the first chronologically – is in the social sciences, where the main interest is the analytical study of social networks. In general, social networks have people acting as the nodes, connected by edges depending on the presence or absence on some sort of social interaction, which is application-dependent. The most notable examples include communication networks, friendship networks, and collaboration networks, just to name a few. The network analysis in such cases can provide insightful information on the social structure that would be impossible to identify with more qualitative analysis, such as the emergence of community structure and the formation of social hierarchies.

Network science has also been successfully applied in engineering, for example with transportation networks, power grids, and communication networks. Transportation networks, such as highways and transit systems, can see optimised travel times and reduced congestion as a result of a network analysis optimisation. Similarly, power grids undergo constant load balancing that ensure that energy is distributed efficiently and minimise the risks of a cascading failure. Cascading is a concept derived from network science, that refers to a property being propagated through all nodes of network in a very short time and would be particularly catastrophic in a power gird failure, potentially resulting in a widespread blackout and cutting off millions of

[1]Despite *network* usually referring to the real-world systems and *graph* referring to the mathematical object that is used to model it, often the two terms are used interchangeably

people from electrical power. Communication networks can be studied with tools from network science with the objective to improve data transfer rates, reduce latency, and improve network security. Thus, understanding the dynamics of complex networks can help optimise these systems and improve their efficiency.

Biology is another possible area of application. Indeed, systems like gene regulatory networks, protein-protein interaction networks, brain networks, and food webs have been studied extensively with methods and techniques derived from network science. Some results in network science directly come from the study of biological network, such as the idea of trophic coherence, which was developed from the concept of trophic levels in food chains.

The present work will be mostly application-agnostic and will focus on the underlying mathematical structures and properties, without too much regard on the origin of the network under analysis. This is, however, a strength rather than a weakness. First, understanding the interplays between the dynamical and structural properties of a network provides valuable insights into the very nature of complex systems. Additionally, because we do not limit ourselves *a priori* to one field or area of application, the results can be extremely portable from one application to the other. The only requirement is the modelling of an applied problem yielding a network that shares the same mathematical properties that are assumed as hypothesis of our analysis. While such hypothesis is in general quite broad, we will emphasise throughout this thesis which applications are more likely to benefit from the results we prove.

1.1 Graph structure and dynamics

Networks exhibit a plethora of different structures and behaviours. The two are closely related: structural features such as hubs, bridges, and communities can impact the overall dynamics on the graph. Vice-versa, the dynamical properties of random

walks on the network contain important information that can be used to infer the structural features of the network. For example, highly connected nodes – or *hubs* – will be visited more often than nodes with fewer connections. Densely interconnected subgraphs – sometimes referred to as *communities* – can give rise to local effects that separate the dynamics into different time scales. Bridges, which are connections across different clusters, facilitate the diffusion across separate parts of the network and are both a crucial bottleneck and a weak point with respect to perturbations of the original graph.

This bidirectional relation between the structure and the dynamics of networks has proven incredibly fruitful. For example, the scale-free property for real world networks has helped the forecasting of the spread of computer viruses [4]. The aforementioned effect of communities on random walks by has been successfully exploited to detect clusters [5].

This thesis builds on these results and develops some novel ideas that are based on the same duality of structure and dynamics, as well as their interdependence.

1.2 Outline and contributions

This thesis is composed by three research projects, each independent from the other two, but all related to the underlying theme of this thesis: the interplay between the network structure and the random walk dynamics on the network.

First, in chapter 2 we introduce the mathematical objects that will be relevant in the subsequent chapters. As the title suggest, we will focus on both structural elements and dynamical phenomena. In particular, we pay special attention to random walks, which will be the starting point for developing our dynamical insights.

In chapter 3 we introduce a novel coarse-graining algorithm for directed network, grounded in the study of random-walks dynamics. This chapter contribution

is twofold. First and foremost, it provides a novel, efficient, dynamic-preserving compression algorithm (algorithm 3). In addition, it also introduces a generalisation of some dynamical structures on directed graphs, that we called ergodic sets. While the intuition upon which these subgraphs are defined is simple, definition 26 provides a simple, constructive definition for such structures, and proposition 5 puts them in relation with other network structures that are widely established in the literature.

Chapter 4 investigates random walks more closely. It starts from an apparent contradiction between the dependence of random walks return time distributions on global properties of the network only and the use of random walks in community detection algorithms. The main contribution of this chapter is the formulation of the conjecture that random-walks-based community detection algorithm work because of a finite size effect. It also provides a new message-passing approach for fast calculation of the first return time distribution on large, sparse graphs (equation (4.41)).

Finally, chapter 5 examines the random cluster similarity, a novel similarity measure for nodes in a network, grounded in the global network structure. We provide its definition (equation (5.1)) and its relations with some network models from statistical mechanics and the theory of percolation. We analyse the behaviour of the measure for the critical limits of its parameter (equation (5.9)), and we provide an analytical result for its value in the appropriate regimes.

Chapter 2

Preliminaries

This chapter introduces some fundamental mathematical tools and techniques that will be used throughout this thesis, as well as contextualizing their use in some important applications. The fundamental mathematical object this thesis considers are graphs upon which all subsequent elements are based and on which all subsequent techniques are applied. Matrices would be a notable exception to this claim, having developed their own mathematical literature clearly separated from graph theory and network science, but we reconcile this discrepancy by only considering those matrices that are generated by graphs, such as adjacency matrices (see section 2.1) and stochastic matrices (see section 2.2).

In general, a graph – or *network* – is a set of interconnected entities – often called *nodes*, or *vertices* – which are connected by links – called *edges* – in a pairwise fashion. The resulting structure is made up of a list of elements, the vertex set, and a list of pairs of such elements, the edge set. This encoding has proven a useful mathematical model in a large number of very different fields, from social science to biology, from computer science to logistics. Complex graphs exhibit non-trivial topological features such as complex or hierarchical community structures, high clustering coefficient, or a heavy-tailed degree distribution with the formation of hubs. Often these proper-

ties result in collective behaviours that go beyond the behaviour of their individual nodes; for this reasons they are considered an example of complex systems, and the corresponding network is referred to as a *complex network*.

2.1 Graphs

While it is assumed that the reader possesses some familiarity with the basic ideas and results from graph theory and network science, this section presents a more formal introduction of the basic concepts upon which the results presented in future chapters are built.

The aim of this section is not to be an exhaustive introduction to the fields of graph theory and network science, but rather a primer on the subject containing the most important structures and results that are going to be used throughout this thesis. For this reason, this section has been purposefully kept brief, some intermediate results have been skipped, and many adjacent topics that are not strictly necessary for the self-coherence and completeness of this thesis have been entirely omitted. This introduction will also allow to fix the naming conventions and the notation to avoid any confusion in the forthcoming pages.

We start by giving the definition of graph from the ground up, introducing the concept of the inverse graph, and naming some graph structures, such as nodes neighbourhoods and special subgraphs.

Definition 1. *A* ***graph*** $\mathcal{G}$ *is given by a couple* $\mathcal{G} = (V, E)$*, representing a finite, discrete set of* ***vertices*** $V = \{v_1, \dots, v_N\}$ *and the set of* ***edges*** $E \subseteq 2^V$*. Any edge* $e \in E$ *can be represented by the couple* $e = (v, w)$ *of its vertices, which can be assumed either* ***unordered*** *or* ***ordered****. Correspondingly, we say that the graph is* ***undirected*** *and* ***directed****, respectively.*

Definition 2. *Let $\mathcal{G} = (V, E)$ be a directed graph. The* ***inverse graph*** *of $\mathcal{G}$ is the graph $\mathcal{G}' = (V, E')$, in which an edge $e = (v, w) \in E' \Leftrightarrow (w, v) \in E$. In other words, the inverse graph of $\mathcal{G}$ is the graph that has the same nodes but in which all edges have been reversed.*

Definition 3. *Let $\mathcal{G}$ be a directed graph. Let $v \in V = V(\mathcal{G})$. We call the* ***out-neighbourhood*** *of v the set vertices:*

$$\mathcal{N}_{out}(v) = \{w \in V | (v, w) \in E(\mathcal{G})\} \tag{2.1}$$

Similarly, we call the ***in-neighbourhood*** *of v the set vertices:*

$$\mathcal{N}_{in}(v) = \{w \in V | (w, v) \in E(\mathcal{G})\} \tag{2.2}$$

Let now $\mathcal{G}$ be an undirected graph. For each vertex $v \in V(\mathcal{G})$, out-neighbourhood and in-neighbourhood coincide, we call both set ***neighbourhood*** *and we write*

$$\mathcal{N}(v) = \mathcal{N}_{in}(v) = \mathcal{N}_{out}(v)$$

Definition 4. *Let $\mathcal{G}$ be a directed graph and let $v \in V(\mathcal{G})$. We call the vertex* ***in-degree*** *the cardinality of the in-neighbourhood, and the* ***out-degree*** *the cardinality of the out-neighbourhood. If $\mathcal{G}$ is undirected, the two quantities coincide and we call both the* ***degree****.*

$$d^{in}(v) = |\mathcal{N}_{in}(v)| \qquad d_{out}(v) = |\mathcal{N}_{out}(v)| \qquad d(v) = |\mathcal{N}(v)| \tag{2.3}$$

Definition 5. *Let $\mathcal{G}$ be a directed graph. We say $v \in V(\mathcal{G})$ is a* ***source*** *if $d_{in}(v) = 0$.$v \in V(\mathcal{G})$ is a* ***well*** *if $d_{out}(v) = 0$.*

Now we turn to define the global property of connectedness for graphs, which is

intuitively described as *the graph can be drawn all in one piece.*

Definition 6. *Let $\mathcal{G}$ be an undirected graph. We say that $\mathcal{G}$ is* ***connected*** *if, for each pair of vertices $v, w \in V(\mathcal{G})$, there exist a sequence of vertices $v_1, v_2, \ldots, v_n \in V(\mathcal{G})$ such that*

$$(v, v_1), (v_1, v_2), \ldots, (v_{n-1}, v_n), (v_n, w) \in E(\mathcal{G})$$

Definition 7. *Let $\mathcal{G}$ be a directed graph. We say that $\mathcal{G}$ is* ***connected*** *if the underlying undirected graph is connected. We say that $\mathcal{G}$ is* ***strongly connected*** *for each pair of vertices $v, w \in V(\mathcal{G})$, there exist a sequence of vertices $v_1, v_2, \ldots, v_n \in V(\mathcal{G})$ such that $(v, v_1), (v_1, v_2), \ldots, (v_{n-1}, v_n), (v_n, w) \in E(\mathcal{G})$.*

Intuitively, connectedness corresponds to having a sequence of consecutive undirected edges (even if the graph is directed) between all possible pairs of nodes.

Remark 1. *The two definitions above can be combined to reach this standard result in graph theory. In an undirected network, the concepts of connectedness and strong connectedness are equivalent. In directed network, this is not true, and strong connectedness is stronger requirement.*

Having introduced the fundamental properties of graphs that will be used in this thesis, the following definitions introduce some further properties of graphs that, while a direct consequence of the definitions above, provide some useful language for an effective treatment of more complex results.

Definition 8. *Let $\mathcal{G}$ be a directed graph. We call the* ***in- (resp. out-) degree sequence*** *the list of the in- (out-) degrees of all nodes $v \in V(\mathcal{G})$. If $\mathcal{G}$ is undirected instead, we call the* ***degree sequence*** *the list of the degrees of all nodes $v \in V(\mathcal{G})$.*

Definition 9. *Let $\mathcal{G}(V, E)$ be a graph. We can represent $\mathcal{G}$ through a matrix $A \in \mathbb{R}^{|V|\times|V|}$, called the* ***adjacency matrix****, in which the entries are assigned as follows:*

$$A_{vw} = \begin{cases} 1 & if (v, w) \in E(\mathcal{G}) \\ 0 & otherwise \end{cases} \tag{2.4}$$

Note that A is symmetric for undirected graphs, while this is not true in general for $\mathcal{G}$ directed.

2.1.1 Models of random graphs

Random graphs were first introduced during the fifties and sixties by Hungarian mathematicians Paul Erdős and Alfred Rényi, who were the first to create a network from the conscious application of a set of probabilistic rules, leading to the model that today bears their names [6]. Their idea was simple: given a set of nodes, each edge is present in the graph with a certain probability controlled by a set of i.i.d. stochastic Bernoulli variables. Yet, it proved incredibly fruitful.

Half a century later, more complex models of random graphs have been introduced,

Models of random graphs are pivotal in the study of complex networks due to their ability to provide a theoretical framework for analysing the intricate structures and behaviours of various real-world systems. These models offer essential insights into the underlying principles that govern network connectivity, robustness, and the emergence of large-scale patterns. By representing networks with a high degree of abstraction, random graph models allow researchers to predict emergent properties, such as the formation of giant connected components and the spread of information or diseases. This predictive power is crucial for understanding phenomena in fields ranging from epidemiology and sociology to computer science and biology.

Moreover, random graph models serve as fundamental tools for developing and testing algorithms designed to operate on networks. They provide benchmark struc-

tures against which the performance, efficiency, and scalability of network algorithms can be measured. In addition, these models facilitate the comparative analysis of real-world networks, helping to identify significant deviations or similarities that can lead to new hypotheses and deeper insights. The broad applicability and versatility of random graph models underscore their importance in advancing both theoretical research and practical applications across diverse scientific and engineering disciplines.

Models of random graphs are surged to prominence in network science and are now a cornerstone of the research conducted in the field. While the test against the real world is the ultimate gold standard for any scientific result, networks drawn from applications can be expensive to source and encode, and it might not be possible to test against all possible cases. It is much more convenient to employ models of random graph: sets of rules that can generate a large number of graphs all different from each other but with a common set of properties all of them share. We normally say that graphs which are generated by the same model using the same parameters *belong to the same ensemble.*

This approach provides a lot of advantages:

- provides a controlled environment for the development of analytic results: as the generation procedure is known, it is possible to draw conclusions that are valid in expectation value over the ensemble
- can construct a sample set with the statistically significant size to prove analytic results through simulations
- allows testing on limit cases that are often not available directly from real networks, or are very difficult to source

A few models have proven especially fruitful in the study of networks. Here we are going to discuss four of the most popular models: the Erdős-Rényi (ER) model,

the stochastic block model (SBM), the Barabasi-Albert (BA) model, and the Watts-Strogatz (WS) model.

The Erdős-Rényi model is one the earliest and simplest models of complex networks. In the literature, this has become synonymous with *random graph*, and two slight variations of this model are often considered to fall under the same name. In the first model, a fixed number N of nodes are connected pairwise with a fixed probability p. The presence of each edge is independent from the presence of all other edges, and thus the resulting degree distribution follows a binomial distribution. This version of the Erdős-Rényi model is often indicated as $ER(N, p)$, from the initials of its proposers and the variables of the model. The second variation, indicated as $ER(N, M)$, also has N vertices but the number of edges M is also fixed. The edges then assigned randomly by selecting which M of the total $N(N-1)/2$ possible edges are realised in the graph. This model effectively performs a change of variable in the stochastic formulation of the model. In fact, the single edge probability p and the total number of edges are related, in expectation, as:

$$\langle E \rangle = \frac{N(N-1)}{2} p \tag{2.5}$$

The ER model is very simple and easy to treat analytically because of the probabilistic independence of all the edges and the fact that only has two parameters N and p (or M). However, while the ER model is useful for studying some properties of networks, it fails to capture a number of other properties often observed in real-world networks, such as the presence of communities, the emergence of hub nodes, and the small world property. The following three models will address each of these three shortcomings.

The stochastic block model (SBM) is a popular alternative to the ER model whenever we want to encode a planted community structure. In an SBM, nodes are

partitioned into groups (or *communities*), and the probability of a connection between any two nodes depends on which community they belong to. Formally, the probability that an edge exists between two nodes in the same community, often indicated as p_{in} is different to the connection probability for nodes belonging to different communities p_{out}. Because communities are normally more tightly connected than the rest of the network, a phenomenon called *homophily* or *assortativity*, we normally choose $p_{in} > p_{out}$. If the converse is true, we say the graph shows *eterophily* or *disassortativity*, but such a case is quite rare in practice. We indicate the ensemble of graphs generated from an SBM model with $SBM(N, p_{in}, p_{out})$. The SBM can capture the community structure often observed in real-world networks and has thus been used to study a wide range of phenomena in which such structure plays an important role, including epidemic spreading and social influence.

The Barabasi-Albert (BA) model is a mechanism for generating scale-free networks, which are characterised by an exponential distribution in the degree distribution. In this model, we start with a fully connected core and we add new nodes one at a time, each new node connecting to one of the existing nodes (either the ones in the core or the ones added before it) with a probability proportional to the degree of the receiving node. This trick forces a power-law degree distribution in expectation on the ensemble by making nodes that are already highly connected more likely to gain an additional connection whenever a new node is introduced. We often call such highly connected nodes *hub nodes*. The BA model has been used to model protein-protein interaction networks, semantic network, and airline networks.

The Watts-Strogatz (WS) model aims to mimic small-world networks, which have both high clustering coefficient and short diameter. In this model, nodes are initially arranged in a regular ring lattice, and then each edge is rewired to a new random node with a fixed probability p. This model forces the small world property in the network at the expense of an unrealistic degree distribution.

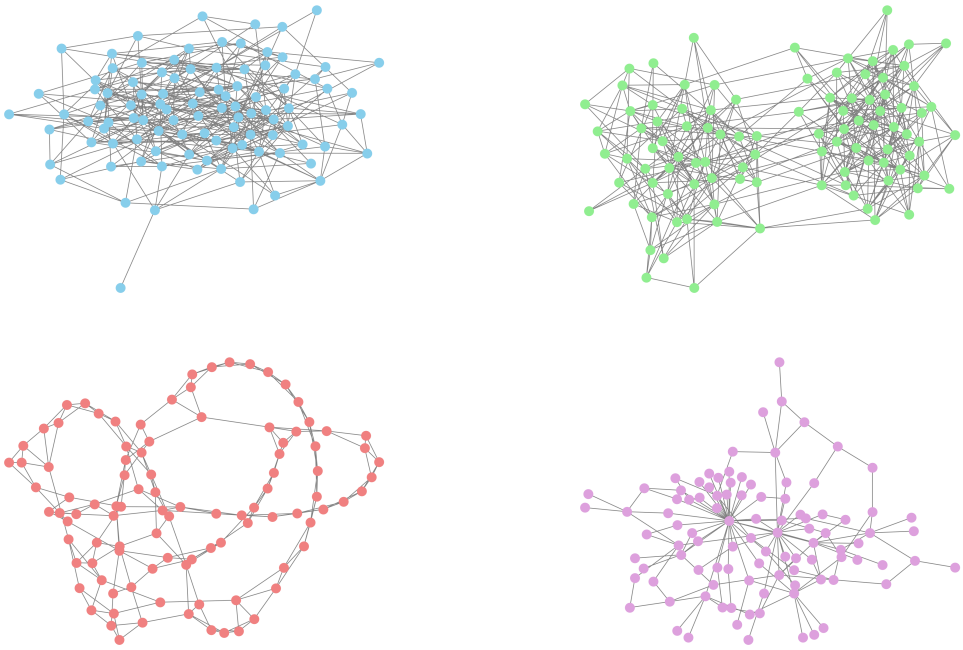


Figure 2.1: Examples of instances of the four random graph models discussed. From the top left, in clockwise direction: Erdős-Rényi random graph $ER(100, 0.07)$, SBM model $SBM(100, 0.15, 0.01)$, a Watts-Strogatz model $WS(4)$, and a Barabasi-Albert Model $BA(100, 1.5)$.

2.2 Random walks on graphs

Random walks have long been a fundamental stochastic process, formalised at the beginning of the twentieth century [7] and today are used in a variety of problems across many applications, from modelling stock market fluctuations [8] to predicting financial risks [9], from understanding diffusion processes in physics [10], to the study of patterns in population genetics [11]. They are also central in computer science [12], particularly in search and optimization algorithms [13]. In network science, random walks on graphs are a powerful tool for analysing the structure and properties of complex networks.

We start with an informal introduction: a brief description of what a random walk on a graph is in practice. In simple terms, it is a stochastic process that starts at a given node and explores the network crossing a random edge at each of a series of

discrete-time steps. Because this process moves from node to node, it is often referred as *node-centric* random walk; conversely, an *edge centric* random walks would occupy an edge at each step and would evolve moving from that edge to the next. At each step, the process moves to a neighbouring node with some probability that - in the general case - is proportional to the weight of the edges. We will consider unweighted network, so our random walkers will choose the edge to cross uniformly at random at each step. This process then either repeats continuously or stops. This latter case often happens after a certain number of steps have been taken or until a stopping condition is met. A random walk results in a node sequence given by all the nodes it visits, in the order they are visited. This, in turn, provides information about the structure of the underlying graph, and can be used to estimate various properties of the graph, including node centrality, the clustering coefficient, and the community structure. The latter in particular will be the topic of chapter 4, where the interplay between the community structure, random walks, and the SBM phase transition is analysed.

Now we move to the more formal introduction. There are many strong parallels between the structural properties in the theory of ergodic sets which we will introduce in chapter 3 and the theory of random walks on networks [14]. The latter is also well-established in the literature [15], and therefore establishing connections can be helpful in leveraging the theory that has already been developed.

In addition to this, some results are more naturally expressed in terms of random walks rather than network connectivity structure, so it is always helpful to have an alternative representation of the same phenomenon available in our toolkit.

Among the many available sources, for the presentation we will mainly follow the book from Norris [16], from which we will take some basic definitions and established results before proceeding with further original results.

Definition 10. *A **random walk** on a [directed] graph $\mathcal{G} = (V, E)$ is a Markov Chain*

process $X(t)_{t\geq 0}$ such that the transition probability depends only on the [out-]degree of the current state:

$$\mathbb{P}\left(X_{t+1} = w | X_t = v\right) = \frac{1}{d_{out}(v)} \mathbb{1}_{\mathcal{N}(v)}(w) \tag{2.6}$$

Hence, to define a graph random walk we need to introduce Markov chains. For this, we need two main ingredients, namely:

- a probability distribution λ on $V(\mathcal{G})$
- a transition matrix P, which is a *stochastic matrix*

For the vector λ to be a probability distribution we only need it to be normalized ad its components to be non-negative:

$$\lambda_i \geq 0; \qquad \sum_{i \in V} \lambda_i = 1$$

The probability matrix P need to be a *stochastic matrix*, namely its elements must be non-negative and smaller than one, and its rows must be normalised:

$$0 \leq P_{ij} \leq 1; \qquad \sum_{j \in V} P_{ij} = 1 \, \forall i \in V$$

Definition 11. *We say that an $N \times N$ matrix P is **stochastic** if every row represents a probability distribution. This means all the entries P_{ij} are such that $0 \leq P_{ij} \leq 1$ and each row sums to one: $\sum_j P_{ij} = 1$. Each stochastic matrix generates a Markov chain via the following relation:*

$$P_{ij} = \mathbb{P}\left(X_{t+1} = j | X_t = i\right) \qquad \forall i, j \in \{1, 2, 3, \ldots, N\} \tag{2.7}$$

We now further connect the graph structure with the random walk using the graph adjacency matrix.

Remark 2. *Let $\mathcal{G}$ be a graph, and A its adjacency matrix. Then the matrix P_A, defined as the row-normalised adjacency matrix, is a stochastic matrix and the associated Markov chain is the random walk on $\mathcal{G}$. The element-wise definition of P_A is:*

$$(P_A)_{ij} = \frac{A_{ij}}{\sum_j A_{ij}} \tag{2.8}$$

While definition 10 is the most direct definition for a random walk on a graph, it, it somewhat loses sight of the empirical properties of the random walk. Random walks are a subset of a more general family of stochastic process, which all share the same fundamental property.

Definition 12. *We say a discrete-time random process on a discrete state space $\mathcal{X}$, which we indicate as $(X_t)_{t\geq 0}$, is a **Markov chain** if and only if, for any $\tau > 0$ the following condition (also referred to as **Markov's property**) holds.*

$$\mathbb{P}\left(X_{\tau+t}|X_\tau\right) \perp\!\!\!\perp X_0, X_1, \ldots, X_{\tau-1} \qquad \forall t \geq 0 \tag{2.9}$$

*where the symbol $\perp\!\!\!\perp$ indicates stochastic independence. The condition above is also sometimes called **memory loss** property and summarized with the phrase "the future is only dependent on the current state".*

Now let's continue with some properties of the distribution λ that will be useful in the forthcoming.

Definition 13. *We say a probability distribution λ is **invariant** under a certain stochastic matrix P (called transition matrix) if:*

$$\lambda P = \lambda \tag{2.10}$$

Now we further connect the properties of the random walks with the structure of the graph. In order to do so, we need a few concept that we will build upon soon.

Definition 14. *Let $\mathcal{G}$ be a [directed] graph. Given two vertices $v, w \in V$, we say that v **leads to** w, and we write $v \to w$, if there exist a sequence of vertices $\{v_1, \dots, v_{n-1}\}$ and a sequence of [directed] edges $\{e_1, \dots, e_n\}$ such that:*

$$v \xrightarrow{e_1} x_1 \xrightarrow{e_2} x_2 \xrightarrow{e_3} \dots \xrightarrow{e_{n-1}} x_{n-1} \xrightarrow{e_n} w \tag{2.11}$$

*Furthermore, we say that v and w **communicate**, and we write $v \leftrightarrow w$, if we have both $v \to w$ and $w \to v$.*

Remark 3. *The relation "to communicate" $\leftrightarrow$ is an equivalence relation.*

Proof. We need to show that $\leftrightarrow$ is reflexive, symmetric, and transitive. But $\forall v \in V$ $v \to v$ with a zero-length path (or in zero step), hence $v \leftrightarrow v$. For symmetry, if $v \leftrightarrow w$, we have that $v \to w$ and $w \to v$, which imply $w \leftrightarrow v$. Finally, for transitivity, we note that $v \to w$ and $w \to z$ implies $v \to z$, by chaining the two paths coming from definition 14. The same is true in the other direction, thus $v \leftrightarrow w, w \leftrightarrow z \Rightarrow v \leftrightarrow z$, which prove transitivity and the remark. □

Definition 15. *We call $C \subseteq X$ a **communicating class** if it's one of the equivalence classes of the "to communicate". Furthermore, we say a communicating class C is **closed** if the following condition is satisfied:*

$$v \in C,\, v \to w \Longrightarrow w \in C \tag{2.12}$$

We are now ready to relate the communicating classes construction with the properties of the random walks. This will allow us to get some important insights on the behaviours of the random walks on directed networks, particularly on their asymptotic behaviour.

Proposition 1. *Let $X(t)_{t\geq 0}$ be a Markov chain $MC(\lambda, P)$ on the graph $\mathcal{G}$. Let $\mathcal{V}_{CC} = \{C_1, C_2, \dots, C_n\}$ be the set of the closed communicating classes of $\mathcal{G}$. Then,*

the following holds:

$$\lim_{t\to+\infty}\left(\lambda P^t\right) = \sum_{i=1}^{n} a_i \mathbb{1}_{C_i} \tag{2.13}$$

for some normalised coefficients a_i*, i.e.* $\sum_{i=1}^{n} a_i = 1$

Proof. As this is a standard proof, also seen in Norris [16], we refer to the bibliography. □

This proposition sums up the intuitive problem of random walks on directed, not strongly connected graphs: in the long-time limit, the majority of the random walkers are drawn out of the core of the network and dragged into the closed connected classes, or the outputs of the networks. This is a formal description of what Page and Brin [17] had to solve in their algorithm for the estimation of nodes centrality, and the starting point for the investigation of control theory on networks. In short, the core of the graph gets depleted as more and more weight of the probability distribution gets absorbed by the outputs.

Sometimes it is convenient to divide a long random walk into smaller chunks. One way of doing so that will become useful in the following is by using *excursions*, sections of a random walk that happens in between two consecutive visits to a given node. More formally:

Definition 16. *We call an* ***excursion (of length n from node v)*** *a random walk* $(X_t)_{0\leq t\leq n}$ *such that:*

$$X_0 = X_n = v \quad ; \qquad T_i \neq v \quad \forall\, 1 \leq i \leq n-1 \tag{2.14}$$

Note that excursions start and finish at the same node, but they are a more general concept than cycles. In the graph theory language, they are sometimes referred to as *circuits*.

2.2.1 Return time distributions

One interesting property of a random walk is the return times distributions, the number of steps that the random walker takes – on average – to go back to its starting point. Closely related to the return time is the first return time, the number of steps that the random walker takes to return to its starting point for the first time. In this section, we provide a more formal definition and we show some useful property to work with these quantities.

Definition 17. *Consider a random walk $X(t)_{t\geq 0}$ on the vertex set of a finite, connected, undirected graph $\mathcal{G}$. The* ***return time*** *$T_i(t)$ denotes the probability that a walk that started at node v_i is back at node v_i after t steps. more formally:*

$$T_i(t) = P(X_{t_0+n} = v_i | X_{t_0} = v_i) \tag{2.15}$$

Definition 18. *Consider a random walk $X(t)_{t\geq 0}$ on the vertex set of a finite, connected, undirected graph $\mathcal{G}$. The* ***first return time*** *$Y_i(t)$, defined for each node of the graph $v_i \in V(\mathcal{G})$ as the probability that a random walker starting in v_i at time t_0 and returns to the same vertex for the first time after exactly a t time-steps. More formally:*

$$Y_i(t) := P(X_{t_0+n} = v_i | X_{t_0} = v_i, X_{t_0+1} \neq v_i, X_{t_0+2} \neq v_i, \ldots, X_{t_0+n-1} \neq v_i) \tag{2.16}$$

Remark 4. *Since we will always consider finite graphs, all random walks are recurrent (i.e. $Y_i < +\infty$ for all i), and therefore $Y_i(t)$, as a function of t, is a probability distribution, namely:*

$$\sum_{t=1}^{+\infty} Y_i(t) = 1 \quad \forall v_i \in V(\mathcal{G}) \tag{2.17}$$

The two quantities look quite similar but have different properties. Chiefly, $T_i(t)$ simply asks whether the walk has returned to v_i after a time t, whereas $Y_i(t)$ require

that the visit is the first one except for the starting position. This implies that there is only one value of t for which the first return time is realised at each instance of the random walk, and this implies that $Y_i(t)$ is asymptotically decreasing to 0 in the limit of large times, while $T_i(t)$ is such that, for all choices of t^* arbitrarily large, there exist an infinite number of $t > t^*$ such that $T_i(t) > 0$ strictly.

This concludes the expository part drawn from [16].

2.2.2 Generating functions

The connection between the return time distribution $T_i(t)$ and the first return time distribution $Y_i(t)$ is most clear if we consider the *generating functions* of $T_i(t)$ and $Y_i(t)$.

Definition 19. *Let $p(t)$ be a probability distribution. The **generating function** is the quantity defined as:*

$$F(z) = \sum_{t=1}^{\infty} p(t)\, z^t \tag{2.18}$$

So, in the case of the return time distribution $T_i(t)$ and the first return time distribution $Y_i(t)$, we have:

$$R_i(z) = \sum_{t=1}^{\infty} T_i(t)\, z^t \qquad F_i(z) = \sum_{t=1}^{\infty} T_i(t)\, z^t \tag{2.19}$$

Proposition 2. *The generating function $F(z)$ of the distribution $p(t)$ satisfies the following relations:*

$$\langle p \rangle = \left. \frac{dF}{dz} \right|_{z=1} \tag{2.20}$$

Which is a standard result from probability theory [18].

We can now use the generating function to manipulate the return time probabili-

ties when "unpacking" the random walk, i.e. splitting the sequence of visited nodes into smaller sequences followed one after the other. We can then apply this reasoning to see a return time event as the succession of a first return event plus subsequent return times. This is turn allows us to create a simple relation among the generating functions of the two events.

For example, a walk that returns to node i for the second time at time t is equivalent to two successive excursions that return at t_1 and t_2, with $t_1 + t_2 = t$. The probability for any walk to arrive back at i for the second time at t can thus be written

$$\begin{aligned} T_i(t,2) &= \sum_{t_1+t_2=t} Y_i(t_1)\, Y_i(t_2) \\ &= \sum_{t_1=1}^{\infty}\sum_{t_2=1}^{\infty} \delta(t, t_1+t_2)\, Y_i(t_1)\, Y_i(t_2) \end{aligned} \tag{2.21}$$

where $\delta(x,y)$ is the Kronecker delta. In general, the probability for a walk to arrive back at i for the m-th time is

$$T_i(t,m) = \sum_{t_1=1}^{\infty} \cdots \sum_{t_m=1}^{\infty} \delta(t, \textstyle\sum_{u=1}^{m} t_u) \displaystyle\prod_{u=1}^{m} Y_i(t_u). \tag{2.22}$$

Since any walk that arrives back at i must have already been back to i $m \geq 1$ times, we can write $T_i(t) = \sum_{m=1}^{\infty} T_i(t,m)$. Thus, to relate Y_i to T_i, for $t \geq 1$, we write

$$T_i(t) = \sum_{m=1}^{\infty} \left[\sum_{t_1=1}^{\infty} \cdots \sum_{t_m=1}^{\infty} \delta(t, \textstyle\sum_{u=1}^{m} t_u) \displaystyle\prod_{u=1}^{m} Y_i(t_u) \right], \tag{2.23}$$

and $T_i(0) = 1$.

The formula for $T_i(t)$ given in Eq. (2.23) is formally correct, but it is not easy to work with. Rather than consider $T_i(t)$ directly we will work with its generating

functions, which are closely related as follows:

$$
\begin{aligned}
R_i(z) &= \sum_{t=0}^{\infty} T_i(t) z^t \\
&= 1 + \sum_{t=1}^{\infty} \sum_{m=1}^{\infty} \left[\sum_{t_1=1}^{\infty} \cdots \sum_{t_m=1}^{\infty} \delta(t, \textstyle\sum_{u=1}^{m} t_u) \displaystyle\prod_{u=1}^{m} Y_i(t_u) \right] z^t \\
&= 1 + \sum_{m=1}^{\infty} \left[\sum_{t_1=1}^{\infty} \cdots \sum_{t_m=1}^{\infty} \prod_{u=1}^{m} Y_i(t_u) z^{t_u} \right] \\
&= 1 + \sum_{m=1}^{\infty} \left[\prod_{u=1}^{m} \sum_{t=1}^{\infty} Y_i(t) z^t \right] \\
&= \sum_{m=0}^{\infty} F_i(z)^m \qquad (2.24) \\
&= \frac{1}{1 - F_i(z)} \qquad (2.25)
\end{aligned}
$$

which shows that we only need to calculate one of $R_i(z)$ or $F_i(z)$ in order to know both.

This completes the introduction to random walks and return time distributions. We will use this framework mostly in chapter 4, where we will build upon the quantities introduced here in the case of large finite graphs to draw some conclusions about the relations of the community structure and the random walks on a network.

2.3 Graph matrices

In the network science literature, there is a well established duality between graphs and matrices [19], as each graph can be represented by its *adjacency matrix*, see definition 9. Extensions are possible for weighed graphs and multi-graphs, whereas directed graphs are already comprised in the basic adjacency matrix formalism [20].

Definition 20. *Let $\mathcal{G}(V, E)$ be a graph with n vertices. Its* ***adjacency matrix*** *$A \in \mathbb{R}^{n \times n}$ is a matrix where the element A_{ij} is defined as:*

$$A_{ij} = \begin{cases} 1, & \textit{if there is an edge from vertex } i \textit{ to vertex } j \\ 0, & \textit{otherwise.} \end{cases} \tag{2.26}$$

If the graph is undirected, the adjacency matrix is symmetric, meaning $A_{ij} = A_{ji}$.

Many other graph matrices are used, chiefly the graph Laplacian, see definition 21. It's the fundamental matrix for spectral graph theory, and its eigenvalues and eigenvector can be used for filtering in graph signal processing [21], community detection [22], the study of diffusion on graph topology [23, 24], and more [25].

Definition 21. *Let* $\mathcal{G}(V, E)$ *be a graph with* n *vertices. The* ***graph Laplacian*** $L \in \mathbb{R}^{n \times n}$ *is defined as:*

$$L = D - A \tag{2.27}$$

where D *is the degree matrix, a diagonal matrix with elements* D_{ii} *representing the degree of vertex* i*, and* A *is the adjacency matrix of the graph.*

Next, we define the submatrix of a matrix, which is a simple but convenient formalism to manipulate the graph adjacency matrix when we are only interested in a subgraph, for example a single community, a strongly connected component, or any general subset of nodes.

Definition 22. *Let* $M \in \mathbb{R}^{n \times m}$ *be a matrix, and let the two sets* $I = \{n_1, n_2, \ldots, n_i\}$ *and* $J = \{m_1, m_2, \ldots, m_i\}$ *be two subset of its indices:* $n_i \leq n$ *and* $m_j \leq m$*. We indicate with* $M_{I,J}$ *the* ***submatrix*** *obtained by selecting all the elements in the form* $M_{a,b}$*, with* $a \in I$*,* $b \in M$*.*

2.3.1 Singular value decomposition (SVD)

A standard result from linear algebra concerns *matrix diagonalisation*, or *eigenvalue decomposition*, a process by which a square matrix $M \in \mathbb{R}^{n\times n}$ can be written[1] as the product of two unitary matrices and a diagonal matrix Δ, as:

$$M = U\Delta U^T \tag{2.28}$$

This process corresponds to decompose the effect of the original matrix M into its principal subspace, on which M acts like a pure component multiplication. The associated diagonal values of Δ, often indicated as $\lambda_1, \lambda_2, \ldots, \lambda_n$, and called *eigenvalues* and represent the dilation coefficient on each subspace, which are the span of the associated *eigenvectors* $\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_n$.

Diagonalization only applies to square matrices, but its generalisation, called *singular value decomposition*, or SVD for short, can be extended to matrices of any shape.

Definition 23. *Let $A \in \mathbb{R}^{m\times n}$ be a real matrix. The **Singular Value Decomposition (SVD)** of A is a factorization of the form:*

$$A = U\Sigma V^T \tag{2.29}$$

where:

- *$U \in \mathbb{R}^{m\times m}$ is an orthogonal matrix, whose columns are the left singular vectors of A.*
- *$\Sigma \in \mathbb{R}^{m\times n}$ is a rectangular diagonal matrix (i.e. such that $\Sigma_{ij} = 0$ if $i \neq j$) with nonnegative real numbers on the diagonal. The diagonal elements are known as the **singular values** of A.*

[1] under certain broad assumptions that falls beyond the scope of this introduction, see [26]

- $V \in \mathbb{R}^{n \times n}$ *is an orthogonal matrix, whose columns are the right singular vectors of* A.

The singular values in Σ *are ordered such that* $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r \geq 0$, *where* $r = rank(A)$.

This is a standard result in linear algebra; see [26] for more details.

2.4 Community detection

Community detection in network science refers to the task of finding groups of vertices that are more densely connected to each other than to all the other nodes in the graph. More formally, a community structure is a partition of the graph:

Definition 24. *Let* $G = (V, E)$ *be a graph. A* ***partition*** *of the graph* G *is a division of the vertex set* V *into* k *disjoint subsets* $\{V_1, V_2, \ldots, V_k\}$, *such that:*

$$V = \bigcup_{i=1}^{k} V_i \quad \textit{and} \quad V_i \cap V_j = \emptyset \quad \textit{for all} \quad i \neq j \tag{2.30}$$

The partition respects the condition that every vertex $v \in V$ *belongs to exactly one community* V_i.

We call each subset V_i a **community** if the nodes are more densely connected withing each partition set than they are across different partition sets V_i, V_j. The community detection problem is often framed as the problem of finding the partition sets $V_1, V_2, \ldots, V_k$ that maximise the infra-partition edges and minimize the inter-partition edges. There are many approaches to this problem, all roughly but not exactly equivalent [27] and indeed there is no unanimous consensus for a unified definition of what a community even is. We will steer clear of such problem, and we take a more inferential approach: we suppose that there is a *ground truth* community structure built into the graph. Most often we are the ones who planted the structure

in the first place using a SBM algorithm, but even when dealing with real-world networks we try to avoid the most problematic cases and indeed for the purpose of the results in this report, the existence of a community structure is always assumed when required.

For practical purposes, we will utilise a modularity-based approach to detect the planted community structure. The Newman-Girvan modularity [22] is a scalar quantity Q which depends on the community assignment of all nodes of the graph. For a given community assignment $\{c_v\}_{v\in V}$ we compute the modularity as:

$$Q = \frac{1}{2M}\sum_{v,w}\left[A_{vw} - \frac{d_v\, d_w}{2M}\right]\delta(c_v, c_w) \tag{2.31}$$

and represents the proportions of the edges that connect nodes that are in the same assigned community minus the expected value we would get if all the edges were distributed at random.

One modularity-based community detection method worth mentioning is the Louvain method [28], which has become one of the *de facto* standards when performing community detection. It provides a hierarchical structure of community of different sizes by first finding small communities from local modularity optimization. It then uses the coarse-grained graph to merge such small communities into larger ones, until such merging results in a modularity increase. When further operations cannot increase the modularity any further, the method stops and the final community structure is returned.

2.4.1 The SBM phase transition

There in an interesting point at the confluence of the topics discussed in the two previous sections. The SBM model can be seen as a generalization of the ER model, and indeed it reduces to an ER model whenever the edge probability is the same

within communities and across communities, $p_{in} = p_{out} = p$. The two ensembles $ER(N, p)$ and $SBM(N, p, p)$ are then completely indistinguishable from each other. Another way of giving the condition is by enforcing $\Delta p = p_{in} - p_{out} = 0$. This is useful because it gives us a single parameter Δp to control the strength of the SBM community structure. In fact, when $\Delta p = 1$, this forces $p_{in} = 1$ and $p_{out} = 0$ (as they are both probabilities), and the resulting SBM ensemble will consist of networks in which each community is a fully connected graph and there are no edges between any pairs of communities.

A natural question therefore arises: at which value Δp the community structure becomes relevant? This is an easy question if we know the *a priori* planted community. As soon as $\Delta p > 0$, looking at enough graphs we can detect the non-zero difference between p_{in} and p_{out} with a high statistical significance, as the model ensemble contains $2^{N(N-1)/2}$ different graphs.

Conversely, if we don't know have prior knowledge of the planted community structure, the same questions turn out to be very difficult. Suppose we are given a graph $\mathcal{G}$ and we need to decide whether it has been generated by an ER or an SBM model. We first need to find the communities, but for small values of Δp the community detection will fail. As it turns out, this problem is called detection and it's the easiest of three connected problems. It has been shown that the detection problem has a phase transition in the constant-degree regime. If the community structure is too weak or the graph too sparse, no algorithm can predict better than chance the model that has generated $\mathcal{G}$.

An increased difficulty is the reconstruction problem (sometimes referred to as weak reconstruction), which requires labelling the vertices with a community assignment c_{est} that is correlated with the ground truth assignment c_{true}. On the other hand, exact reconstruction consists of finding the planted assignment exactly, labelling every vertex correctly up to a permutation of the groups. The exact threshold at which

these problems become impossible is beyond the scope of this work, but recent works have provided a calculation for such values, see for example [29, 30].

We will elaborate more on this in chapter 4, when discussing the effects the community structure has on random walks on the graphs.

2.4.2 InfoMap

Many community detection algorithms have been proposed and used successfully, often built on different intuitions or heuristics to identify network communities. For example, modularity maximization methods [22] (including the Louvain algorithm [28]) determine the optimal clustering by optimizing the Newman-Girvan modularity function:

$$Q = \frac{1}{2|E|} \sum_{i,j} \left[A_i j - \frac{d_i \, d_j}{2|E|} \right] \delta(C_i, C_j) \tag{2.32}$$

in which A is the graph adjacency matrix and $\delta(C_i, C_j)$ is an indicator function which is defined as:

$$\delta(C_i, C_j) = \begin{cases} 0 & \text{if } i \text{ and } j \text{ belong to the same community, } C_i = C_j \\ 1 & \text{otherwise} \end{cases} \tag{2.33}$$

Other successful approaches to community detection include label propagation [31], spectral clustering [32], clique percolation [33], and more.

Here, we are especially interested in methods based on random walks, which are all based on the central assumption that because communities are more tightly connected internally than they are with the outside world, random walks are more likely to stay within the community and will take a longer time – on average – to venture across a sparsely connected region that separates the original community with the others.

One particularly smart application of this idea is the foundation of the InfoMap algorithm, which exploiters the duality between community detection and the problem

of minimizing the description length statistics for a prefix-free Huffman code used to identify the position of the random walker in the network [34]. This method has been shown to be effective in locating communities in large networks, including those with overlapping or hierarchical community structures [35] and its popularity for community detection has increased since its introduction.

Here we include a quick primer on the method, mostly based on [34]. The InfoMap target function, called the Map equation and whose optimisation yields the final community partition, is expressed as follows:

$$L(M) = q_{\curvearrowright} H\left(\mathcal{Q}\right) + \sum_{i=1}^{m} p_{\circlearrowright}^{i} H\left(\mathcal{P}^{i}\right) \tag{2.34}$$

in which $q_{\curvearrowright}$ is the probability of exiting a community, $H\left(\mathcal{Q}\right)$ is the entropy of the exit distribution across communities, $p_{\circlearrowright}^{i}$ is the total probability of staying within the i-th community, and $H\left(\mathcal{P}^{i}\right)$ is the entropy of the movement within the i-th community.

The expression in equation (2.34) represents the expected description length of a code representing the position of a random walk, taken over all possible partitions of the network. Minimising $L(M)$ corresponds to finding the most accurate network clustering [36].

2.4.3 Normalised Mutual Information

The normalised mutual information (often shortened in the literature as NMI) is a statistical measure that has been widely used in the network science literature to evaluate the similarity of graph partitions. This is particularly useful in the theoretical study of the community detection algorithm, where we can use the normalised mutual information to compare the planted community structure C with the clustering C' resulting from the algorithm under consideration.

Definition 25. *Let $\mathcal{G}$ be a graph, C, C' two clustering on the vertex set $V(\mathcal{G})$. The*

normalised mutual information *between the two community structures* C, C' *is calculated as:*

$$NMI(C, C') = \frac{2I(C, C')}{H(C)\, H(C')} \tag{2.35}$$

in which $I(C; C')$ is the *mutual information* of the two clusters, calculated as:

$$I(C; C') = \sum_{c \in C} \sum_{c' \in C'} P(c, c') \log \left(\frac{P(c, c')}{P(c) P(c')} \right)$$

with $P(c, c')$ the joint probability of a node to be in cluster c in the clustering C and in cluster c' in clustering C', while $P(c)$ and $P(c')$ are the two marginal probabilities, each conditioned on the other event. Furthermore, $H(C)$ is the clustering Shannon entropy, calculated as:

$$H(C) = -\sum_{c \in C} P(c) \log P(c)$$

If the two clustering C, C' are identical, their NMI will be $NMI(C, C') = 1$. Conversely, $NMI(C, C') = 0$ indicates perfect uncorrelation between the two clustering. However, the normalised mutual information has shown some limitations in the literature. For example, when one clustering has a different number of communities than the other, the NMI might not accurately represent the similarity between them [37]. We encountered this problem in a slight modified form, in which the community detection algorithm returned too many clusters (up to isolated vertices) as the optimal result with respect to the distance from the planted community structure. We have taken such limitations into account by limiting the number of communities allowed in the community detection problem, for more details see section 4.4.3.

2.5 Message passing

In network science, we often associate the nodes of a graph with some variables, and many common problems can be formulated as an inference task aimed at determining the values of those variables. The inference problem is often dependent on the structure of the graph, as the variables influence each other based on the network connections.

The most famous problem in this class is the calculation of the centrality measure, solved by L. Page and S. Brin in their seminal work [38] introducing the PageRank algorithm to efficiently calculate the centrality of the node as the eigenvector corresponding to the eigenvalue of the random surfer matrix $\mathcal{S}$:

$$\mathcal{S} = (1 - \alpha)A + \alpha\mathbb{1} \tag{2.36}$$

in which A is the graph adjacency matrix, the elements of the $\mathbb{1}$ matrix are all identically 1, and α is a parameter that controls the teleportation probability for the random surfer and has been traditionally set at $\alpha = 0.15$.

This is far from the only problem that can be represented this way. This representation can be used to model, among others:

- community detection problems, in which the value to be inferred for each node is a community label, indicating to which community the node belongs,
- opinion dynamics and consensus formation in social networks, where we want to infer the final opinions of the nodes after a series of interactions,
- graph signal processing, in which we want to infer de-noised signal values for signals (e.g. sensor readings) defined on the nodes of the graph,
- protein function prediction, where we are interested in the functional characteristics for each protein in the protein interaction network, which are based for

each protein on its interaction patterns.

This problem has been studied extensively in the literature, and one approach in particular has been proven successful. Due to the breadth and diversity of applications this problem sees, it has been developed independently in different disciplines and this resulted in a family of similar methods with different names. It is often called *message passing algorithm* in network science but is also known as *belief propagation* in the artificial intelligence literature, *cavity method* in statistical physics, and *sum-product algorithm* in coding theory.

In the next chapter, we use the belief propagation algorithm to infer the first return times of random walks on a network. Here we present a short introduction to the method, mostly following the reference book by Mézard and Montanari [39].

2.5.1 BP algorithm

The basic idea behind the algorithm is to predict the values for the inference problem by starting with a guess for the feature vector of each node h_v. Then, each node compute a *message* m to send to all its neighbouring nodes:

$$m_{v\to w} = m\left(h_v, h_w, e_{uv}\right) \tag{2.37}$$

which in general depends both on the state of the sending node h_v and of the receiving node h_w, and the property (usually, the weight) of the edge connecting them e_{uv}. Messages are normally defined as probabilities, so that $m_{v\to w} \geq 0$ and $\sum_v m_{v\to w} = 1$.

Then, each node computes the *aggregation* of all the messages $m^{\mathcal{A}}$ that it receives. This is often a simple sum of all the messages, but in principle the combination of all the messages could be a more complex aggregation function $\mathcal{A}$:

$$m_{\to v} = \mathcal{A}\left(m_{u\to v},\, \forall u \in \mathcal{N}(v)\right) \tag{2.38}$$

in which $\mathcal{N}(v)$ denotes the graph neighbourhood of node v (see definition 3).

Finally, the resulting aggregate message is used to update the feature vector of each corresponding node:

$$h_v^{(t)} = \mathcal{U}\left(h_v^{(t-1)}, m_{\to v}^{(t)}\right), \tag{2.39}$$

in which $\mathcal{U}$ is the update function.

This process is then repeated until convergence or a fixed number of iterations, to obtain an approximation for the correct values. For this reason, another dependency is often added in the notation, identifying the (usually, discrete) time step at which the message is calculated, propagated, and the feature vectors are updated. When introducing this dependency, we can write the full belief-propagation equations as:

$$h_v^{(t)} = \mathcal{U}\left(h_v^{(t-1)}, \mathcal{A}\left(m_{u\to v}^{(t-1)}, \forall u \in \mathcal{N}(v)\right)\right), \tag{2.40}$$

Furthermore, the specific functional expressions of the guesses h_v, the messages $m_{u\to v}$, the aggregation function $\mathcal{A}$, and the update function h'_v are all application-dependent.

We will not spend much time digging into the theory and the proofs of the general methods, preferring to devote more ample space to the specific application we use in chapter 4. However, it's important to note the general principles upon which the method is based. The first is the concept of locality, which is the notion that each node can only directly interact with its immediate neighbours, while further nodes require indirect interaction via the neighbours. This multi-level interaction builds the expressive power of the message passing algorithm, by virtue of which the nodes can gather information from increasingly distant parts of the graph at every iteration, which allows the algorithm to capture complex dependencies and long-distance interactions.

2.5.2 Population dynamics algorithm

It is a well-known result that the belief propagation algorithm is exact on trees [39]. More recently [40], extensions of the belief propagation framework have been developed to extend its applicability to graphs with loops.

Here, we use those extension methods to describe the *population dynamics* algorithm , which is a special type of belief propagation based on a Markov chain Monte Carlo approach [39]. We will use to numerically solve equation (4.26), due to the specificity of this method and its application in this thesis, we will suspend our generalist approach for this section, instead describing the population dynamics algorithm on the quantities of relevance in chapter 4. We want to estimate the message $P_{\leftarrow}^{(z)}(F, d)$, propagating the estimation of the first return time distribution F, subject to the node degree d.

First, for the sake of argument, suppose we already had a set of N samples for F, called $Q = \left(F_i, d_i\right)_{i=1\ldots N}$. For now, we do not worry about how we obtained these samples. Assuming Q is correctly sampled from the true probability distribution $P_{\leftarrow}^{(z)}$, we could generate a new sample as follows. First, generate a degree d proportional to $d\rho_d$, then take d randomly chosen members of Q, call these (F_j, d_j), and set $F = \sum_{j=1}^{k} \frac{z^2}{dd_j(1-F_j)}$. After doing this, the pair (F, d) will also be a correct sample from $P_{\leftarrow}^{(z)}$. This is the key observation behind the population dynamic algorithm [39].

In the population dynamic algorithm, we begin with an initially incorrect set of samples—we initialize Q to be sampled from some other tractable distribution, e.g., a uniform distribution. Then, we create new (also initially incorrect) samples by using the above process and replace the samples in Q with the new samples. A full sweep of this process is as follows:

As stated previously, if Q were a correct set of samples, the above sweep would leave us with a new set of correct samples. Or, in other words, a correct set of samples is the stationary distribution of this sweep process. So, starting out from an arbitrary

Algorithm 1: Population dynamics

Input : d_i degree distribution
Output: F first return times

for $i \in 1, \ldots, N$ **do**
  $F \leftarrow 0$
  $k \sim d\rho_d / \langle d \rangle$
  for $u = 1, \ldots, k$ **do**
    $j \sim \text{Uniform}(N)$
    $F \leftarrow F + \frac{z^2}{d_i d_j (1 - F_j)}$
  $(F_i, d_i) \leftarrow (F, d)$

distribution, we simply need to run a large number of sweeps to reach the stationary distribution, and thus numerically sample the distribution of interest, $P_{\leftarrow}^{(z)}$.

2.6 Graph percolation

Another important phenomenon in graphs – and one that has been proven fruitful in terms of the many results it has produced – is percolation theory. Originally developed in statistical physics as a model for porous materials, its significance has developed so much that it has become a research question in its own right.

In its original form [41], percolation was assumed on a lattice graph, and the problem was formulated as follows: *given a cubic lattice in d dimensions with side N and a uniform probability p for each first-neighbour connection to be in the graph, what is the percolation probability $\mathcal{P}(N, d, p)$ for a path connecting two opposite sides of the lattice?*

On a graph, this original problem has developed in two distinct but closely related forms: the first one is often referred to as *site percolation*, in which each node in the graph is *occupied* with some uniform probability p_n and the path spanning the graph can only include non-occupied nodes. The other version, called *bond percolation*, works much the same way but now are the edges that can only be traversed if they

aren't occupied, which is a random property occurring with probability p_e for each one independently.

Percolation appears in many areas of Network Science: for example, the Erdős-Renyi random graph starts as point cloud on N nodes and insert each edge with uniform probability p, so that it can be seen as a case of unweighted link percolation on the complete graph, with probability $1-p$. This leads to the well-known result of the emergence of the giant connected component (GCC) of the Erdős-Renyi random graph emerging at the critical threshold [42]:

$$p_c = \frac{1}{N} \tag{2.41}$$

Thus, p_c controls a phase transition between a regime in which the graph is split in small, disconnected components to one where a connected component spanning the entire graph emerges.

In network science application, percolation is often used to study the robustness of transportation networks (for example, power grids, transportation systems, and communication networks), habitat fragmentation in the ecological study of animal habitats, and in opinion dynamics to characterise how consensus forms across social groups or how polarisation occurs among them.

An important idea deriving from percolation, and one that we will use in chapter 5, is the definition of a new centrality measure that is based on percolation. This centrality measure, called *percolation centrality* was introduced in [43] as a way to include the percolating state of each node into the centrality measure, as opposed to other commonly used centralities that only depend on the topology of the network, disregarding its dynamics. For each node, at a given time, the percolation centrality is defined as the proportion of *percolated paths* that pass through that node. Here, by percolated path is intended a path in which the source node (that is, the starting

node in the ordered path) is occupied (this often means, in the epidemiological language, *infected*, or in transportation networks, either *failed*, *overloaded*, or otherwise unserviceable).

Following the notation from [43], let us indicate the percolation state of of a node v at time t as:

$$x_v^t = \begin{cases} 1 & \text{if } v \text{ is in a non-percolated state at time } t \\ 0 & \text{if } v \text{ is in a percolated state at time } t \end{cases} \tag{2.42}$$

With this notation, the percolation centrality of node v is defined as:

$$\mathcal{C}_p(v) = \frac{1}{N-2} \sum_{s \neq r \neq v} \frac{\sigma_{sr}(v)}{\sigma_{sr}} \frac{x_s^t}{(\sum_i T_i^t) - x_v^t} \tag{2.43}$$

in which the sum is taken over all possible pairs of (distinct) nodes s, r that do not include v, $\sigma_{sr}(v)$ is the number of shortest paths between nodes s and r that pass through node v, $\sigma_{sr}(v)$ is the total number of shortest paths between nodes s and r, regardless of their passage through v.

Chapter 3

Ergodic sets and network compression

When we introduced random walks in section 2.2, our focus was on the mathematical formulation of the phenomenon. Here, we turn our attention to their applications. Indeed, random walks are a powerful mathematical tools that can be applied to a surprisingly large and diverse set of phenomena across seemingly unrelated areas of science, from diffusion processes in physics and chemistry [10] to forecasting stock prices and option pricing in finance [44], from population dynamics in ecology [11] to the spreading of infections and diseases in epidemiology [45]. In network science, the realm of application of random walks is similarly large. They have been used in contexts as diverse as community detection [36, 46], centrality measures [17, 47], link prediction [48], graph sampling [49], and more.

One famous example of their application is the groundbreaking work by S. Brin and L. Page [17], which introduced the modern concept of search engines on the world wide web [50] and in the wider sense of applications [51]. In the paper, Page and Brin use a random walk to calculate the network centrality of all the nodes in a directed network, assigning each node a weight that is proportional to the inverse recurrence

time of each node. On directed networks, there is an additional complexity. The existence of sources and – especially – wells (cfr. definition 5) influences the flow of the random walks so that the recurrence time is no longer well defined: in the limit of long times, the random walk moves away from the source notes and transient nodes and the probability distribution shift onto the well nodes, so that all nodes but the wells are transient, and their recurrence time becomes infinite. This problem is connected with remark 1: undirected networks are connected if and only if they are strongly connected, but this isn't true in directed networks, for which strong connectedness is a stronger property.

The authors worked around this problem with the now famous idea of substituting a random walk with a *random surfer*. The effect of this change on the dynamics of the random walk can be explained in two equivalent ways. We can see the random surfer as an instance of a teleporting random walk, which reset its position to a random node at each step with some fixed probability. Alternatively, we can describe it thought the equivalent (standard) random walk on the complete directed network with no sources nor wells obtained by adding a constant weight to all edges. This idea proved successful for the intended purpose of assign the centrality measure, but the resulting information from the random walk is degraded in the process. Each time a teleportation occurs, in fact, all the information of the walked trajectory is destroyed, as the correlation function is truncated.

Random walk applications for community detection works under similar assumptions. For example, the network clustering algorithm InfoMap use random walks through the application of the Map equation approach, which minimise the description length of a random walker's path through the network. Because the network partition is based on the analysis sequence of successive steps of the random walk, wells once again disrupt the analysis, and the method uses the same approach to the one suggested by Brin and Page [34].

A recent analysis has shown that sources and wells in directed networks are the norm rather than the exception; in other words, most real-world directed networks are connected but not strongly connected [52]. We decide to take a different approach and treat wells and sources as a feature rather than a nuisance to get rid of.

The most obvious insight they offer they offer relates to the directness of the network. Any random walk that starts in a source will, in the limit of long times, end on a well. This is connected to the hierarchical nature of directed networks. Their structure can be divided in different levels, shaped by the directed flow across the edges. This global hierarchy is the result of a local hierarchy between adjacent nodes, in which the nodes at the beginning of the directed edge can influence or control the ones at the end. This directed flow can lead to the emergence of leader nodes, gatekeeper nodes, or central nodes that occupy the highest positions in the hierarchy. Hierarchies produce cascading effects and can also shape the formation and boundaries of communities [53]. The hierarchical structure was studied first in the biological context of food chain, where it led to the definition of the trophic coherence [54], which in itself derives from the concept of trophic levels [55, 56].

This duality between local and global hierarchies reflects in the multi-scale nature of the effects of the network directness [54]. In other words, two phenomena are at play in making random walks fundamentally different on directed networks than on their undirected counterparts: a local effect – the presence of non-ergodic structures, such wells and sources – and a global effect – the emergence of hierarchies. A natural question is therefore what structures should be used to synthesise the hierarchy. There have been many attempts to define structures based on the network flow: see, for example, [57].

We believe that a natural choice is to use structures that arise directly from the random walk dynamics, as they are the random processes that are very sensitive to the change between connectedness and strong connectedness that is at the vary base

of the distinction between undirected and directed networks. We call the fundamental block of our construction *ergodic set*, a group of nodes that form a strongly connected subgraph that is connected in either way to the rest of the graph, but not in both ways. The name highlights their connection to both ergodicity and the multi-scale nature of the phenomena described. On a local level, the random walk dynamics is ergodic *on* the set; on the other, on a global level, these sets are responsible for breaking the ergodicity on the network as a whole.

In section 3.1 we give a more formal definition of ergodic sets, starting from purely structural considerations of directed networks. In section 3.2 we connect these structures with other structures defined on directed networks, chiefly strong communicating classes and strongly connected components, and we show that ergodic sets are an intermediate structure, more restrictive than the former but less restrictive than the latter. We then use ergodic sets to propose a new, dynamics-based compression algorithm in section 3.3, and apply it to both models of random network (section 3.4) and real-world networks taken from a variety of applications (section 3.5). Finally, we conclude the chapter with an analysis between the interplay of the core-periphery structure of the network and our algorithm in section 3.6.

3.1 Idea and informal introduction

The fundamental idea behind the definition of ergodic sets is the generalization of the concept of sources and sinks. In this sense, ergodic sets are more complex structures which behaves very similarly to sources and sinks with respect to random walks while not satisfying the definition of being either source nor a sink. Perhaps the simplest possible example is a proper sink v in a graph $\mathcal{G}$, to which a vertex w and the directed edges (v, w) and (w, v) are added. The resulting nodes pair $\{v, w\}$ does not satisfy the definition of a sink (as it's a subgraph, not a vertex), nor does any single vertices

v, w on their own (as neither has zero out-degree). Yet, a random walker entering the subgraph $X = \{v, w\}$ would be unable to escape, just as it would happen in the case of a proper sink. Despite the simplicity of this example, it is already clear that there is room for generalizing the concepts of sources and sinks in directed networks.

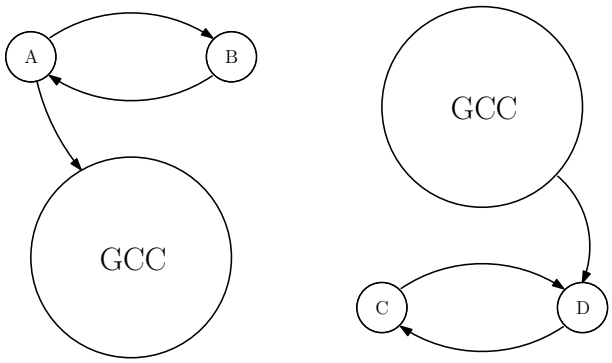


Figure 3.1: Example of simple ergodic sets $\{A, B\}$ and $\{C, D\}$ that do not satisfy the conditions of being a sink or a source. Here GCC stands for the graph's giant connected component, a strongly connected subgraph that contains most of the nodes of the network. No individual node in the graphs would satisfy definition 5 of source or well node.

We can see the example described above in figure 3.1, in which we are generalising a source (left) and a sink (right). Both the ergodic sets and hierarchical methods would be able to identify the structure. Because of this, we can already see how ergodic sets are more powerful than the notion of sinks and well and are able to identify global flow properties while being a local construction.

On the other hand, not all nodes in the networks are accounted for in such framework. The giant connected component is left untouched and so are nodes that sits in between connections among different strongly connected components. See figure 3.2 for an example.

This on purpose: firstly, because a complete model of complex networks that accounts for all nodes already exists, in the form of the bow-tie representation of the World Wide Web, proposed in [2]. Secondly, because this representation only focuses on the nodes that are dynamically relevant **??**, so we are not interest in the behaviour

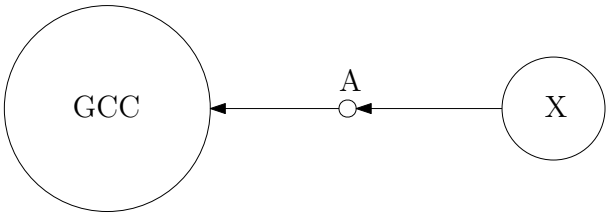


Figure 3.2: Ergodic sets do not account for all vertices in a graph. Node A, for example, sitting along the path from one ergodic set X (read: strongly connected component for now, see section 3.2) to the giant (strongly) connected component.

of transient nodes like node A in figure 3.2. By sacrificing the ability of including all nodes in the network, we are able to scale the method to large network efficiently and obtain network compression while preserving the dynamics, see section 3.3. The bow-tie representation, despite describing the network more comprehensive, is a conceptual tool that lacks this possibility.

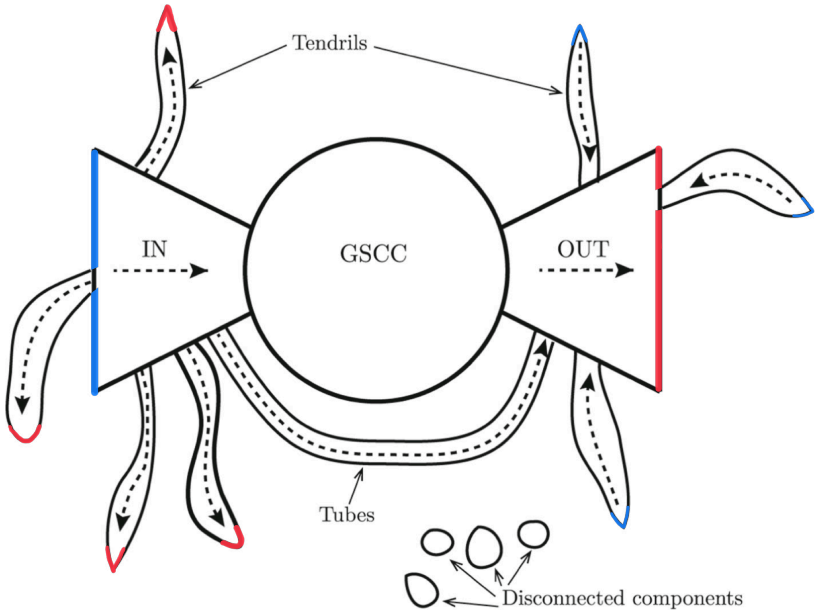


Figure 3.3: Graphical representation of the input and output sets in relation to the bow-tie representation of the World Wide Web. The inputs sets are represented in blue, while the output sets are represented in red.

3.2 Erogdic sets: structural definition in directed networks

We now lay the foundation upon which we will build a formal definition of ergodic sets in the context of a directed graph. We first introduced them based on purely structural (i.e. non dynamical) considerations, then we clarify its connection with the random walk dynamics.

Definition 26. *Let $\mathcal{G} = (V, E)$ be a directed graph. We say $X \subseteq V$ is an **ergodic set** if X is strongly connected and **either** of the two following sets of conditions hold:*

1. $X = V$
2. $X \subset V$ *and* ***all*** *the following conditions hold:*
   (a) *if* $\exists x^* \in X$, $v^* \in V \setminus X$ *such that* $(x^*, v^*) \in E$, *then*

$$\forall x \in X, \forall v \in V \setminus X \Rightarrow (v, x) \notin E \tag{3.1}$$

   (b) *if* $\exists x^* \in X$, $v^* \in V \setminus X$ *such that* $(v^*, x^*) \in E$, *then*

$$\forall x \in X, \forall v \in V \setminus X \Rightarrow (x, v) \notin E \tag{3.2}$$

We further specialize the ergodic sets in two cases, according to which of the two conditions (2a) and (2b) in definition 26 the subset satisfy.

Definition 27. *We say X is an **output** if X is an ergodic set for which condition (2a) is an empty condition (i.e. the hypothesis is not satisfied, formally $\nexists\, x^* \in X$, $v^* \in V \setminus X$ such that $(x^*, v^*) \in E$). We say X is an **input** if X is an ergodic set for which condition (2b) is an empty condition.*

Before we continue, there is an important remark about the notation. Due to the different disciplines that independently developed the ideas upon which the concept of ergodic sets are based, there are different and competing notations. In our work, we will use interchangeably the terms input, source, and backward ergodic set and the terms output, sink, forward ergodic set. The former expressions refer to the idea of a subgraph to which one can never return once it is left, while the latter expressions identify a subgraph that can never be left once reached.

Note how definition 26 and definition 27 formalise the idea we have already introduced informally: ergodic sets are strongly connected subgraphs that a random walk can reach but never leave (definition 26, condition (2a)), or leave but never reach (definition 26, condition (2b)). We now want to connect ergodic sets with the concepts (introduced in chapter 2) of strongly connected component and closed communicating classes. It is obvious from definition 26 that the property of a subset $X \subseteq V$ of being a strongly connected component of $\mathcal{G}$ is more general. We aim to prove here that an ergodic set is more general a requirement than a closed communicating class.

Proposition 3. *Let $X \subseteq V(\mathcal{G})$ be a closed communicating class. X is an ergodic set.*

Proof. X is a closed communicating class by hypothesis. If $X = V$, X strongly connected, so X is an ergodic set. So, let's suppose $X \subsetneq V$. We verify conditions (2a) and (2b) separately.

(2a) we show that the hypothesis can never be satisfied, making this an empty condition which is always true. Suppose, by contradiction, that $\exists\, x^* \in X, \exists\, v^* \in V \setminus X$ such that $(x^*, v^*) \in E$. Since X is a closed communicating class, $x^* \in X$, $x^* \to v^* \Rightarrow v^* \in X$, in contradiction with having chosen $v^* \in V \setminus X$.

(2b) suppose that $\exists\, x^* \in X$, $\exists\, v^* \in V \setminus X$ such that $(v^*, x^*) \in E$. We need to prove that $\forall x \in X$, $\forall v \in V \setminus X$, $(x, v) \notin E$. Suppose, by contradiction, that $\exists\, x' \in X$,

$\exists\, v' \in V \setminus X$ such that $(x', v') \in E$. This implies $x' \to v'$ and so, from closure of X, we conclude $v' \in X$, contradiction.

Therefore, X is an ergodic set. □

Proposition 4. *Not all ergodic sets are closed communicating classes.*

Proof. We give the following counterexample. Consider the strongly connected subset $X \subset V$, connected to the rest of the graph, which is also strongly connected.

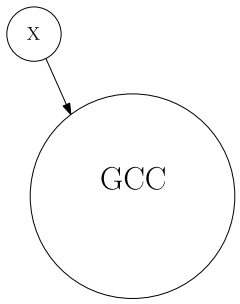


Figure 3.4: Counterexample. Both X and GCC are strongly connected components.

Therefore, X is an ergodic set (specifically, an input) but not a closed communicating class. Hence the result. □

Intuitively, this result is quite obvious: since ergodic sets are more general than closed communicating classes, the set of all closed communicating class $\mathcal{V}_{CC}$ is a strict subset of the set of all ergodic sets $\mathcal{V}_{ES}$. We have already proven the inclusion $\mathcal{V}_{CC} \subsetneq \mathcal{V}_{ES}$. We will now show a stronger result, namely that the set of all ergodic set is the direct sum of the set of all closed communicating classes and the set of all closed communicating classes of the inverse graph.

Proposition 5. *Let $\mathcal{G}$ be a connected directed graph. The set $\mathcal{V}(\mathcal{G})_{ES}$ of all ergodic sets of $\mathcal{G}$ is the union of the set of all closed communicating class and the set of all closed communicating class of the inverse graph $\mathcal{G}'$.*

$$\mathcal{V}(\mathcal{G})_{ES} = \mathcal{V}(\mathcal{G})_{CC} \cup \mathcal{V}(\mathcal{G}')_{CC} \tag{3.3}$$

Moreover, if the graph $\mathcal{G}$ is not strongly connected, the union becomes a direct sum:

$$\mathcal{V}(\mathcal{G})_{ES} = \mathcal{V}(\mathcal{G})_{CC} \oplus \mathcal{V}(\mathcal{G}')_{CC} \tag{3.4}$$

Corollary 1. *If the graph $\mathcal{G}$ is connected but not strongly connected, the set of all the closed connected classes of the network and $\mathcal{V}(\mathcal{G})_{CC}$ and the set of all the closed connected classes of the inverse network $\mathcal{V}(\mathcal{G}')_{CC}$ are disjoint:*

$$\mathcal{V}(\mathcal{G})_{CC} \cap \mathcal{V}(\mathcal{G}')_{CC} = \emptyset \tag{3.5}$$

Remark 5. *The corollary is necessary for the derivation of the second result in proposition 5 and will therefore be proven during the proof of proposition 5.*

Proof. To prove the first part of the proposition, we show both inclusions

$$\mathcal{V}(\mathcal{G})_{ES} \subseteq \mathcal{V}(\mathcal{G})_{CC} \cup \mathcal{V}(\mathcal{G}')_{CC} \quad \text{and} \quad \mathcal{V}(\mathcal{G})_{ES} \supseteq \mathcal{V}(\mathcal{G})_{CC} \cup \mathcal{V}(\mathcal{G}')_{CC}$$

We start from the first one: we claim $X \in \mathcal{V}(\mathcal{G})_{ES} \Rightarrow X \in \mathcal{V}(\mathcal{G})_{CC} \cup \mathcal{V}(\mathcal{G}')_{CC}$. To prove this, let X be an ergodic set. If $X = V$, then $X \in \mathcal{V}(\mathcal{G})_{ES}$ and $X \in \mathcal{V}(\mathcal{G})_{CC}$ so there is nothing to prove. So, let's assume $X \subset V$. Hence, X must satisfy the two condition (2a) and (2b) of definition 26.

We firstly show that at least one of the two conditions is nonempty. Suppose, by contradiction, that both conditions are empty, namely: $\nexists x^* \in X$, $v^* \in V \setminus X$ such that $(x^*, v^*) \in E$ and $\nexists x^* \in X$, $v^* \in V \setminus X$ such that $(v^*, x^*) \in E$. This implies that there are no edges (in either direction) between X and the rest of the graph $V \setminus X$. But $\mathcal{G}$ is connected, contradiction. Furthermore, we recall that the two conditions cannot be empty at the same time, as this leads to a contradiction between the two hypothesis and the two theses. Hence, exactly one condition among (2a) and (2b) is non-empty.

We claim that, if condition (2a) is non-empty, then $X \in \mathcal{V}(\mathcal{G})_{CC}$. If, otherwise, condition (2b) is non-empty, then $X \in \mathcal{V}(\mathcal{G}')_{CC}$.

Suppose condition (2a) is non-empty. Hence, $\forall x \in X, \forall v \in V \setminus X \Rightarrow (v, x) \notin E$. We need to prove that X is a closed communicating class of $\mathcal{G}$ according to definition 15, namely that if $v \in X$, $v \to w$ in $\mathcal{G}$, then $w \in X$. This is now obvious: since there are no edges leaving X toward the rest of the graph because of condition (2a), all paths starting in v can never escape X, hence we must have $w \in X$.

Now suppose condition (2b) is non-empty. Hence, $\forall x \in X, \forall v \in V \setminus X \Rightarrow (x, v) \notin E$. We need to prove that X is a closed communicating class of $\mathcal{G}'$ according to definition 15, namely that if $v \in X$, $v \to w$ in $\mathcal{G}'$, then $w \in X$. As per definition 2, we know that the inverse graph $\mathcal{G}'$ is obtained by swapping the ordering of all the edges of the graph. Hence, this is equivalent to prove that $v \in X$, $w \to v$ in $\mathcal{G}$ implies $w \in X$. Since there are no edges of $\mathcal{G}$ entering into X form the rest of the graph, all paths ending in v must have originated from a vertex in X, so that the condition implies $w \in X$. This concludes the proof of the first inclusion, $\mathcal{V}(\mathcal{G})_{ES} \subseteq \mathcal{V}(\mathcal{G})_{CC} \cup \mathcal{V}(\mathcal{G}')_{CC}$.

We now prove the second inclusion, $\mathcal{V}(\mathcal{G})_{ES} \supseteq \mathcal{V}(\mathcal{G})_{CC} \cup \mathcal{V}(\mathcal{G}')_{CC}$. We need to prove $X \in \mathcal{V}(\mathcal{G})_{CC} \cup \mathcal{V}(\mathcal{G}')_{CC} \Rightarrow X \in \mathcal{V}(\mathcal{G})_{ES}$.

Let us suppose $X \in \mathcal{V}(\mathcal{G})_{CC}$, that is, X is a closed communicating class of $\mathcal{G}$. By proposition 3, X is an ergodic set, hence $X \in \mathcal{V}(\mathcal{G})_{ES}$. If $X \notin \mathcal{V}(\mathcal{G})_{CC}$, then by hypothesis $X \in \mathcal{V}(\mathcal{G}')_{CC}$. This implies $v \in X$, $v \to w$ in $\mathcal{G}' \Rightarrow w \in X$. This statement is equivalent to $v \in X$, $w \to v$ in $\mathcal{G} \Rightarrow w \in X$. We need to prove that these conditions are sufficient to make X an ergodic set of $\mathcal{G}$.[1]

Following the same proof as in proposition 3, if $X = V$, then X strongly connected because of the closed communicating class property (this time on the inverse graph $\mathcal{G}'$). Hence, X is an ergodic set of $\mathcal{G}$. Thus, we only need to prove the thesis for

[1]Note that we can not use proposition 3 again here. That would only prove that $X \in \mathcal{V}(\mathcal{G}')_{ES}$. Indeed, we have $\mathcal{V}(\mathcal{G})_{ES} = \mathcal{V}(\mathcal{G}')_{ES}$, but this is a consequence of proposition 5, see corollary 2. Therefore, we can't use this equality to go further, and we need to prove the claim by directly testing definition 26.

$X \subsetneq V$.

To do so, we prove that both conditions (2a) and (2b) from definition 26 hold. Starting from condition (2a), let us suppose that $\exists x^* \in X$, $v^* \in V \setminus X$ such that $(x^*, v^*) \in E(\mathcal{G})$. We need to prove that $\forall x \in X, \forall v \in V \setminus X \Rightarrow (v, x) \notin E(\mathcal{G})$. Suppose, by contradiction, that $\exists x \in X$, $x \neq x^*$, $\exists v \in V \setminus X$, $v \neq v^*$ such that $(v, x) \in E(\mathcal{G})$. So, when we take the inverse graph, we have $(x, v) \in E(\mathcal{G}')$ for those same $x \in X$, $x \neq x^*$, $v \in V \setminus X$, $v \neq v^*$. By closure of X in $\mathcal{G}'$, we have $x \in X$, $x \to v$, hence $v \in X$. However, we assumed $v \in V \setminus X$, absurd.

Moving to condition (2b), we see that this is an empty condition. Suppose, by contradiction, that $\exists x^* \in X$, $v^* \in V \setminus X$ such that $(v^*, x^*) \in E(\mathcal{G})$. When taking the inverse graph, we would get $(v^*, x^*) \in E(\mathcal{G}')$ for those same $x^* \in X$, $v^* \in V \setminus X$. This would imply, by closure of X in $\mathcal{G}'$, $v^* \in X$, absurd. Hence, condition (2b) is always satisfied.

This concludes the proof of the second inclusion, $\mathcal{V}(\mathcal{G})_{ES} \supseteq \mathcal{V}(\mathcal{G})_{CC} \cup \mathcal{V}(\mathcal{G}')_{CC}$.

We have hence completed the proof of the first part of the proposition.

We now move on to prove the second claim, namely that if $\mathcal{G}$ is not strongly connected, then $\mathcal{V}(\mathcal{G})_{ES} = \mathcal{V}(\mathcal{G})_{CC} \oplus \mathcal{V}(\mathcal{G}')_{CC}$. Because of the first part, we only need to prove that, if $\mathcal{G}$ is not strongly connected, then $\mathcal{V}(\mathcal{G})_{CC} \cap \mathcal{V}(\mathcal{G}')_{CC} = \emptyset$.

To do so, we need to prove the two implications $X \in \mathcal{V}(\mathcal{G})_{CC} \Rightarrow X \notin \mathcal{V}(\mathcal{G}')_{CC}$ and $X \in \mathcal{V}(\mathcal{G}')_{CC} \Rightarrow X \notin \mathcal{V}(\mathcal{G})_{CC}$.

Suppose $X \in \mathcal{V}(\mathcal{G})_{CC}$. Since X is strongly connected and $\mathcal{G}$ is not, there exist $v \in V \setminus X$. Since $\mathcal{G}$ is connected, we can have, for an appropriate $x \in X$, $x \to v$ or $v \to x$. If we had $x \to v$, by closure of X in $\mathcal{G}$, we would conclude $v \in X$, absurd. Hence, the second case needs to hold, $v \to x$. Suppose, by contradiction, $X \in \mathcal{V}(\mathcal{G}')_{CC}$. Then we would have, for closure of X in $\mathcal{G}'$, $v \in X$, absurd. Hence, we conclude that $X \notin \mathcal{V}(\mathcal{G}')_{CC}$.

The second implication $X \in \mathcal{V}(\mathcal{G}')_{CC} \Rightarrow X \notin \mathcal{V}(\mathcal{G})_{CC}$ follows the same proof. This, in turns, concludes the second and last part of the proof. □

Corollary 2. *For any connected graph $\mathcal{G}$, the following equation holds:*

$$\mathcal{V}(\mathcal{G})_{ES} = \mathcal{V}(\mathcal{G}')_{ES} \tag{3.6}$$

Proof. The claim follows directly from proposition 5 and the fact that taking the inverse of a graph is an involutory transformation, that is, it's equal to its inverse. Therefore, $(\mathcal{G}')' = \mathcal{G}$. In fact:

$$\mathcal{V}(\mathcal{G}')_{ES} = \mathcal{V}(\mathcal{G}')_{CC} \cup \mathcal{V}\left((\mathcal{G}')'\right)_{CC} = \mathcal{V}(\mathcal{G}')_{CC} \cup \mathcal{V}(\mathcal{G})_{CC} = \mathcal{V}(\mathcal{G})_{CC} \cup \mathcal{V}(\mathcal{G}')_{CC} = \mathcal{V}(\mathcal{G}')_{ES}$$

□

Remark 6. *All results in this section assume the graph $\mathcal{G}$ is connected. This condition can be relaxed by applying the same results to each connected components and taking the union set across the ergodic sets of all different directed components.*

We have so far introduced a formal definition of ergodic sets, based only on the network structural properties, as well as their connections with strongly connected components and closed communicating classes. In the remainder of this section, we will rephrase all the results shown so far in terms of the graph adjacency matrix (see definition 20). This step does not introduce new results but sets the ground for using ergodic sets to define a compression algorithm for the network. Matrices are a more convenient language that will be used throughout section 3.3, where we discuss the algorithm.

Here we give a definition of an ergodic sets, equivalent to definition 26, using the graph adjacency matrix and we prove the equivalency of the two. This will have important consequences on the ergodic sets detection problem, see section 3.3.1.

Proposition 6. *Let $\mathcal{G}$ be a [directed] graph and let $A = A(\mathcal{G})$ be its adjacency matrix. Let X be a proper subset of vertices of $V(\mathcal{G})$. Then, X is an ergodic set if and only if X is strongly connected and **either** of the two following sets of conditions hold:*

1. *$X = V$*
2. *$X \subset V$ and **all** the following conditions hold:*
   (a) *if the submatrix $A_{V\setminus X,V}$ has a nonzero element, then the submatrix $A_{X,V\setminus X}$ is zero (element-wise)*
   (b) *if the submatrix $A_{X,V\setminus X}$ has a nonzero element, then the submatrix $A_{V\setminus X,V}$ is zero (element-wise)*

Proof. The proposition is a restatement of definition 26 in terms of the adjacency matrix. Because of this, we only need to prove that the two versions of condition (2a) and (2b) are equivalent. This follows from the definition of the adjacency matrix. In particular, given any two subsets of vertices $V_1, V_2 \subset V(\mathcal{G})$, we have that:

$$A_{V_1,V_2} \text{ has a nonzero element } \Leftrightarrow \exists v_1 \in V_1, v_2 \in V_2 \text{ such that } (v_1, v_2) \in E(\mathcal{G}) \quad (3.7)$$

$$A_{V_1,V_2} \text{ is zero (element-wise) } \Leftrightarrow \forall v_1 \in V_1, v_2 \in V_2, \quad (v_1, v_2) \notin E(\mathcal{G}) \quad (3.8)$$

Thus, conditions (2a) and (2b) are equivalent to their counterparts in definition 26, so that the definition is equivalent to proposition 6. □

Remark 7. *All results derived from definition 26 can be derived in the adjacency matrix formalism from proposition 6. All proofs are the same, except for the different expression of conditions (2a) and (2b).*

This simple equivalence introduced in proposition 6 is just a rephrasing in terms of the adjacency matrix, but it is at the foundation of the algorithm we use to detect the ergodic sets (see section 3.3.1).

Finally, let's briefly touch on an ergodic sets property that we will extend to define a distance on ergodic sets. Given two vertices v_i, v_j in a graph $\mathcal{G}$, and given an unbiased random walker on $\mathcal{G}$, we define the dynamical distance between the two vertices as follows:

$$d(v_i, v_j) = \frac{|\lim_{t\to+\infty}(\mathbb{1}_i P^t) - \lim_{t\to+\infty}(\mathbb{1}_j P^t)|}{|\lim_{t\to+\infty}\mathbb{1}_i P^t| + |\lim_{t\to+\infty}\mathbb{1}_j P^t|} \tag{3.9}$$

It is easy to see that the definition does not satisfy all the requirements of a distance. In particular, it does not satisfy the identity of indiscernible property, namely $d(x, y) = 0 \Leftrightarrow x = y$. Thus, the dynamical distance is actually a pseudo-distance. In order for it to become a distance, we would need to move to equivalence classes; whose defining equivalence relation is:

$$\forall v, w \in V(\mathcal{G}) \quad v \sim w \text{ iff } \lim_{t\to+\infty} \mathbb{1}_v P^t = \lim_{t\to+\infty} \mathbb{1}_w P^t \tag{3.10}$$

This satisfies the properties of reflexivity, symmetry, and transitivity necessary for an equivalence relation, and trivially solves the problem with the property of the identity of indiscernible of d, as two vertices that have zero distance are automatically in the same equivalency class, and therefore quotient to the same element in the quotient space $V_\sim$. The next step with this derivation will be to study the equivalency classes, and to see if they hold a significant correlation with other network structures, such as communities or frequently visited paths. As a further step, should we find a relation between these equivalency classes, we aim to define a way to study and quantify such correlation.

3.3 Ergodic Sets Compression Algorithm (ESCA)

The aim is now to use the ergodic sets defined in the previous sections to propose a new algorithm to efficiently compress directed networks, while using the dynamical properties of ergodic sets to preserve the random walk dynamics on the network during the process.

We call our compression algorithm ESCA, short for *Ergodic Sets Compression Algorithm*, because the fundamental idea behind it is to coarse-grain each ergodic set into a single node and rewire the edges of the graph in such a way that the marginal transition probabilities, conditioned upon the random walk leaving from (or arriving at) the coarsed-grained node, are preserved during the process.

We divide the description of the algorithm in three steps, which are the three subsequent problems we encounter in the process described.

1. **ergodic sets detection**: given a directed graph, how can we identify the ergodic sets as defined in definition 26? In other words, given a graph $\mathcal{G} = (V, E)$ how do we find the set of sets $\{X_1, X_2, \ldots, X_n\}$ in which X_i is an ergodic set according to definition 26?

2. **ergodic sets compression**: after we have identified the graph ergodic sets $\{X_1, X_2, \ldots, X_n\}$, how do we rewire the graph in a way that the in- or out- transition probability is conserved? More formally, given X_i output (resp. input) set of $\mathcal{G}$ and the set of edges $\{e_1, \ldots, e_m\}$ in the form $e_j = (x, v)$, $x \in X_i$, $v \in V \setminus X_i$ [resp. $e_j = (v, x)$], how do we define the edges in the form $e_j = (x^*, v)$ [resp. $e_j = (v, x^*)$], in which x^* is the coarse-grained node representing X_i?

3. **core compression**: how do we efficiently coarse-grain the rest of the network $V \setminus (\cup_{i=1}^{n} X_i)$ while also preserving the dynamics?

3.3.1 Step I: ergodic sets detection

The problem we encounter when dealing with ergodic sets in an applied context takes the form of a detection problem. In other words, how can we detect the network subgraphs satisfying the conditions in definition 26? By *detecting* we mean the following: given a graph $\mathcal{G}$, find a list $\mathcal{X} = \{X_1, \ldots, X_m\}$ of all ergodic sets X_i in the graph. Note that m is not known *a priori* and it's part of the detection problem.

To do so, we first search for the strongly connected components of the network. We then use the result from proposition 5 to limit our search inside the subset of all strongly connected components of $\mathcal{G}$. Algorithms to detect the strongly connected components of a directed graph have been extensively studied in the literature, we use Tarjan's algorithm [58] with Nuutila's [59] modifications.

This algorithm returns a list $\mathcal{C} = \{C_1, \ldots, C_n\}$ in which C_i are all the strongly connected component of $\mathcal{G}$. By using definition 26, point 1, we know that $\mathcal{X} \subseteq \mathcal{C}$ as if X_i is an ergodic set of the network, there exists $C_j \in \mathcal{C}$ such that $X_i = C_j$.

From now on, we are assuming weak connectedness without loss of generality: if $\mathcal{G}$ is not weakly connected, we can just treat each weakly connected component separately; if $\mathcal{G}$ is strongly connected, the whole graph is a single ergodic set, and the decomposition is trivial, $\mathcal{X} = \mathcal{C} = \{V\}$.

All is left to do now is to implement an algorithm to efficiently check whether the strongly connected components C_j satisfy the additional conditions in the definition of ergodic sets, equations (3.1) and (3.2). To do so, we consider each element $X \in \mathcal{C}$ separately and we use proposition 6 to check the sum of the in- and out-degrees of all nodes in the set $v \in X \subseteq V$. We compare the in- and out-degrees in the subgraph X properly and in X as induced subgraph X_I. We claim that we can rewrite the conditions we are checking using the following relations:

$$\left(\sum_{v\in X} d_{out}(v) - \sum_{v\in X_I} d_{out}(v) \neq 0 \Rightarrow \sum_{v\in X} d_{in}(v) - \sum_{v\in X_I} d_{in}(v) = 0\right) \Leftrightarrow \text{equation (3.1)} \tag{3.11}$$

$$\left(\sum_{v\in X} d_{in}(v) - \sum_{v\in X_I} d_{in}(v) \neq 0 \Rightarrow \sum_{v\in X} d_{out}(v) - \sum_{v\in X_I} d_{out}(v) = 0\right) \Leftrightarrow \text{equation (3.2)} \tag{3.12}$$

which allows us to quickly and efficiently test whether the strong connected component is also an ergodic set.

Note that equation (3.11) and equation (3.12) are just a rewriting of equation (3.1) and equation (3.2) respectively, so there is nothing to prove. To see that the rewriting in equations (3.11) and (3.12) is correct, note that for any subgraph $Y \subseteq V$ and the corresponding induced subgraph Y_I, the expressions above are counting all the internal and outgoing [resp. incoming] edges (when we are summing the subgraph degrees), then we are subtracting the sum of the inner edges only (when summing the induced subgraph degrees). Hence, we can check the conditions of definition 26 by simply counting incoming and outgoing edges, as follows:

$$\sum_{v\in Y} d_{out}(v) - \sum_{v\in Y_I} d_{out}(v) = \#\text{ of edges from } Y \text{ to the rest of the graph } \mathcal{G}\setminus Y \tag{3.13}$$

$$\sum_{v\in Y} d_{in}(v) - \sum_{v\in Y_I} d_{in}(v) = \#\text{ of edges to } Y \text{ from the rest of the graph } \mathcal{G}\setminus Y \tag{3.14}$$

We report here the complete pseudo-code for the detection algorithm:

This algorithm solves the detection problem and returns a complete list $\mathcal{X}$ of all ergodic sets in the graph, thus completed in the first step of the graph coarse-graining procedure.

Algorithm 2: Detecting an Ergodic Set

Input : $\mathcal{G}(V, E), V \supset X$ strongly connected
Output: Yes/No

```
Function IsErgodicSet(X)
    X_I ← G[X]  Δ_X ← 0
    for x ∈ X do
        Δ_X ← Δ_X + d_X(x) − d_{X_I}(x)
    if Δ_X = 0 then
        return True
    else
        return False
```

Remark 8. *Note that the detection algorithm is symmetric for input and outputs, i.e. the same procedure can be followed to detect both types of ergodic sets. This will not be the case during the rewiring phase, in which the edge directionality breaks the problem symmetry.*

After the completion of the ergodic sets detection step, the network is divided into the direct sum of a number of inputs (forward ergodic sets) and outputs (backwards ergodic sets) and a central core, see figure 3.5. Note that the resulting network partition is different to the traditional bow-tie structure. For example, the network core is not – in general – strongly connected, while the bow tie "knot" is; furthermore, the concepts of tendrils and tubes disappear in favour of a finer subdivision of the input and output structures.

3.3.2 Ergodic sets coarse-graining rules

The next step in the Ergodic Sets Compression Algorithm is to coarse-grain the strongly-connected ergodic sets into single nodes and modify the network edges in order to preserve the random-walk dynamics. The compression rules are generated separately for outputs and inputs. In summary, we proceed as follows. For inputs, we equate the probabilistic 1-step in-flow from all the nodes that are the origin of any

edge ending in one node of any backward ergodic set. For outputs, we simply copy the outflow from any node that has been clustered into the input. We now provide the complete rules.

Outputs coarse-graining rules

If X is an output, collapse all vertices in a single node n_X. This will amount to the deletion of all vertices $v \in X$ and the addition of the new node v_X with no in- or out-edges. This vertex inherits the incoming links from all the nodes $w \in V \setminus X$ for which - before the replacement - there exists $v \in X$ such that $(w, v) \in E(\mathcal{G})$. A weight is assigned to each edge (which is a standard procedure when coarse-graining networks, see for example [28]): from a given w, it is equal to the number of links from w to the nodes in X - in the case of weighted networks, it would be the sum of the weights of links to nodes in X. The new weights can further be normalised using the normalization $\sum d_{out}(v_X) = 1$.

No out-edges are added leaving v_X, in accordance with the definition of output: this is mirrored by the fact that, before the substitution of X with n_X, there were no edges from any vertex $v \in V \setminus X$ to any vertex $w \in X$. The following pseudo-code illustrate the algorithm step-by-step.

To check that the node n_X has the same dynamical properties of the set X (specifically, the same transition probabilities from external nodes and same hitting time), we can observe that the transition probabilities from any vertex $v \in V \setminus X$ to X have been preserved. In fact, given a random walker Y on $\mathcal{G}$:

$$\mathbb{P}\left(Y_{n+1} \in X | Y_n = v\right) = \mathbb{P}\left(Y_n = v\right) \frac{|\{e = (v, x)\, s.t.\, x \in X\}|}{d_{out}(v)} = \mathbb{P}\left(Y_n = v\right) \omega_X \quad (3.15)$$

in which the last equality holds because of the rewiring rule.

This means that the choice of the weights preserves the marginal 1-step probability

from all the nodes that are the origin of any edge ending in any node of X, as required.

Inputs coarse-graining rules

If X is an input, collapse all vertices into a single node n_X. This is equivalent to the deletion of all nodes $v \in X$ and the introduction of a single new node v_X with no in- or out-edges. Then, we connect all the nodes $w \in V \setminus X$ for which - before deletion - there exists $v \in X$ such that $(v, w) \in E(\mathcal{G})$. Then, set the weights of all the out-edges introduced in the second step above as follows: for any edge (w, n_X), its weight ω_w is

$$\omega_w = \frac{1}{\sum_{v \in X} d_{in}(v)} \sum_{v \in X} \frac{d_{in}(v)}{d_{out}(v)} \mathbb{I}_E(v, w) \tag{3.16}$$

in which the degrees are considered before the deletion of the X subgraph, and the indicator function $\mathbb{I}_E(v, w)$ is defined as follows:

$$\mathbb{I}_E(v, w) = \begin{cases} 1 & \text{if} (v, w) \in E \\ 0 & \text{otherwise} \end{cases} \tag{3.17}$$

Intuitively, equation (3.16) is derived by setting the outgoing probability from n_X to any vertex $w \in V \setminus X$ equal to the sum over all $v \in X$ of the equilibrium probability of being in v times the probability of reaching w in one step. More formally, we want to

It is clear by construction that the reachable vertices w, as well as the probability of reaching them from the ergodic set X does not change when substituting X with its coarse-grained vertex v_X. In fact, we can rebuild equation (3.16) from the ground up from a random walker Y on $\mathcal{G}$:

$$\mathbb{P}\left(Y_{n+1} = w_i | Y_n \in X\right) = \mathbb{P}\left(Y_n = x \in X\right) W_{x, w_i} \tag{3.18}$$

in which W_{ij} is the random walk transition matrix, equivalent to the row-normalised

adjacency matrix A_{ij} of the graph.

The probability $\mathbb{P}(Y_n = x \in X)$ only depends on the structure of the input set X, and can be written explicitly as:

$$\mathbb{P}(Y_n = x \in X) = \frac{d_{in}(x)}{\sum_{x' \in X} d_{in}(x')} \tag{3.19}$$

in which the first-degree approximation of the PageRank centrality measure of the node is used to write the closed-form equation (3.19). The complete expression for the edge weights leaving the coarsed grained input can therefore be found by summing the quantity in equation (3.19) over all nodes of the set, multiplied by the exiting probabilities, which take the form:

$$W_{x,w_i} = \frac{\mathbb{I}_E(x, w_i)}{\sum_{x' \in X} \mathbb{I}_E(x', w_i)} = \frac{\mathbb{I}_E(x, w_i)}{d_{out}(x)} \tag{3.20}$$

Combining equation (3.20) and equation (3.19) yields the explicit expression of the weights as in equation (3.16).

The rules above are all we need to replicate the correct exit probabilities of any input X to the coarse-grained node n_X. However, they lead to an exit time of 1. Most of the times, this is not a problem, and indeed can be convenient if we don't care about the dynamics inside the input. If we want to preserve the average exit time, we have an additional degree of freedom to exploit: the weight ω_s of a self-loop for v_X. Using the fact that the exit time of a random walk is an exponential variable, it follows that we need to set ω_s such that:

$$\frac{\omega_s}{\omega_s + \sum_i \omega_i} = \sum_{v \in X} \frac{d_{in}(v)}{\sum_{w \in X} d_{in}(w)} \frac{d^{\dagger}_{out}(v)}{d_{out}(v)} \tag{3.21}$$

in which we have indicated the (out-)degrees of the vertices in X as a proper subset of V with $d(v)$, while $d^{\dagger}(v)$ indicates the (out-)degree of v as a vertex of X_I, the

subgraph induced by X.

Finally, a quick note on the quantities involved. While the notation is not the most user-friendly, all the quantities involved can be easily calculated in linear time using the appropriate data structure for the graph (as they are just degrees) and so there is no major barrier to apply the method for large graphs.

At the end of the first step, we have replaced the original network with a *compressed network* where each ergodic set, input or output, is a node. By construction, these nodes are sinks (or forward ergodic sets) - subsets with zero out-degree - and sources (or backwards ergodic sets) - subsets with zero in-degree - respectively. We denote by N_{bw}, N_{fw} and N_{core} the number of backwards ergodic sets, the number of forward ergodic sets, and number of nodes in the core respectively. At that stage, we have retained all the nodes that are part of the transient core. This feature, together with the fact that we have retained the product of the marginal probabilities times the transition probabilities, means that this simplified network exactly described how probability percolates from the inputs to the outputs via the core. The structure of the graph is shown in figure 3.9.

3.3.3 Step II: core SVD compression

Each of the three network components that we have identified at the end of the first compression step (see figure 3.9) is associated with its own network flow. Hence, the dynamic on the network can be decomposed into three regimes: the inflow from the backward ergodic sets to the core, the mixing in the core, and the outflow to the forward ergodic sets. These flows take place one after the other and the transition from one to the next is governed by the hitting time of the network core and the forward ergodic sets. Because node hitting times are stopping times in the sense of Markov chains [16], the overall flow can be factorised and decomposed into a sequence

of three independent flows.

As a next step, we consider the asymptotic of the process and consider the core as a black box connecting backward and forward ergodic sets, respectively interpreted as inputs and outputs. Let us consider the compressed network obtained at the end of the first step. As a starting point, we consider a rectangular matrix capturing the probability to finish at a certain sink when starting on a certain source, and then find ways to factorise it in order to uncover the dominant patterns of mixing by the core. The *core mixing matrix* is defined as:

$$B_{ij} = \lim_{t\to\infty} p(y_j, t|x_i, 0), \tag{3.22}$$

that is the probability to be absorbed by one of the sinks y_i when starting on a source x_i. By definition, $\sum_j B_{ij} = 1$.

A natural way to reduce the dimensionality of the system even further and to uncover the dominant patterns in the mixing matrix is to perform an SVD [60]

$$B = M_{bw}\, C\, M_{fw}. \tag{3.23}$$

where $M_{bw} \in \mathbb{R}^{N_{bw}\times k}$ and $M_{fw} \in \mathbb{R}^{k\times N_{fw}}$ encode the positions of the sources and the sinks in a k-dimensional latent space, and where the diagonal matrix $C \in \mathbb{R}^{k\times k}$ encodes the importance of each dimension in that space.

In doing this, from an original network of N nodes, we have first constructed a compressed network made of $N_1 = N_{bw} + N_{fw} + N_{core} \leq N$ nodes, and then, as a second step, an even reduced system composed of $N_2 = N_{bw} + k + N_{fw}$. Note that the factorisation (3.23) can be performed exactly, hence leading to optimal, lossless compression, in which case $k \leq \min(N_{bw}, N_{fw})$ is the rank of the mixing matrix, but that it can also serve as a basis for a further compression, approximate this time, by keeping only dominant dimensions, i.e. associated to significant singular values.

This concludes the algorithm description, and we now report a summary pseudo-code for the entire procedure. In the following sections, we analyse the performance of the algorithm on different classes of networks.

Algorithm 3: Ergodic Set Compression Algorithm

Input : Graph $\mathcal{G}(V, E)$
Output: Compressed graph $\mathcal{G}'$

```
1  Function ESCA(G)
2      SCC ← FindSCC(G);
3      ES ← ()
4      for X ∈ SCC do
5          if IsInput(X) then
6              G ← InputCompress(X)
7              ES ← ES ∪ X
8          else if IsOutput(X) then
9              G ← InputCompress(X)
10             ES ← ES ∪ X
11     CORE ← V \ ∪_{X∈ES} X
12     CORE ← SVD(CORE)
13     G' ← CORE ∪_{X∈ES} X
```

The result of the algorithm is a multi-step compression, which is low to moderate for the first step (the coarse-graining of the ergodic sets) and a second, more aggressive core compression during the second step.

In order to assess the algorithm compression power, we will it to a number of different networks, both synthetic random graphs (see section 3.4) and real-world networks (see section 3.5) taken from various areas of applications. We perform the two compression steps separately so that we can estimate the impact of each part of the algorithm. The main metric we use to score the algorithm is the two *compression factors* C_1 and C_2, which represent the relative reduction in the number of nodes in the network after the first and second steps of the algorithm, respectively. More

explicitly, if N is the number of nodes in the original network, N_1 and N_2 are the number of nodes in the network after the first and second coarse-graining step, as defined above, and the two compression factors are defined as follows:

$$C_1 = 1 - \frac{N_1}{N} \qquad C_2 = 1 - \frac{N_2}{N}. \tag{3.24}$$

The compression factors are thus $0 \leq C_1 \leq C_2 \leq 1$ by definition, and higher compression factors correspond to a higher coarse-graining power of the algorithm.

3.4 Generalised ergodic sets in random networks

In this section we aim to give a theoretical estimate of the number of ergodic sets in the most used synthetic graph models. We perform the calculations for the Erdős-Rényi model, which is the simplest synthetic model of random graphs, and then extend our observations to other networks. Let $\mathcal{G} = \mathcal{G}_{ER}(N, p)$ be our directed Erdős-Rényi graph, where we extract each directed edge independently in both directions.

We further compare this with the simulation of a large number of samples from the Erdős-Rényi ensemble and verify the agreement in some simple but significant limits. The simulation generates N_{iter} graphs for a given combination of N, p and counts the number of ergodic sets that are present; this is then averaged over the N_{iter} instances. The results of such simulation are shown in figures 3.10 to 3.12.

The following fitting function have been used:

- for the number of ergodic sets [ES], no fit was performed, as it's just an exponential decay with different plateaus at 1/N
- for [NES], the total number of vertices in ANY ergodic set:

$$f_{NES}(p, [\alpha_1, \beta, \gamma, \delta]) = e^{-\alpha_1 p} + \left[\frac{1}{(1 + e^{-\beta(x-\gamma)})}\right]^{\delta} \tag{3.25}$$

- for [GSCC], the size of the giant strongly connected component [GSCC]:

$$f_{GSCC}(p, [\alpha_2, \beta, \gamma, \delta]) = \mathbb{1}_{p<\alpha_2} \frac{1}{N} + \mathbb{1}_{p>\alpha_2} \left[\frac{1}{(1 + e^{-\beta\,(x-\gamma)})} \right]^{\delta} \tag{3.26}$$

The exponential decay factor α_1 stays constant for all the sizes considered:

$$N = [100, 200, 300, 400, 500, 600, 700, 800, 900, 1000]$$

and the reported value is

$$\alpha_{ES} = 0.998 \pm 0.004 \tag{3.27}$$

The parameter α_2 depends on the granularity of the dataset and thus is not reported, as the probability step is not fine enough to provide an accurate estimate.

The exponential prefactor β has no clear distribution, but the results are consistent between the two methods. The following figure shows the agreement. The exponential prefactor β is slightly dependent on the method, and calling β_1 the estimate from the fitting of equation (3.25) and β_2 the estimate from the fitting of equation (3.26)

$$\beta_{NES} = 1.056 \pm 0.017 \tag{3.28}$$

$$\beta_{GSCC} = 1.026 \pm 0.024 \tag{3.29}$$

The two values are compatible and differs by $1.0200\,\sigma$

The previous two slides showed that the fitting of the GSCC function is much less accurate than the NES function. This is due to the fact that the continuous parameter α_2 controls a measure that is only discretely represented in the dataset.

As we can see, the normalised number of ergodic sets decreases exponentially for increasing values of p, independently of N. The minimum is reached when the increasing number of edges creates a giant strongly connected component, at which

point only one ergodic set is possible: the plateau value is indeed $1/N$.

By comparing figures 3.12 and 3.13 we can also note that the decay is faster in an SBM model than an ER model, when the parameters p for the two models is constant. We recall that for the Erdős-Rényi and SBM model such parameter is chosen as follows:

$$p = p_{ER} \qquad p = p_{SBM} = \frac{p_{in} + p_{out}}{2} \tag{3.30}$$

We can explain this by the following reasoning. Start from the limit case $p = 0$, where there are no edges in the network. As soon as the number of edges increases to 1, an output and input of size 1 are created with certainty. This is because the single edge turns the origin node into an input and the destination node as an output. Since no other edges are in the graph, all other points are isolated, and no other inputs or outputs are possible. When a second edge is inserted, with high probability is will connect two nodes that are distinct from the two considered above. This will in turn create another input and output.

In the SBM model, the parameters p_{in} and p_{out} are the one driving the presence of edges, but it's their average p that is driving the model. For values of p at which $p_{in} = p + \Delta p/2$ is large enough to produce a non-zero number of edges in expectation, while p_{out} generates no edge on average, the creation of ergodic sets is governed by the largest probability among all locally defined probabilities of the network.

Thus, in the low probability regime, it is the higher p_{in} that drives the creation of ergodic sets inside the planted communities and causing the faster decay in figure 3.13.

3.5 ESCA compression of real-world networks

We have so far investigated the mathematical properties of ergodic set in synthetic networks, and we have found the analytical expression of the average number of

ergodic sets in an Erdős-Rényi random graph, as well as for the average total number of nodes that are included in an ergodic set.

The practical interest of ergodic set lies in information compression. In fields such as Networked Control System (NCS) [61], people have used graphs to regulate the behaviour of other devices or systems [62]. While the understanding of the whole control system is important in the design phase, the end user is often only interested in the output of the network (the output sets), conditional upon a certain set of data that is fed in the network inputs (the input sets).

We can express the data compression effect of the ESCA approach more clearly in terms of the network adjacency matrix. If we are disregard the specific paths information takes on the networks, and we are only interested in how the output reacts to different stimulation of the network inputs, we can simplify the network adjacency matrix $A \in \mathbb{R}^{n\times n}$, in which n is the total number of nodes, with the reduced matrix $B \in \mathbb{R}^{n_{in}\times n_{out}}$. Here, $n_{in} = |\mathcal{I}|$ is the total number of inputs of the network, and $n_{out} = |\mathcal{O}|$ the total number of outputs. Each element of the matrix B_{ij} represents the probability that a random walker X will finish its trajectory in the j-th output, conditional upon having started in the i-th input.

$$B_{ij} = \mathbb{P}\left(\lim_{n\to+\infty} X_n \in O_j | X_0 \in I_i\right) \tag{3.31}$$

The test bench for this dimensional reduction is a synthetic network generated by a Stochastic Block Model (SMB)

$$\mathcal{G} = SBM(40, 0.3, 0.7, 0.3, 0.1)$$

that is, a SBM model with 40 vertices, two communities of relative size $[0.3, 0.7]$, and a node connection probability that is community independent (it only depends on whether the pair of nodes is in the same community or in two different ones).

The probability of connection within the same cluster is $p_{in} = 0.3$, while the same probability for pair of vertices in different communities is $P_{out} = 0.01$. Thus, the block probability matrix for the complete model is $A \in \mathbb{R}^{40\times 40}$, whose elements are defined as follows:

$$A = \begin{bmatrix} 0.3 & \cdots & 0.3 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0.3 & \cdots & 0.3 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 0.1 & \cdots & 0.1 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0.1 & \cdots & 0.1 \end{bmatrix} \tag{3.32}$$

The outputs O_1, O_2, O_3 are attached to random nodes of the SBM core, and are labelled as follows:

$$\mathcal{O} = \{O_1 = \{50, 51\}, O_2 = \{54, 55, 56\}, O_3 = \{58, 59, 60\}\} \tag{3.33}$$

Meanwhile, the Inputs I_1, I_2, I_3, I_4 follows the same linking protocol and are labelled:

$$\mathcal{I} = \{I_1 = \{40, 41\}, I_2 = \{42\}, I_3 = \{43, 44\}, I_4 = \{49\}\} \tag{3.34}$$

thus bringing the adjacency matrix up to $A \in \mathbb{R}^{60\times 60}$.

The matrix B we defined above simplifies the core SBM and contains the probability of reaching any well from any given source. In the graph under consideration, we have that $Q(\mathcal{G}) \in \mathbb{R}^{4\times 3}$, and the simulation estimate for those probabilities are:

$$B = \begin{bmatrix} 0.702 & 0.033 & 0.265 \\ 0.606 & 0.087 & 0.307 \\ 0.704 & 0.022 & 0.274 \\ 0.654 & 0.07 & 0.276 \end{bmatrix} \tag{3.35}$$

Ergodic sets are also interesting for a more theoretical perspective. We expect some classes of real-world networks to exhibit a larger number of ergodic sets than others. For example, food chain networks in ecology show a particularly high number of both inputs (the herbivores) and outputs (the apex predators) [63], while we expect an efficient transportation network to have very few of both [64] (in commercial flight routes networks, for example, each input-output pair requires an aircraft to perform an empty relocation flight, that brings no revenue to the carrier). In other words, we expect to be able to distinguish the nature of a given network through a measure of the prevalence of ergodic sets in the network itself. In this sense, we can regard the number of ergodic sets as a measure of how far a directed network is from being strongly connected.

We have run simulations on a number of real-world networks taken from the ICON Colorado Network database and plotted the number of vertices that belongs to an ergodic set versus the total size of the network, divided by input and output. The results are shown in figure 3.15. For the sake of generality, we have chosen a mix of networks from biology, information technology, and transportation. In our selection, biological networks represent protein-protein interaction (PPI) networks, which have been extensively studied [65] and whose characteristics have been thoroughly explored. They represent large, densely connected networks with a high average degree. The transportation networks used are airlines networks, which are known to possess a few, highly connected hubs that facilitate the transfer of passengers [66]. Finally, for technological networks, we have chosen subsets of the World Wide Web, specifically the websites of the institutional websites of the 50 states of the United States of America. Web networks tend to possess fat-tailed degree distributions, well approximated by a power-law [67].

Together, these three classes of networks include a wide range of characteristics and degree distributions, which is the only information needed for the detection al-

gorithm, in addition to strongly connected components. Thus, having a large variety of degree distributions is important to test both the prevalence of ergodic sets and the efficiency of the detection algorithm.

We report the results in the following figures and tables. In figures figures 3.15 and 3.16 we report the total number of compressed nodes versus the total size of the network, for each network in the dataset. In other words, we plot the quantity $N - N_1$ and $N - N_2$ versus N, respectively. The data are further subdivided by the type of ergodic set that was responsible for the compression, so nodes compressed by the algorithm acting on an input and nodes compressed by the algorithm acting on an output are reported separately[2]. The total compression is therefore the sum of the two data points for each graph.

We also report the total compression results in figure 3.17 for both the total compression rate C_1 after the first step of the algorithm and for the final compression coefficient C_2, as defined at the end of Section 3.3.3.

	Min	Average	Max	Median
Web	0.22	0.40	0.63	0.37
Protein	0.32	0.88	0.99	0.97
Air traffic	0.05	0.09	0.14	0.09
Total	0.05	0.32	0.99	0.31

Table 3.1: Statistical metrics for the total compression rates C_2 for the graphs in the dataset, grouped by the of origin.

[2]For the first step, each node is compressed as a direct result of a single ergodic set. For the second step, each core node is assigned to input or output based on the total number of inputs and outputs it connects with.

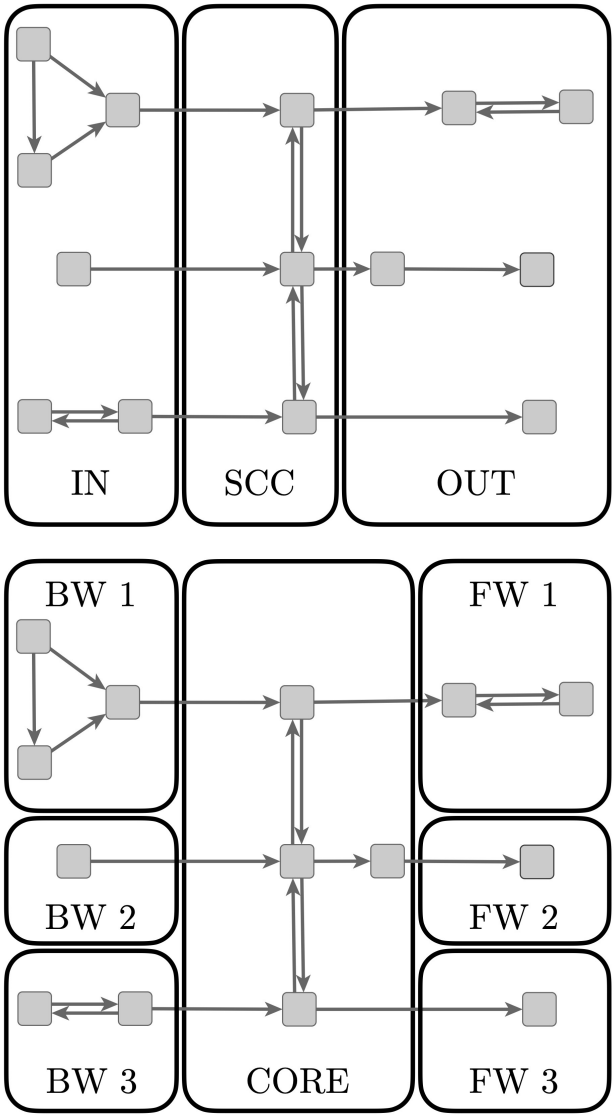


Figure 3.5: A representation of a generic network division. **Top:** the classical *bow-tie* structure [2]. **Bottom:** the division in ergodic sets. BW: backward ergodic set, FW: forward ergodic set.

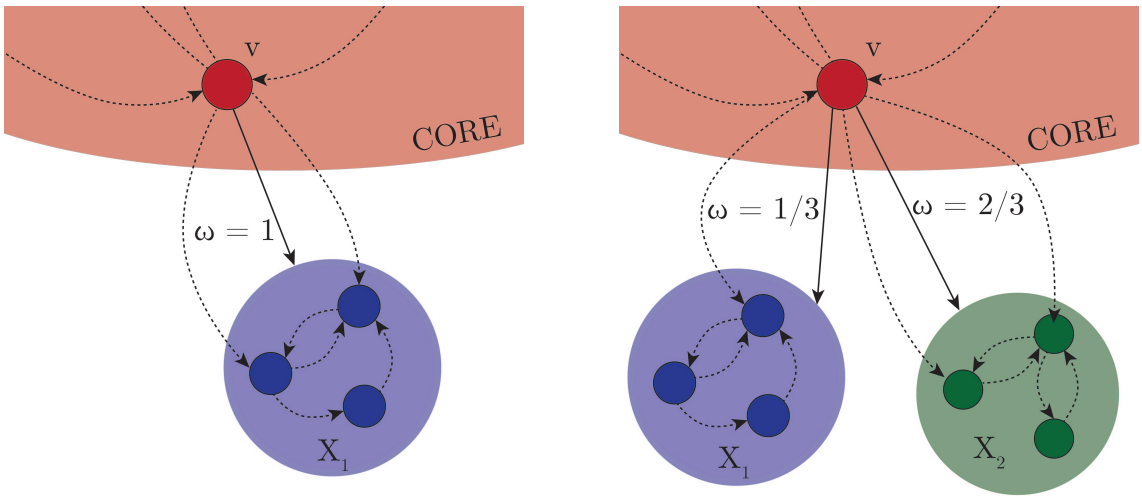


Figure 3.6: Graphical representation of the weight assignment rules for the output sets. In dark colours the old vertices of X, in pale colours the new coarse-grained vertex n_X. Old edges before the rewiring are shown as dotted, the new edges inserted by the *ESCA* algorithm are shown in solid lines.

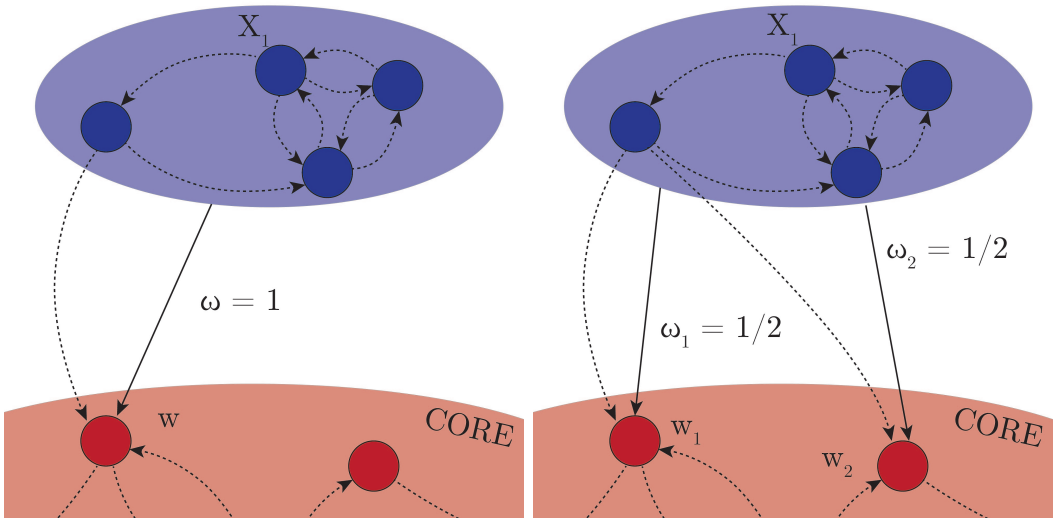


Figure 3.7: Graphical representation of the weight assignment rules for the input sets. In dark colours the old vertices of X, in pale colours the new coarse-grained vertex n_X. Old edges before the rewiring are shown as dotted, the new edges inserted by the *ESCA* algorithm are shown in solid lines.

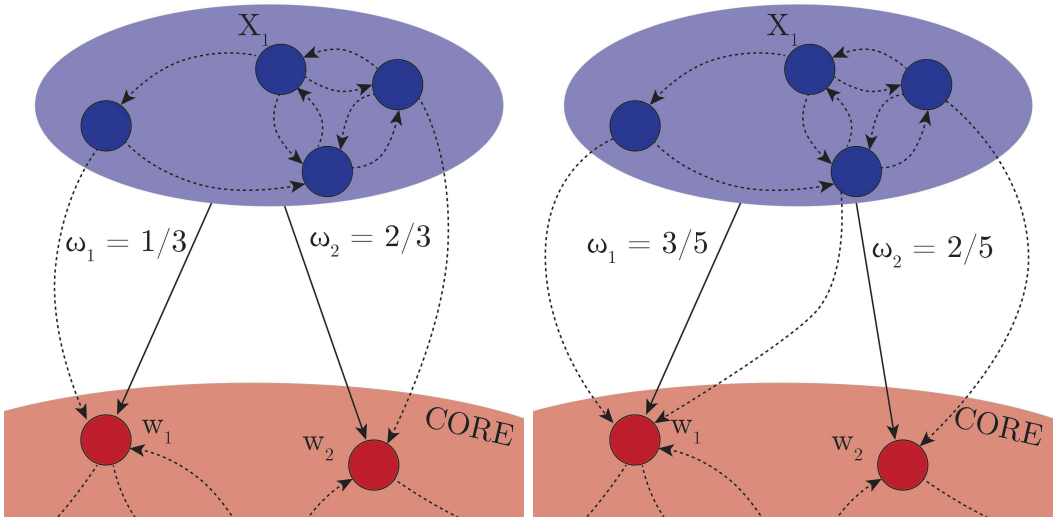


Figure 3.8: Graphical representation of the weight assignment rules for the input sets. In dark colours the old vertices of X, in pale colours the new coarse-grained vertex n_X. Old edges before the rewiring are shown as dotted, the new edges inserted by the *ESCA* algorithm are shown in solid lines.

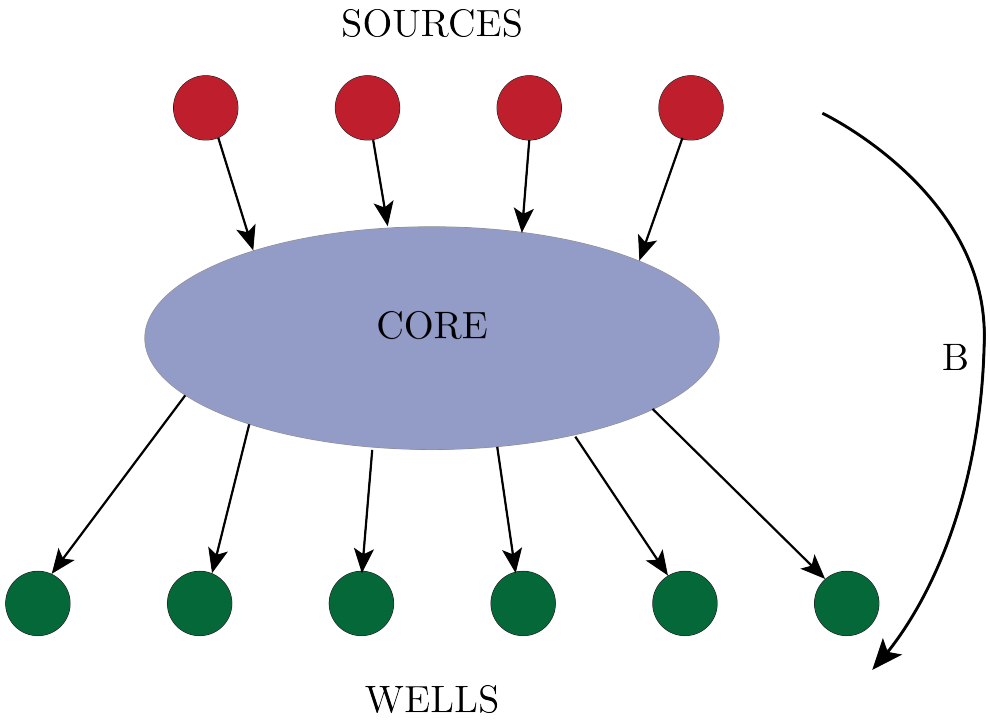


Figure 3.9: The resulting network at the end of the first compression step of the Ergodic Sets Compression Algorithm. Each (now proper) source is the result of the coarse-graining of an input set, while each (proper) is the result of the coarse-graining of an output set.

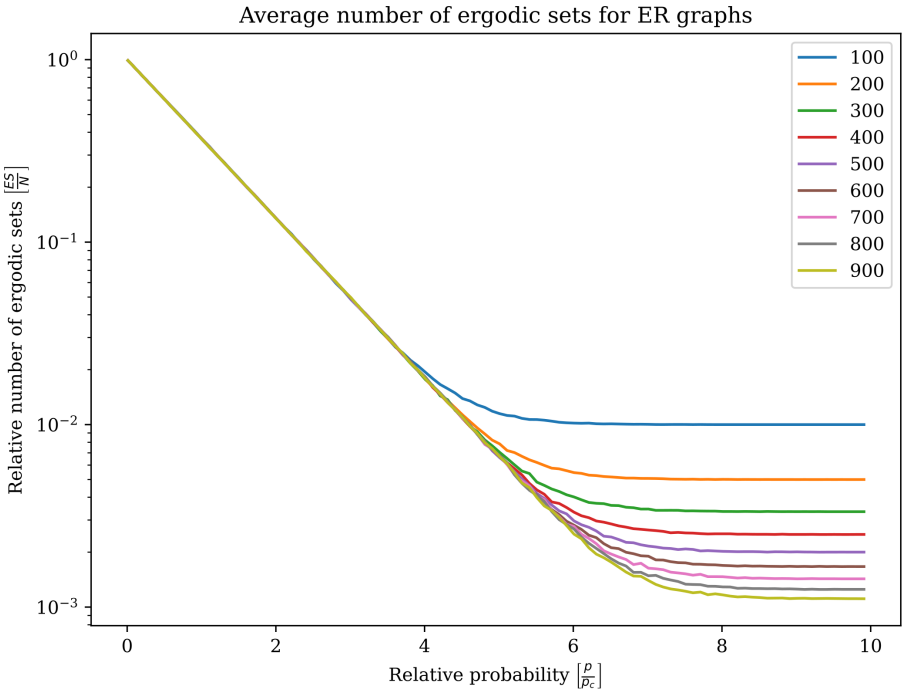


Figure 3.10: Average number of ergodic sets [inputs shown] for various choices of (N, p). Here, $N_{iter} = 1000$ for each sample.

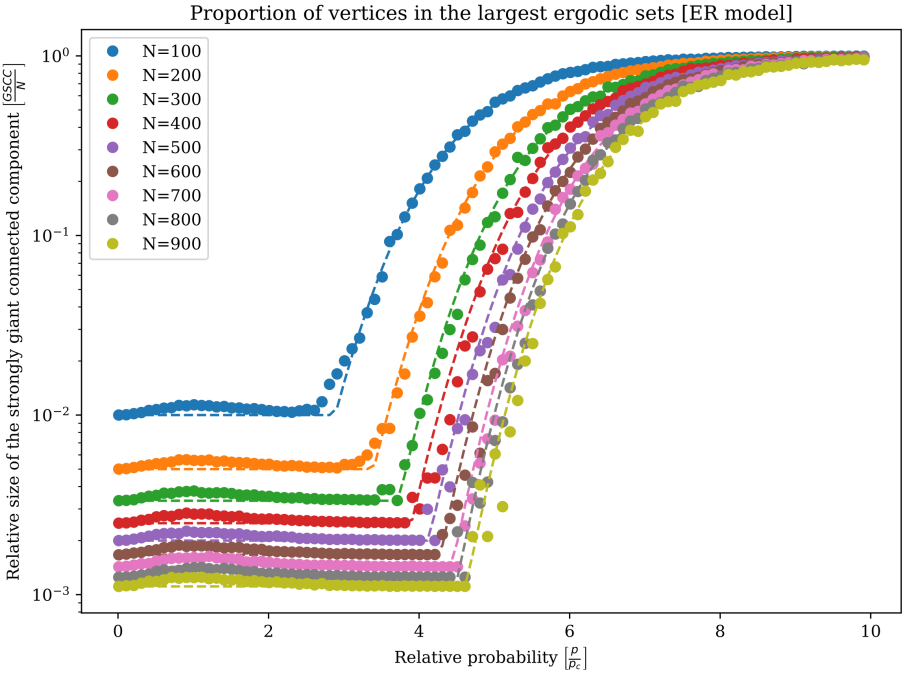


Figure 3.11: Number of nodes belonging to the largest ergodic set averaged over $N_{samples} = 1000$ instances of an Erdős-Rényi model $ER(N, p)$. The dependence on p is on the x axis, while different sizes N corresponds to different data series.

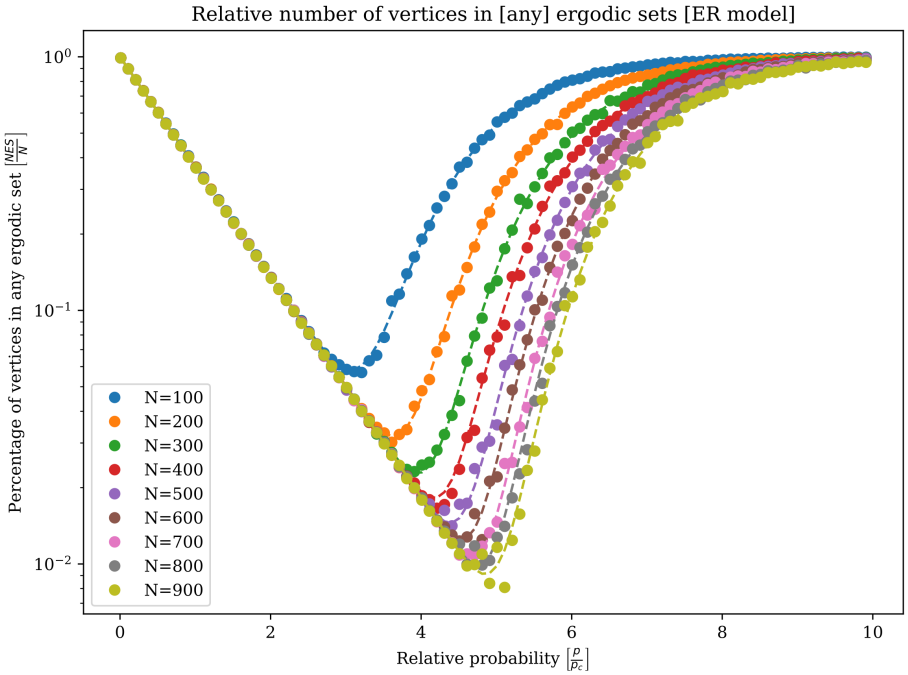


Figure 3.12: Number of nodes belonging to *any* ergodic set averaged over $N_{samples} = 1000$ instances of an Erdős-Rényi model $ER(N, p)$. The dependence on p is on the x axis, while different sizes N corresponds to different data series.

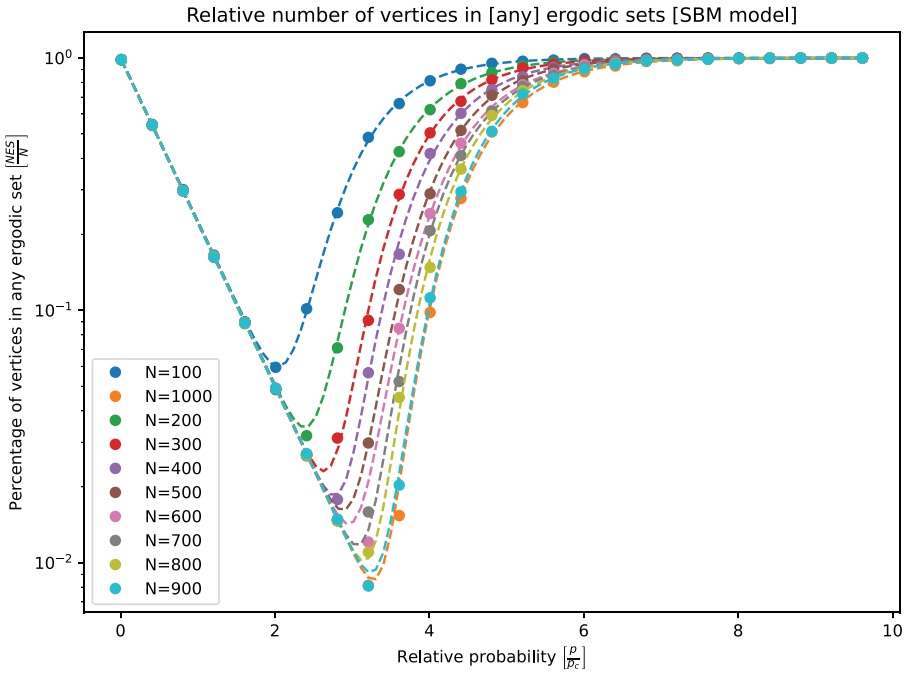


Figure 3.13: Number of nodes belonging to *any* ergodic set averaged over $N_{samples} = 1000$ instances of an SBM model $SBM(N, p_{in}, p_{out})$. The dependence on $p = \frac{p_{in}+p_{out}}{2}$ is on the x axis, while different sizes N corresponds to different data series.

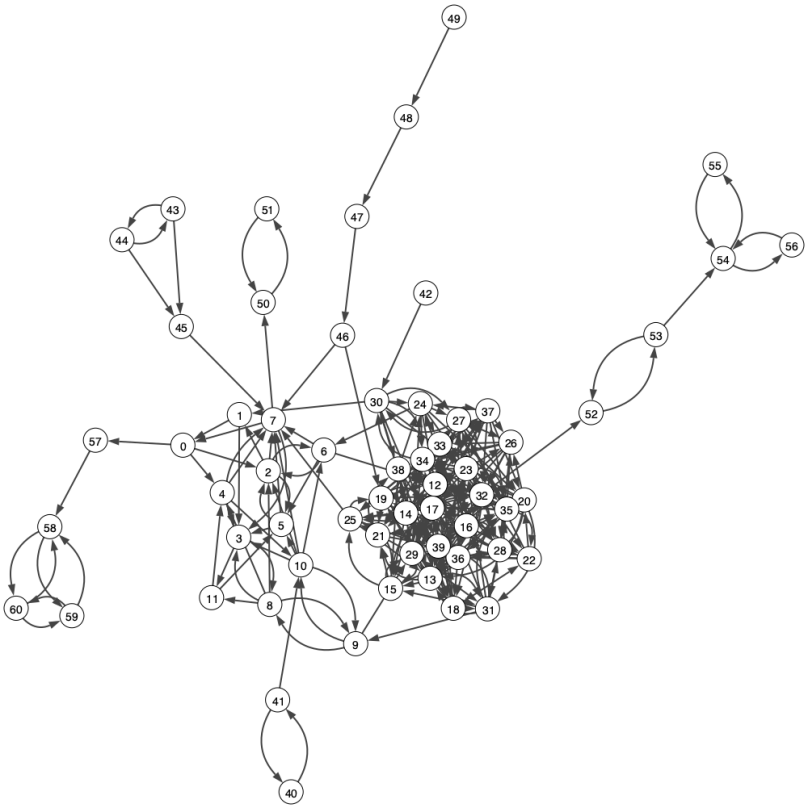


Figure 3.14: The original synthetic graph. The SBM core with its two communities is clearly visible, with vertices labelled from 0 to 39. The attachments of the inputs and outputs were randomly generated, including the possibility of multiple attachments (note vertex 46, attached to both vertex 7 and 19).

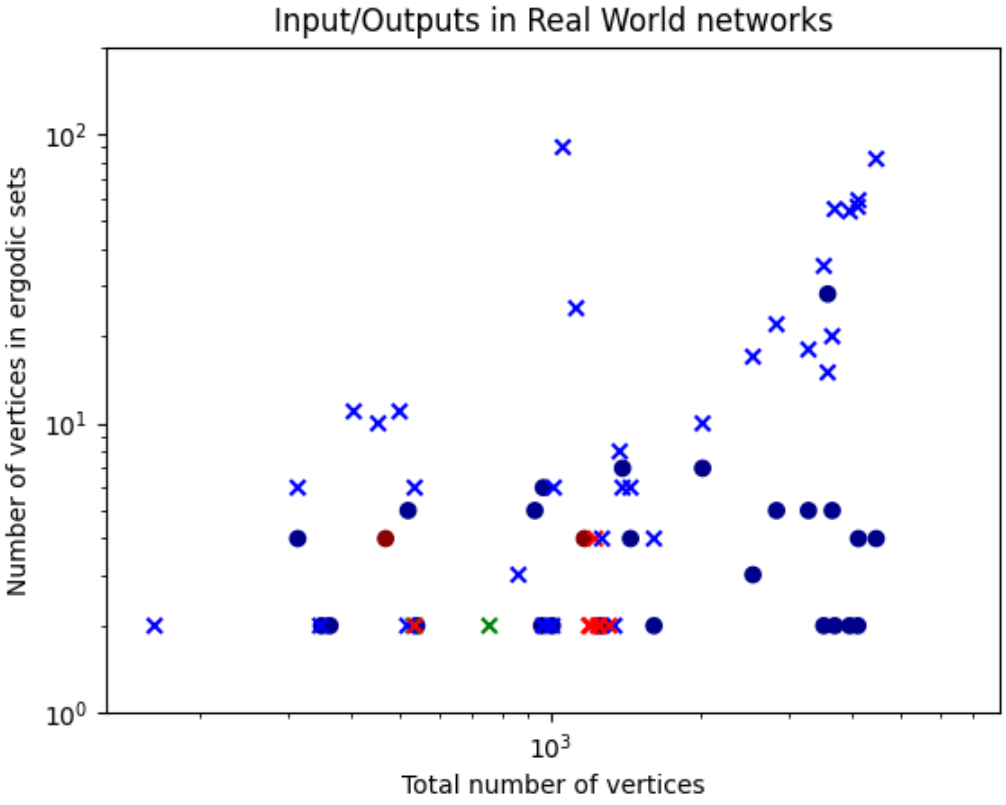


Figure 3.15: Number of inputs and outputs compression in the dataset of real-world networks, as function of the total number of vertices in the network. Data are arranged as input and output data and color-coded based on the application of origin. Axes in log-log scale.

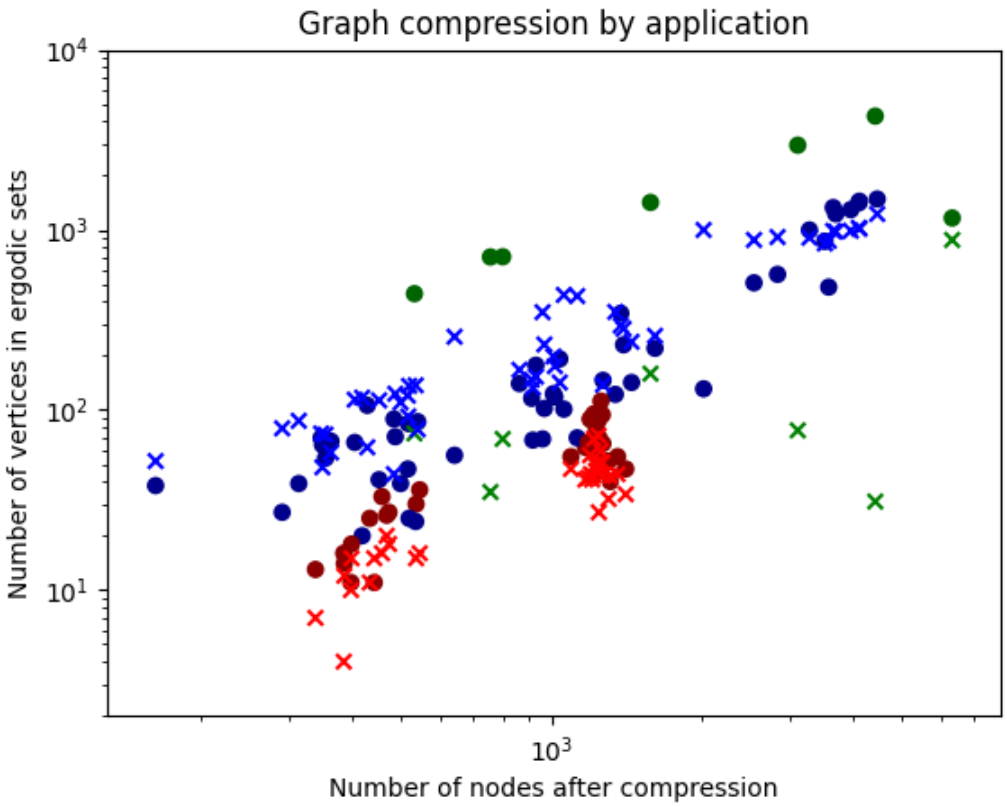


Figure 3.16: Total number of nodes compressed by the ESCA algorithm versus the original network size. Data are arranged as input and output data and color-coded based on the application of origin. Axes in log-log scale.

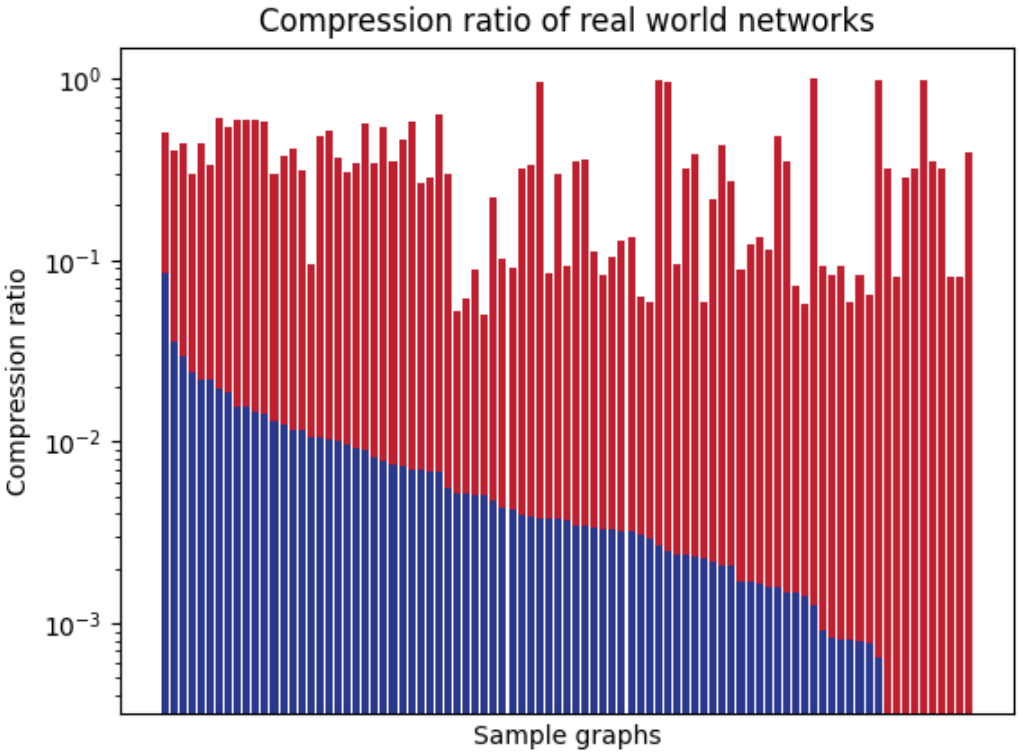


Figure 3.17: Compression rates for graphs in the dataset, showing both the first ESCA step compression factor C_1 and the total compression rate C_2.

The results in the table show results that are in accordance with what we might expect from the network based on their origin. The lack of ergodic sets in airline networks results into a small compressibility, while the high number of inputs in biological networks and the presence of entropy in biological processes makes them a great application for high compressibility. Web networks, finally, are the most balanced of the three and roughly a third of the web pages can be compressed. Note that the dataset is compiled from US governmental websites and might not be generalizable to commercial websites.

There are other methods, mostly community-based, that are able to coarse-grain the network to a much higher degree. Some can even generate a hierarchy of progressively more coarse-grained networks. This is the case, for example, of the Louvain [28] and the Leiden [68] algorithm. However, the power of this method sits in the dynamical properties of the resulting networks. In other coarse-graining algorithms, the dynamics after compression is completely different, as none of them takes into account the resulting dynamics when performing the network compression.

3.6 Core-periphery analysis

The second step of the algorithm substitute the graph core into set of disconnected nodes. While the long-time asymptotic flows are conserved by construction, it is natural to ask how much information is destroyed in the process. In this section, we try to quantify this by measuring the structure of the network core before rewiring. To do so, we study the dynamical distance of the core before the second ESCA step is performed with respect to a random graph.

We measure the mixing power of the network core using the symmetrised version

of the B matrix: $\mathcal{B} = B^T B$. Then, we study the spectral gap of its Laplacian:

$$L\left(\mathcal{B}\right)_{ij} = \sum_k B_{ik}\delta_{ij} - Bij \tag{3.36}$$

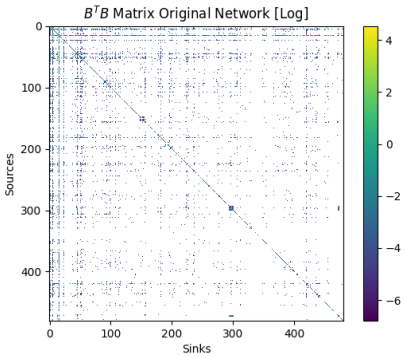
We compare its spectrum $\sigma(L)$ to the spectrum of a graph in which the core has been rewired using a directed configuration model, which preserve the core degree distribution by construction. The rewiring has only affected the edges in the CORE part of the network (see figure 3.5), and the rewiring has been done using a directed configuration model in which self-loops were not allowed. Edges crossing a region boundary in figure 3.5 were not rewired. Our findings indicate differences that are well over the threshold of statistical significance.

In figure 3.18 the $\mathcal{B}$ results are shown for three sample networks – and their rewired counterparts – representing the websites of the US states of California, Oregon, and Utah.

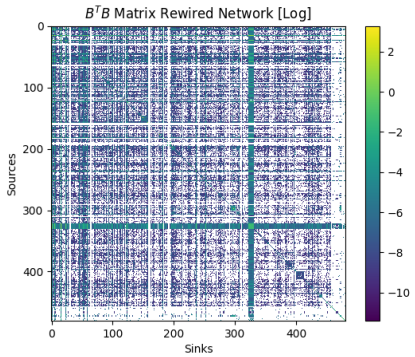
Rewiring the core has invariably determined an increase in the mixing capability of the core, showing that the network cores are different from their rewired counterparts to a statistically significant degree. The average number of zero eigenvalues of $L\left(\mathcal{B}\right)$ has decreased from 100.0 (before rewiring) to 13.7 (after rewiring), with standard deviation 39.1 and 6.7 respectively. This is a statistically significant reduction: using a t-distribution with 51 degrees of freedom and the t-statistic of 15.24, we find a p-value equal to 0, up to the limits of numerical precision.

This result reinforces the idea that the second step of the compression algorithm is well posed, as there is residual structure in the specific connection of the core to compress after the first ESCA step is performed. To further corroborate this result, we study the correlation between the decrease in the number of zeroes in the spectrum of the graph Laplacian before and after the core rewiring and the compression factor C_2. We find a correlation coefficient $\gamma = -0.75$, indicating strong anti-correlation

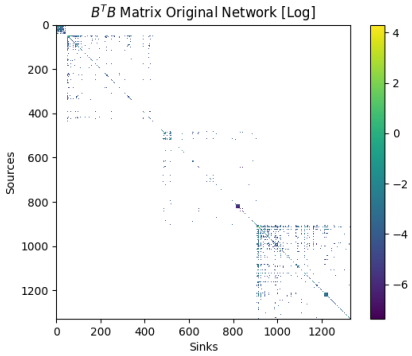
between the two quantities. In other words, graphs whose core is further away from a random graph are more difficult to compress.



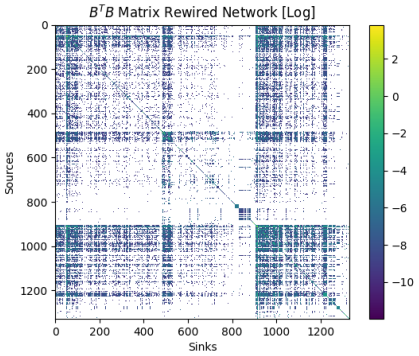
(a) B^TB (original network)



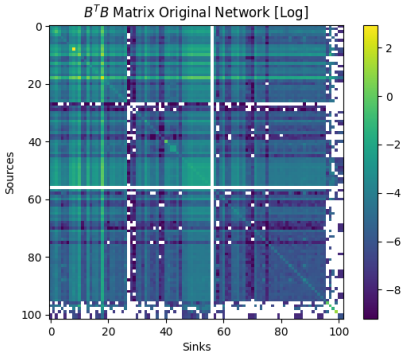
(b) B^TB (rewired network)



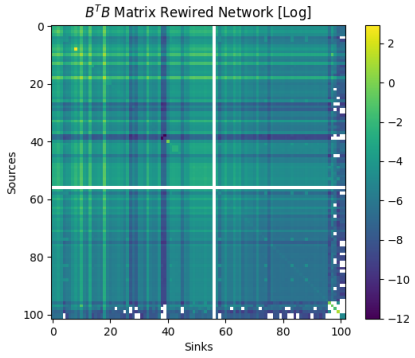
(c) B^TB (original network)



(d) B^TB (rewired network)



(e) B^TB (original network)



(f) B^TB (rewired network)

Figure 3.18: Plots of the symmetrised input-output connection matrix $\mathcal{B} = B^TB$. To the left the plots for the actual networks, to the right the plots for the same networks with rewired cores. By row, from top to bottom, the networks represent the websites of the states of California, Oregon, and Utah.

3.7 Discussion

In this project, we have introduced the Ergodic Sets Compression Algorithm, a two-steps coarse-graining method for directed networks that preserves the random walk dynamics. We started by introducing the concept of generalised ergodic sets, a generalization of sinks and sources in directed networks. We used such structures to perform a first compression step that reduces the complexity of the original network into single nodes structures whose hitting events factorise the dynamics. We then used such dynamical factorization to design a second compression step based on the singular values decomposition of the matrix representing the long-time limit of the probability distribution.

There are other methods, mostly community-based, that are able to coarse-grain the network and are faster and more efficient than our approach. Some can even generate a hierarchy of progressively more coarse-grained networks. This is the case, for example, of the Louvain [28] and the Leiden [68] algorithm. However, the power of the method we propose in this chapter lies in the dynamical properties of the resulting networks. In other coarse-graining algorithms, there is no guarantee for the dynamics after the coarse-graining to be equal, or even similar, to the one of the original networks. In other words, the other methods don't take into account the resulting dynamics when performing the network compression. Our method, on the other hand, guarantees the coarse-grained result to have the same dynamical properties of the original system. This, combined with its lightness and quick runtime, makes it an excellent first step in any dynamic-based investigation on directed networks.

Finally, we have applied our algorithm to a wide class of synthetic and real-world networks. We have found good agreement with the percolation threshold in synthetic networks and excellent compression power for real networks from different domains.

Chapter 4

Return time distributions

The primary contribution of this section is the development of a set of formulas to efficiently compute first return time distributions on several classes of networks. We developed these formulas using a message passing approach and the resulting expression only depends on the local structure around the node we are considering (specifically, the node degree), as well as the global structure of the graph, through the total number of edges. We justify the derivation of the formulas as a combination of a local message-passing approach for estimating the behaviour at short times and a global mean-field approximation accounting for the tail behaviour at long times. We test this approximation on several classes of networks, including both locally tree-like graphs and networks with short cycles, and discuss how the results of this approximation may impact community detection methods that rely on the statistics of finite-length random walks.

More specifically, in section 4.1 we introduce the motivations for the research and the ideas that contributed to the development of the project, section 4.2 contains the explicit derivation of the formulas of the first return times approximations, and section 4.3 shows the numerical results on several classes of graphs. Finally, section 4.4 discusses the implications of our findings on community detection algorithm based on

random walks analysis and section 4.5 contains a broader discussion of our results in the wider context of the subject.

4.1 Random walks and Kac's lemma

The genesis of this chapter comes from a simple yet powerful observation. On the one hand, network science has created a number of community detection algorithms that rely on random walks. Probably the most famous random-walks-based community detection algorithm is InfoMap [36, 34], which rewrites random walks in an ingenious way using minimum description length (MDL) statistics to describe them in terms of codes. The important point for our work is that InfoMap, as well as all such methods rely on the same, identical idea: random walks look fundamentally different on networks with and without communities. This is a reasonable assumption, yet – as we will show in this chapter – fundamentally misleading.

The intuitive reasoning goes that because of the assortative nature of communities (i.e., the fact that nodes are more tightly connected inside the same community than across different communities), random walks will exhibit modular properties. For example, nodes outside the starting community will have longer hitting times, while nodes in other communities will see shorter hitting times. This multi-scale nature extends to path lengths (shorter inside the same community, longer across communities), and mixing times(once again, shorter inside the same community, longer across communities).

However, random walks also exhibit a few universal properties, most notably the local dependence of the mean first return time of the starting node. This is a special case of a much broader result, called Kac's Lemma, which applies to any dynamical process. Given a space X, a discrete-time, measure-preserving transformation $f : (X, \mu) \to (X, \mu)$, and a subset $A \subseteq X$ of strictly positive measure, $\mu(A) > 0$, we can

define the first return time as follows:

$$Y_A(x) = \inf\left\{k \geq 1 \text{ s.t. } f^k(x) \in A\right\} \tag{4.1}$$

And Kac's lemma gives us a closed formula to calculate the expected value of first return time, proving that:

$$\int_A Y_A \mathrm{d}\mu_A = 1 \tag{4.2}$$

hence, when taking the average $\langle Y_a \rangle$ we obtain the simple expression:

$$\langle Y_a \rangle = \frac{1}{\int_A \mathrm{d}\mu_A} = \frac{1}{\mu(A)} \tag{4.3}$$

For the case of random walks on networks (resp. directed networks) satisfy the ergodic requirement as long as the graph is connected (resp. strongly connected). Kac's lemma states that the mean return time is proportional to the inverse of the measure of the set (in the dynamical sense). Because at each time step the graph random walk crosses each edge of the network with the same probability, each node has an associated volume equal to its degree (or in-degree, if in a directed network). Hence, by applying Kac's lemma, we can write for each node $v \in V$:

$$Y_v = \frac{1}{d_{in}(v)/|E|} = \frac{|E|}{d_{in}(v)} \tag{4.4}$$

in which the number of edges E is counted directional, i.e. we double count the edges in an undirected network (once per direction).

We sketch a simple proof of Kac's lemma in the special case of random walks on undirected networks. In this case, the subset we normally consider is a single node, so we lighten the notation and indicate Y_i the average first return time of the i-th node. At every time-step t, the random walk has some probability to return to its starting point; we call this probability $Y_i(t)$. Clearly, $Y_i(t) > 0$ for all t; furthermore, because

we only consider finite networks, all random walks are transient, so all random walks return to the starting node in finite time, hence we have the correct normalization condition $\sum_{i=y}^{+\infty} Y_i(t) = 1$. Now the average first return time is just the average over time of the first return time probability $Y_i(t)$, namely:

$$Y_i = \sum_{t=1}^{+\infty} t\, Y_i(t) \tag{4.5}$$

Let us call $\mu(i \rightsquigarrow j)$ the average time for the random walk to start in node i and reach node j. Note that this notation leads to the simple equality $\mu(i \rightsquigarrow i) = Y_i$. We now want to inspect the structure of $\mu(i \rightsquigarrow j)$. Starting from node j, the random walk will move to some neighbour of j – say k – with a probability governed by the row-normalized adjacency matrix W. If $j = i$, then we have reached the target, if not we are now in a different node k, from which the average time to reach i is simply $\mu(k \rightsquigarrow i)$. Because of the linearity of the average, we can write:

$$\mu(j \rightsquigarrow i) = 1 + \sum_{k \in \mathcal{N}(i)} W_{jk}\, \mu(k \rightsquigarrow i) \tag{4.6}$$

now the sum is taken over the neighbourhood of i, but the transition matrix is zero for all the nodes that are not in the neighbourhood. To extend the sum to all nodes in the graph we therefore only need to subtract the term for which $k = j$.

$$\mu(j \rightsquigarrow i) = 1 + \sum_{k} W_{jk}\, \mu(k \rightsquigarrow i) - W_{ji}\, \mu(i \rightsquigarrow i) \tag{4.7}$$

Now we can multiply both sides by d_j and sum over j, obtaining:

$$\sum_{j} d_j\, \mu(j \rightsquigarrow i) = \sum_{j} d_j \left(1 + \sum_{k} W_{jk}\, \mu(k \rightsquigarrow i) - W_{ji}\, \mu(i \rightsquigarrow i) \right) \tag{4.8}$$

using the fact that $\sum_j d_j W_{ji} = d_i$ and the handshaking lemma, we get Kac's lemma:

$$\sum_j d_j \, \mu(j \rightsquigarrow i) = 2|E| + \sum_j d_j \left(\sum_k W_{jk} \, \mu(k \rightsquigarrow i) - W_{ji} \, \mu(i \rightsquigarrow i) \right) = \tag{4.9}$$

$$= 2|E| + \sum_j d_j \sum_k W_{jk} \, \mu(k \rightsquigarrow i) - \sum_j d_j W_{ji} \, \mu(i \rightsquigarrow i) = \tag{4.10}$$

$$= 2|E| + \sum_k \sum_j d_j W_{jk} \, \mu(k \rightsquigarrow i) - d_i Y_i = \tag{4.11}$$

$$= 2|E| + \sum_k d_k \, \mu(k \rightsquigarrow i) - d_i Y_i \tag{4.12}$$

in which we have swapped sums noting that all sums appearing above are finite and thus convergent. We can now simplify and rearrange the expression to obtain Kac's lemma in the form we are familiar with:

$$Y_i = \frac{2|E|}{d_i} \tag{4.13}$$

Kac's lemma is a surprising result, both for its simplicity and for the fact that it ties one key property of random walks with only a single, local property of the network, the degree distribution, and a global property, the total number of edges. In other words, Kac's lemma seems to suggest that random walks on networks possess a universal property that is unaffected by the presence of communities.

Even more explicitly, the first return time of random walk starting in any node will not depend on the community structure. How would this inform us on the similarity between the two random walks, the former on an Erdős-Renyi graph and the latter on a highly modular SBM network?

Knowing the full first return time distribution for each node in a graph of course provide a lot of information on the random walk process that has generated it. But is this information equivalent to the full knowledge of the stochastic matrix governing the random walk? While there are several intuitive reasons, numerical observation

and empirical results suggesting this is indeed the case, ultimately this is an open question. The extent to which the knowledge of the first return time distribution informs the underlying random walk is not fully understood, and a further discussion on this relation is presented in chapter 6.

4.2 Approximating first return time distributions

4.2.1 Heterogeneous mean-field approximation

The simplest case we consider for writing our approximation is a heterogeneous mean-field approximation in which the walker can move, at each step, to any node i (including the one it is currently located on, "skipping a turn") with a time-independent probability that is equal to the stationary distribution $\pi_i = \frac{d_i}{2|E|}$. This is equivalent to the PageRank random surfer process (see the introduction to chapter 3) in which the teleportation factor is set to $\alpha = 1$.

In the heterogeneous mean-field approximation, the random walk converges to the stationary distribution in one single step, and the stochastic process is equivalent to a random walk on a weighted network with self-loops, in which the elements of the adjacency matrix only depend on the node degree:

$$\left(A^{MF}\right)_{ij} = \frac{d_i\, d_j}{2|E|} \tag{4.14}$$

The approximated return probability $\tilde{X}_i(t)$ is then given by:

$$\widetilde{X}_i(t) = \begin{cases} 1 & \text{for } t = 0 \\ \frac{d_i}{2|E|} & \text{for } t \geq 1, \end{cases} \tag{4.15}$$

From here, we can follow the procedure sketched in section 2.2.2 to calculate the approximated first return time distribution $\widetilde{Y}_i(t)$. The event of the random walker

to return *for the first time* after t steps, is equivalent to the request of it not having returned for the previous $t-1$ steps, and then returning at in one step exactly. Because in the heterogeneous mean-field approximation the walker returns at each time step with equal probability $\pi_i = d_i/2|E|$, we have:

$$\widetilde{Y}_i(t) = \mathbb{P}\left(X_0 = i, X_1, X_2, \ldots, X_{t-1} \neq i\right) \mathbb{P}\left(X_t = i \mid X_{t-1}\right) = \left(1-\pi_i\right)^{t-1} \pi_i \tag{4.16}$$

We can now calculate the generating function using its definition: using equation (2.18) we obtain:

$$\widetilde{F}_i(z) = \sum_{t=1}^{\infty} z^t \, \widetilde{Y}_i(t) = \sum_{t=1}^{\infty} z^t \left(1-\pi_i\right)^{t-1} \pi_i = \frac{z\pi_i}{1-z+z\pi_i}. \tag{4.17}$$

This approximation is a very crude one and will only provide valid estimates in the limit of long times, as it is derived under the assumption of thermalization of the random walk. However, it satisfies two important properties:

$$\sum_{t=1}^{\infty} \widetilde{Y}_i(t) = \widetilde{F}_i(1) = 1 \tag{4.18}$$

and

$$\sum_{t=1}^{\infty} t \, \widetilde{Y}_i(t) = \widetilde{F}_i'(1) = \frac{2m}{k_i} \tag{4.19}$$

both of which must be true on a finite network: the first one formalises the fact that all random walks are recurrent, hence the sum of all first return times probability for all times is a certain event, while the second one is just a restatement of Kac's lemma using the properties of the generating function.

4.2.2 Tree approximation

Now let's set aside the homogeneous mean-field approximation and work on the message passing approximation. The two will eventually be merged together to account for both the short- and long-time behaviour of the first return time distribution.

Let us start by assuming that the network has no cycles, that is, it is a tree. As we have seen in section 2.5.2, belief propagation is exact on trees, so we expect this approximation to be exact in this case. We introduce the quantity $Y_{i\leftarrow j}(t)$ as the first return time distribution to node i, avoiding node j. In other words, $Y_{i\leftarrow j}(t)$ is the probability that a random walk that starts at j returns to j for the first time after t steps and does not visit i in between.

We can relate the quantities by once again investigating the details of the random walk. Suppose that the random walk starts at node i; in its first step, it will move to one node in i's neighbourhood j, each with equal probability $1/d_i$. Next, the walk starts its second step from node j and can either return to i with probability $1/d_j$ or move to a third node $k \neq i$ adjacent to j. The probability of returning after exactly two steps is therefore the sum of all the two-steps walks reaching each neighbour j of i and immediately coming back, that is:

$$Y_i(2) = \sum_{j\in\mathcal{N}(i)} \frac{1}{d_i d_j}. \tag{4.20}$$

Now let us use the fact that the graph is a tree, and we will identify the Y_i quantity in this special case as $\widetilde{Y}_i$. In a tree graph, once the random walk has visited node $k \in \mathcal{N}(j)$, $k \neq i$, it can only go back to the starting node i by transiting through j. This is a straightforward consequence of trees not having any cycles, so there is only one path from i to k, the one going through node j.

This means that there must be some number $q \geq 0$ of excursions that start and end at j and never reach i. Let us indicate the probability distribution for the length

of these excursions as $\widetilde{Y}_{i\leftarrow j}(t)$. We can then write the probability that we return to node i for the first time after exactly t steps as the product of the first step from i to j, plus the length of the excursions $\widetilde{Y}_{i\leftarrow j}(t)$ plus the last step from j to i, all summed over all possible neighbours $j \in \mathcal{N}(i)$ and all possible q excursions. More formally:

$$\widetilde{Y}_i(t) = \frac{1}{d_i} \sum_{j \in N_i} \frac{1}{d_j} \sum_{q=0}^{\infty} \left[\sum_{t_1} \cdots \sum_{t_q} \left(\widetilde{Y}_{i\leftarrow j}(t_1) \widetilde{Y}_{i\leftarrow j}(t_2) \ldots\ldots \widetilde{Y}_{i\leftarrow j}(t_q)\, \delta\big(t-2, \textstyle\sum_s t_s\big) \right) \right].$$

in which $\frac{1}{d_i}$ represent the probability of the first step, $\frac{1}{d_j}$ the probability of the last step, and the sums over $t_1, \ldots, t_q$ all possible locations of the random walks for all possible q excursions, whose length is limited to $t-2$ (the total length minus the first and last steps) by the Dirac δ-function.

We now pass to generating function by multiplying Eq. (4.21) by z^t and summing over t, obtaining:

$$\widetilde{F}_i(z) = \sum_{t=0}^{\infty} z^t\, \widetilde{Y}_i(t) = \tag{4.21}$$

$$= \sum_{j \in N_i} \frac{z^2}{d_i d_j} \sum_{q=0}^{\infty} \left(\sum_{\tau=0}^{\infty} z^\tau\, \widetilde{Y}_{i\leftarrow j}(\tau) \right)^q = \tag{4.22}$$

$$= \sum_{j \in N_i} \frac{z^2}{d_i d_j} \sum_{q=0}^{\infty} \left(\widetilde{F}_{i\leftarrow j}(z) \right)^q = \tag{4.23}$$

$$= \sum_{j \in N_i} \frac{z^2}{d_i d_j} \left(\frac{1}{1 - \widetilde{F}_{i\leftarrow j}(z)} \right), \tag{4.24}$$

in which we have operated the substitution $\tau = t-2$ and $\widetilde{F}_{i\leftarrow j}(z) = \sum_{t=0}^{\infty} z^t\, \widetilde{Y}_{i\leftarrow j}(t)$ is the generating functions for $\widetilde{Y}_{i\leftarrow j}(t)$.

In a similar way we can compute $\widetilde{Y}_{i\leftarrow j}(t)$. Recall that $\widetilde{Y}_{i\leftarrow j}(t)$ is the probability distribution for the excursions that start and end at j and never reach i. So, these random walks must start at node i, first visit a neighbour of j that is not i, and

finally return to j at a later step. Applying these considerations to the same line of argument as above leads to:

$$\widetilde{F}_{i\leftarrow j}(z) = \sum_{k\in N_j\setminus i} \frac{z^2}{d_j d_k}\left(\frac{1}{1-\widetilde{F}_{j\leftarrow k}(z)}\right), \tag{4.25}$$

where the sum over $k \in N_j \setminus i$ is a sum over all neighbours of j except for i. Now we can insert equation (4.25) into equation (4.24) to calculate $\widetilde{F}_i(z)$, the generating functions for the first return times.

Equation (4.25) can be evaluated on any given network to make predictions. We may also be interested in the distribution of solutions. Let us assume a locally tree-like network with degree distribution $\rho(d)$ and mean degree $\langle d\rangle = \sum_d d\rho(d)$. Then the probability density for a randomly chosen message to take value F and lead to a node of degree d is $P_{\leftarrow}^{(z)}(F,d)$. For this probability distribution to be consistent with the message passing equation, it must solve the self-consistent equation:

$$P_{\leftarrow}^{(z)}(F,d) = \frac{d\rho_(d)}{\langle d\rangle}\int\prod_{j=1}^{k} \mathrm{d}P_{\leftarrow}^{(z)}(F_j,d_j)\delta\left(\sum_{j=1}^{k}\frac{z^2}{d_i d_j(1-F_j)} - F\right) \tag{4.26}$$

which can be solved numerically using the population dynamic algorithm [39], see section 2.5.2.

Applying equation (4.26) to a d-regular tree (a tree in which each non-leaf node has the same branching number $d-1$), we get the following simplifications. Both equation (4.24) and equation (4.25) are the same for all pairs $(i,j$, leading to equation (4.24) becoming:

$$\widetilde{F}(z) = \frac{z^2}{d}\left(\frac{1}{1-\widetilde{F}_{\leftarrow}(z)}\right) \tag{4.27}$$

and equation (4.25) simplifying as:

$$\widetilde{F}_{\leftarrow}(z) = \frac{z^2(d-1)}{d^2}\left(\frac{1}{1-\widetilde{F}_{\leftarrow}(z)}\right). \tag{4.28}$$

Yielding the solution:

$$\widetilde{F}(z) = \frac{2z^2}{d+\sqrt{d^2+4z^2-4dz^2}}, \tag{4.29}$$

which is in agreement with standard results by Huges for the random walk statistics on infinite Bethe lattices [69].

Looking at Eq. (4.29) we observe that $\widetilde{F}(1) < 1$ for $d > 2$. Because $\widetilde{F}(1)$ is the sum of the return probabilities at all times, we conclude that our approach predicts a transient random walk. This means that a random walk that starts at i may never return to i, or equivalently, i may only be visited a finite number of times. This is a rephrasing of the famous result by Polya [70], which states that random walks in more than two dimensions are transient. However, as we are considering finite networks, it is clearly wrong and a byproduct of our approximation.

4.2.3 Combined approximation

On a locally tree-like graph, the tree approximation satisfies some key properties of recurrence and transience in infinite networks but fails to account for the network finite size effects, most notably with respect to recurrence $\widetilde{F}_i(1) \neq 1$ and Kac's lemma $\widetilde{F}_i'(1) \neq 2|E|/d_i$.

The heterogeneous mean-field approximation we developed in section 4.2.1 satisfies both conditions. However, it fails to pick up any local structure or transient behaviour. So, the next step in this section will be to combine the two approximations in the hope of getting the best of both, thus yielding an approximation that is well behaved both in the short- and long-time regimes.

Starting with the solution $\widetilde{F}_i(z)$ to the tree approximation, from Eq. (4.24), let us

define the quantity:

$$h_i = \frac{2|E| - d_i \widetilde{F}_i'(1)}{d_i - d_i \widetilde{F}_i(1)} \tag{4.30}$$

and then use to interpolate the two previous solutions. We obtain:

$$F_i(z) = \widetilde{F}_i(z) + \frac{z - z\,\widetilde{F}_i(1)}{z + h_i - z h_i}. \tag{4.31}$$

This interpolation contains the tree approximation and a geometric distribution with mean h_i, with the former being dominant in the short time regime and the latter in the long-time regime. The idea behind this interpolation is the following: with probability $(1 - \tilde{F}_i(1))$ the tree approximation predicts a random walk escaping to infinity. We assume that, on a finite graph, this is equivalent to thermalise to the stationary distribution, as predicted by the homogeneous mean-filed approximation. Thus, for long times, this interpolation reduces to the tree approximation, as can be seen by checking the required properties:

$$F_i(1) = 1 \qquad \text{and} \qquad F_i'(1) = 2m/k_i \tag{4.32}$$

Thus, Eq. (4.31) defines a distribution for a recurrent process with the correct mean and a reasonable approximation at short times on locally tree-like graphs.

4.2.4 Approximation for networks with short cycles

We want to further generalise the approximation to networks with cycles. The fundamental parameter here is the length ℓ of the shortest cycle in the graph. If $\ell \gg 1$, then there are no short cycles in the graph and the graph is locally-tree like. Because the probability of choosing a specific path from i to i is exponentially suppressed with the number of steps, and assuming that the number of independent cycles increase less than exponentially with respect to their length, long cycles will yield a small

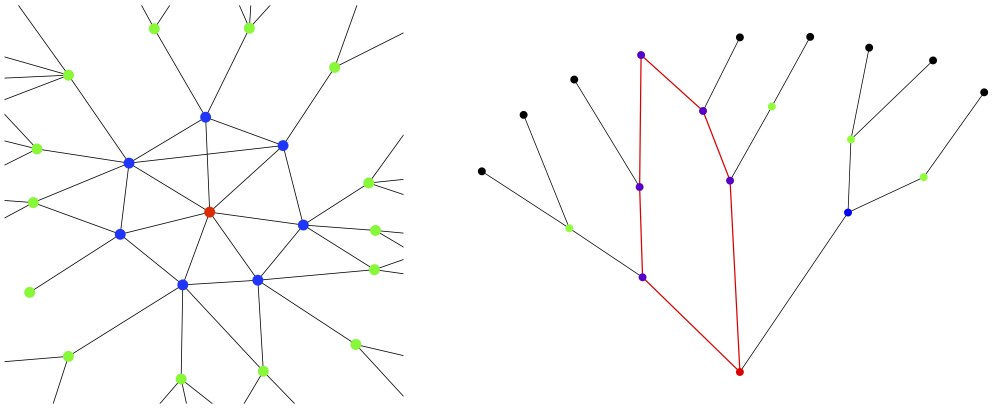


Figure 4.1: Geodesic neighbourhoods $N_i^{(r)}$ versus *message passing* neighbourhoods $\mathcal{N}_i^{(r)}$. On the left, geodesic neighbourhoods around i (in red) of order one (in blue) and two (in green). On the right, message passing neighbourhood of order $r = 3$ (in purple), with the generating cycle highlighted (in red).

correction to the result in section 4.2.3 and the approximation will work well.

For graphs with short cycles, we extend our approximation following [71, 72]. Let's start by defining the r-neighbourhood around each node: for a given integer r, the r-neighbourhood of node i consists of all of the nodes immediately adjacent to node i, along with all edges and all nodes on paths of length r or shorter between nodes adjacent to i. We denote the r-neighbourhood of node i as $\mathcal{N}_i^{(r)}$. This is different form the usual *geodesic neighbourhoods* $N_i^{(r)}$ defined as the set of nodes at a graph distance $d \leq r$ from i, see figure 4.1

This allows us to write the message-passing approximation with minimal changes to the arguments we used before. Any excursion from node i will first move to an immediate neighbour j. Once at j, the excursion will leave the neighbourhood and return to j some $m \geq 0$ times. After this, it will either immediately return to i or move on to some other node in the neighbourhood.

Because now the graph contains cycles, it is no longer true in general that if random walk moves to a neighbour of $k \in \mathcal{N}(j)$, $k \neq i$, it needs to transit over j.

However, we can instead use the r-neighbourhoods to analyse the excursions. In fact, each excursion from node i can be broken down into two parts:

- an excursion in the r-neighbourhood of i
- a number of independent excursions from each node in the neighbourhood.

An excursion from i in the neighbourhood $\mathcal{N}_i^{(r)}$ can be denoted $w = i \to j \to \cdots \to i$. Let the set of all excursions in the neighbourhood of i be labelled W_i. Then the probability of returning at time t is

$$\widetilde{Y}_i(t) = \frac{1}{k_i} \sum_{w \in W_i} \sum_{t_j : j \in w} \delta(t, l + 1 + \textstyle\sum_{j \in w} t_j) \displaystyle\prod_{j \in w} \frac{X_{i \leftarrow j}(t_j)}{k_j} \tag{4.33}$$

where l is the length of the excursion w, and the Dirac δ function again accounts for the length of the excursion as the total first return time minus the first and last steps. From this, we can compute the generating function

$$\begin{aligned} \widetilde{F}_i(z) &= \sum_{t=1}^{\infty} \widetilde{Y}_i(t) z^t = \\ &= \frac{1}{d_i} \sum_{w \in W_i} \sum_{\{t_j : j \in w\}} z^{l+1} \prod_{j \in w} X_{i \leftarrow j}(t_j) \frac{z^{t_j}}{d_j} = \\ &= \frac{z}{d_i} \sum_{w \in W_i} \prod_{j \in w} \frac{z\, R_{i \leftarrow j}(z)}{d_j} = \\ &= \frac{z}{d_i} \sum_{w \in W_i} \prod_{j \in w} \frac{z}{d_j - d_j \widetilde{F}_{i \leftarrow j}(z)}. \end{aligned} \tag{4.34}$$

Following the same arguments as in section 4.2.4, we get:

$$\widetilde{F}_{i \leftarrow j}(z) = \frac{z}{d_j} \sum_{w \in W_{j \setminus i}} \prod_{k \in w} \frac{z}{d_k - d_k \widetilde{F}_{j \leftarrow k}(z)}. \tag{4.35}$$

The sum over all walks can be computed by matrix inversion. Define the following

vector quantities:

$$u_j^i = \frac{A_{ij}}{d_i} = W_{ij} \tag{4.36}$$

$$v_j^i(z) = \frac{A_{ji}}{d_j - d_j \widetilde{F}_{i\to j}(z)}, \tag{4.37}$$

and the matrix

$$B_{jk}^i(z) = \frac{A_{jk}}{d_j - d_j \widetilde{F}_{i\to j}(z)}, \tag{4.38}$$

using which we can simplify equation (4.34) to the vector form:

$$\widetilde{F}_i(z) = \boldsymbol{u}^i \big(1 - B^i(z)\big)^{-1} \boldsymbol{v}^i(z). \tag{4.39}$$

Similarly with the previous approximations, Eqs. (4.34) and (4.35) do not meet the same requirements that $\widetilde{F}_i(1) = 1$ or $\widetilde{F}_i'(1) = 2m/k_i$. This we make the same modification as before: adding the generating function of a geometric distribution with an appropriately defined mean. In other words, letting

$$h_i = \frac{2m - k_i \widetilde{F}_i'(1)}{k_i - k_i \widetilde{F}_i(1)} \tag{4.40}$$

we approximate $F_i(z)$ as

$$F_i(z) = \widetilde{F}_i(z) + \frac{z - z\widetilde{F}_i(1)}{z + h_i - z h_i} \tag{4.41}$$

Which, using the $\tilde{F}_i(z)$ from equation (4.39), is the most general approximation for the first time distribution, which reduces to previous approximations under the correct assumptions. More explicitly, this formula encompasses earlier approximation, which can now be seen as specific examples of this more general form."

4.3 Numerical simulations

In this section, we report on the numerical simulations results verifying the approximations we have developed. In particular, section 4.3.1 contains an explanation of the numerical method used for our estimations, while section 4.3.2 contains the results.

4.3.1 Computing return time distributions

Given a network $\mathcal{G}$, we can in principle estimate the average first return time and its variance from random walk simulations. However, sampling it directly is not efficient. We can instead use its definition in terms of the probability distributions from which it is generated.

$$\langle Y_i \rangle = \sum_{t=0}^{+\infty} t\, Y_i(t) \qquad \left\langle Y_i^2 \right\rangle = \sum_{t=0}^{+\infty} t^2\, Y_i(t) \tag{4.42}$$

To estimate them we can proceed as follows: we take the row normalised adjacency matrix

$$W_{ij} = \frac{A_{ij}}{\sum_j A_{ij}}$$

of the graph and we substitute the i-th row with the indicator vector of the i-th component, that is, the vector that has zero in each component but the i-th one, where it has one. This procedure is equivalent to rewiring the network to have the vertex i to be an absorbing node. We call P_i the matrix obtained with this substitution:

$$P_i = \begin{pmatrix} P_{1,1} & P_{1,2} & \dots & & \cdots & & \dots & P_{1,n} \\ \vdots & \vdots & \dots & & \vdots & & \dots & \vdots \\ P_{i-1,1} & P_{i-1,2} & \dots & & P_{i-1,i} & & \dots & P_{i-1,n} \\ 0 & 0 & \dots & 0 & 1 & 0 & \dots & 0 \\ P_{i+1,1} & P_{i+1,2} & \dots & & P_{i+1,i} & & \dots & P_{i+1,n} \\ \vdots & \vdots & \dots & & \vdots & & \ddots & \vdots \\ P_{n,1} & P_{n,2} & \dots & & \cdots & & \dots & P_{n,n} \end{pmatrix} \tag{4.43}$$

Then, starting from the uniform vector $\vec{x_0} = \vec{1}$, and the first iterated vector $\vec{x_1} = P\vec{x_0}$ we iterate the matrix P_i, and obtain the vectors sequence $\vec{x_2}, \vec{x_3}, \ldots$. This sequence of vectors has the following property: the i-th component of the n-th vector of the sequence, $\vec{x_t}(i)$, is the probability of the return time for the v_i vertex to be smaller or equal than t (since we have set i to be an absorbing state we can't say whether the random walker has just arrived there or has been trapped there for any number of time steps). More formally:

$$\vec{x_t}(i) = P(X_{t_0+t} = v_i | X_{t_0} = v_i) \tag{4.44}$$

From the definition of $Y_i(t)$ in equation (2.16) we can conclude that the vectors $\vec{x_t}$ and the first return times $Y_i(t)$ are related as follows:

$$Y_i(t) = \vec{x_t}(i) - \vec{x_{t-1}}(i) \tag{4.45}$$

We can therefore iterate the matrix P_i, obtain the sequence of vectors $\vec{x_1}, \vec{x_2}, \ldots,$ calculate the distribution of the $Y_i(n)$ and use it to calculate the moments of the distribution. However, in this last step we need to truncate the sums, as follows:

$$\langle Y_i \rangle = \sum_{t=0}^{+\infty} t\, Y_i(t) \approx \sum_{t=0}^{T} t\, Y_i(t) \tag{4.46}$$

$$\left\langle Y_i^2 \right\rangle = \sum_{t=0}^{+\infty} t^2\, Y_i(t) \approx \sum_{t=0}^{T} t^2\, Y_i(t) \tag{4.47}$$

the errors in the last approximations decrease and converge to zero in the limit of large T if $Y_i(t) \sim o\,(1/t^2)$. We know by Kac's lemma that $Y_i(t) \sim o\,(1/t)$, which is not enough to bound the squared average return time. However, we know that once a random walk has normalised, it crosses each edge with equal probability at each step, so the probability of reaching the vertex v_i at each step is asymptotically equal

to:

$$p_i := \mathbb{P}\left(X_{t_0+t} = v_i\right) \sim \frac{d_i^{in}}{2E} \tag{4.48}$$

The probability of not reaching the vertex v_i at each step will therefore be $1 - p_i$, and thus the probability of not reaching it before the first t steps will be $(p_i)^t$ if the probabilities at different steps are independent. In principle they are not, as the random walker will next visit a specific node $X_{t+1} = w$ depends on whether such node is in the neighbourhood of the current node $\mathcal{N}(X_t)$. However, when passing to an ensemble of random walk on the same network that have run long enough to having thermalised, the probabilities *are* indeed independent, as the ensemble average random walk will traverse each edge with equal probability regardless of where the single constituent random walkers are located at the previous time step. As such, we obtain:

$$\mathbb{P}\left(Y_i > t\right) \sim (1 - p_i)^t \tag{4.49}$$

which is a convergence rate fast enough to guarantee finiteness for the variance for all finite size network, and in which exact inequality does not hold as the assumptions are only valid for thermalised random walks, so in the limit of large t. In practice, we observe fast thermalisation in all networks, something that we expect as a result of the small world property and that we are able to check through the use of Kac's lemma and equation (4.46).

When studying the first return time distribution, the method above is quick and convenient whenever one is not interested in the behaviour of the distribution for long times, because it allows for the truncation of the calculation at any time T. The drawback is that higher momenta will have significant errors, as the sums in equations (4.46) and (4.47) must in principle be taken up to infinity, and the power term may yield significant contributions if we are not careful to bound the decay of the $Y_i(t)$.

Another method avoids this problem completely by using the generating functions that we introduced in section 2.2.1. While computationally more expensive, it returns the distribution correct up to the machine rounding error. Let's start by calculating the generating function of the first return time distribution:

$$F_i(z) = \sum_{t=1}^{+\infty} e^{zt} Y_i(t) \tag{4.50}$$

which we can then rewrite using the same argument used for the previous method:

$$F_i(z) = \vec{\delta_i} \left(\sum_{t=1}^{+\infty} e^{zt} P_i^{t-1} \right) P\vec{\delta_i} = \tag{4.51}$$

$$= \vec{\delta_i} \left(\frac{e^z}{1 - e^{sP_i}} \right) P\vec{\delta_i} = \tag{4.52}$$

$$= \vec{\delta_i} \left(e^{-z} - P_i \right)^{-1} P\vec{\delta_i} \tag{4.53}$$

The second step of the method uses the property of the generating function to calculate the entire distribution in one go. We use the relation:

$$\mu(t) = \frac{1}{t!} \frac{\mathrm{d}^t F}{\mathrm{d}z^t} \bigg|_{z=0} \tag{4.54}$$

so, all we need to do is to calculate the derivative on the right-hand side. To do so, we move to the complex plane $\mathbb{C}$ and we use the residual theorem using the unitary circumference border as the integration path. Formally, we will set $z = e^{it}$ and bound t in $[0, 2\pi]$. We obtain:

$$\mu(t) = \frac{1}{t!} \frac{\mathrm{d}^t F}{\mathrm{d}z^t} \bigg|_{z=0} = \tag{4.55}$$

$$= \frac{1}{2\pi i} \oint \frac{F(z)}{z^{t+1}} \mathrm{d}z = \tag{4.56}$$

$$= \frac{1}{2\pi} \int_0^{2\pi} e^{-ist} F\left(e^{is}\right) \mathrm{d}s \tag{4.57}$$

which allows us to calculate the full distribution with a single fast Fourier transform (FFT) of the quantity we obtained in equation (4.53). The bulk of the computational cost is in the matrix inversion in equation (4.53).

4.3.2 Numerical results

We now compare the approximations from section 4.2 and the direct computation of the first return times from the numerical procedure from section 4.3.1. We present the results for three classes for networks:

- Erdős-Rényi random regular graph, a modified version of the traditional random graph in which all nodes have the same degree $d_i = d = \langle d \rangle$
- a traditional SBM model on N nodes, in which the number of communities N_C is fixed and does not change with the size of the network, and the community size grows with the network size: $|C| = N/N_C = O(N)$
- a modified version of the SBM model on N nodes, in which the size of the communities $|C|$ is fixed and does not change with the size of the network, and the number of communities grows with the network size: $N_C = O(N)$

In the case of the random d-regular graph, we have:

$$h = \frac{(d-1)\,(nd-2n-2)}{(d-2)^2} \tag{4.58}$$

and then

$$F(z) = \frac{2z^2}{d+\sqrt{d^2+4z^2-4dz^2}} + \frac{z-z/(d-1)}{z+h-zh}. \tag{4.59}$$

In Fig. 4.2 we compare this approximation to random regular graphs with $n = 2^{15}$ nodes and degree $d = 6$. We find excellent agreement between the predictions of Eq. (4.59) and simulations.

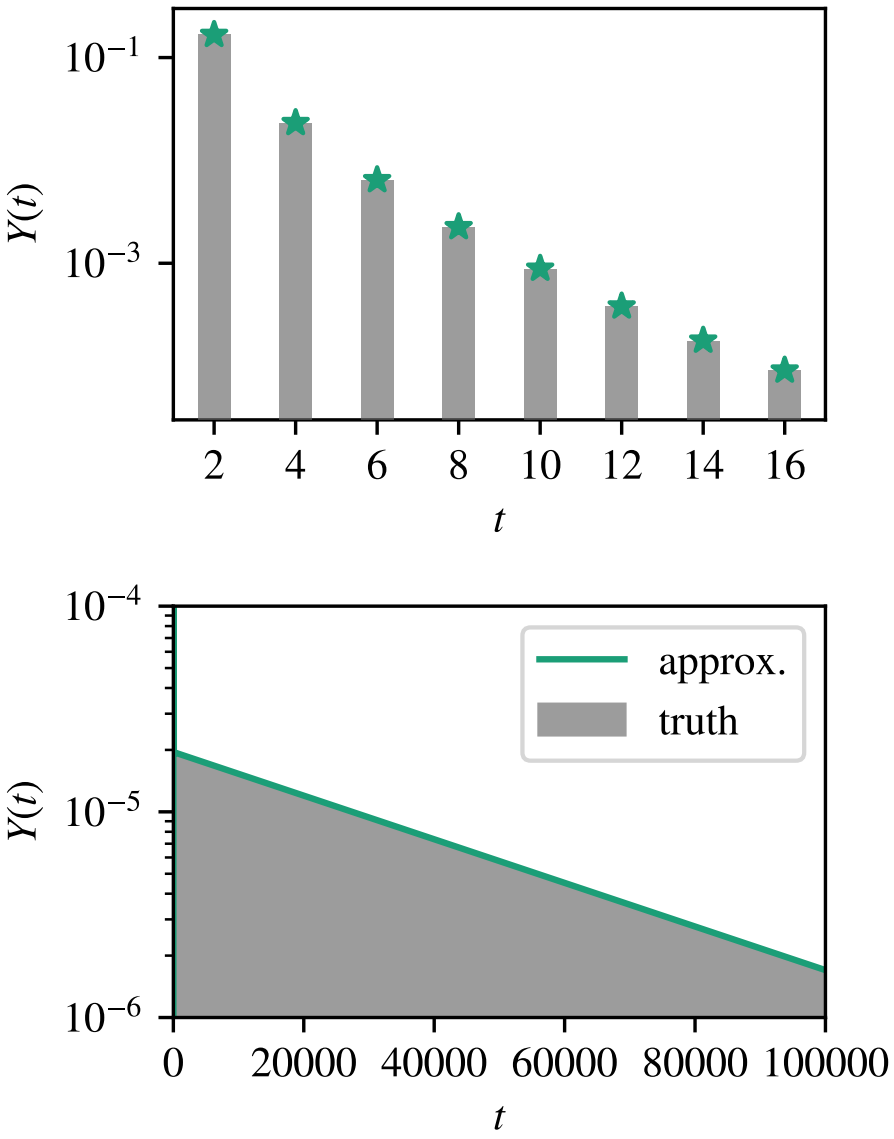


Figure 4.2: First return time distribution, $Y(t)$, for random walks on a 6-regular random graph with $n = 2^{15}$ nodes. The ground truth is computed for an arbitrary node with matrix iteration. The approximation is computed from derivatives of Eq. (4.59) at $z = 0$. Note, the tail of the distribution is straight on a semi-log plot, indicating exponential decay. The slope of this tail on a logarithmic scale is the inverse of the quantity we call the tail-mean, approximate by Eq. (4.30).

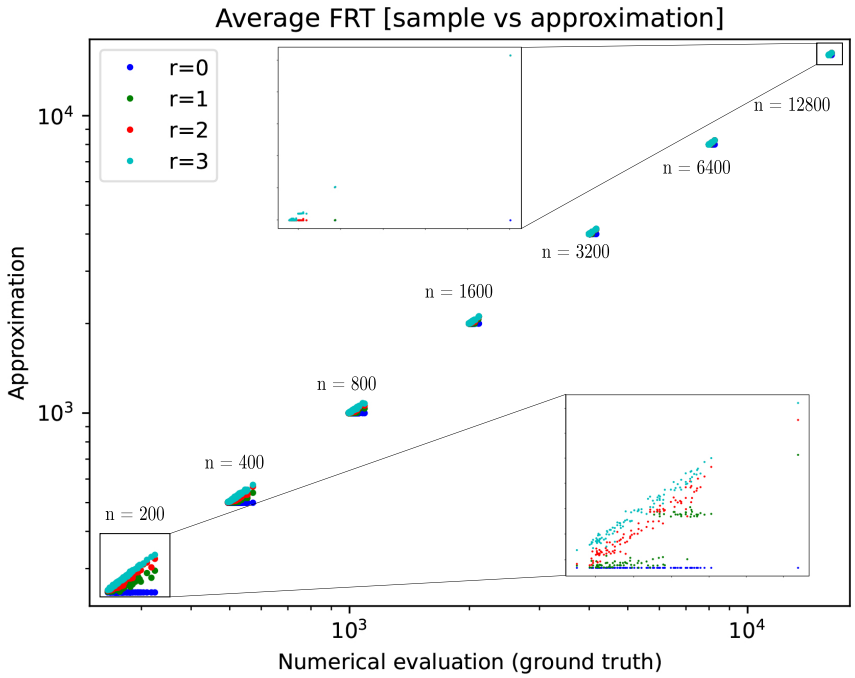


Figure 4.3: First return time distribution approximations against ground truth, for random regular graph of various sizes. The exponential tail of randomly chosen nodes in networks of differing sizes was compared to approximations. As the number of nodes increases, the tree approximation ($r = 0$) moves towards the line $y = x$, showing that the approximation is converging to the truth. For the $r = 3$ approximation, the points are close to the line $y = x$ even for small N.

Interestingly, the quality of the predictions of this approximation do not change regardless of large-scale structure, such as community structure, or core-periphery structure [73]. In other words, the fact that our approximation only depends on a local property (node degree) and a global property of the network (total number of edges) is supported by the numerical simulation.

For example, a stochastic block model with N nodes and N_C equally sized communities is locally tree-like if N_C does not grow with N [74, 75]. Thus, our approximation predicts that, even when the community structure is extremely strong, random walks will appear to behave on the block model as though they are on a random graph for sufficiently large n. To verify this prediction, we compared the tree-like predictions to ground-truth on stochastic block models of increasing size.

In Fig. 4.4, we compare the tail slope predicted by Eq. (4.30) to those found by numerically iterating the random walk matrix. These networks were generated by the degree regular stochastic block model sampler of graph-tool [76]. Each network has 20 communities and each node have degree 6. The community structure is relatively strong. Parameters were set so that, on average, two thirds of the neighbours of any node will be in the same group as the node itself. But, as N increases we see the approximation converges to the ground truth values. Still, one should not conclude that the tree approximation is good for all networks. For instance, in Fig. 4.4 we see that the approximation is less accurate for small networks. This, we believe, is due to the presence of many short cycles in the graphs.

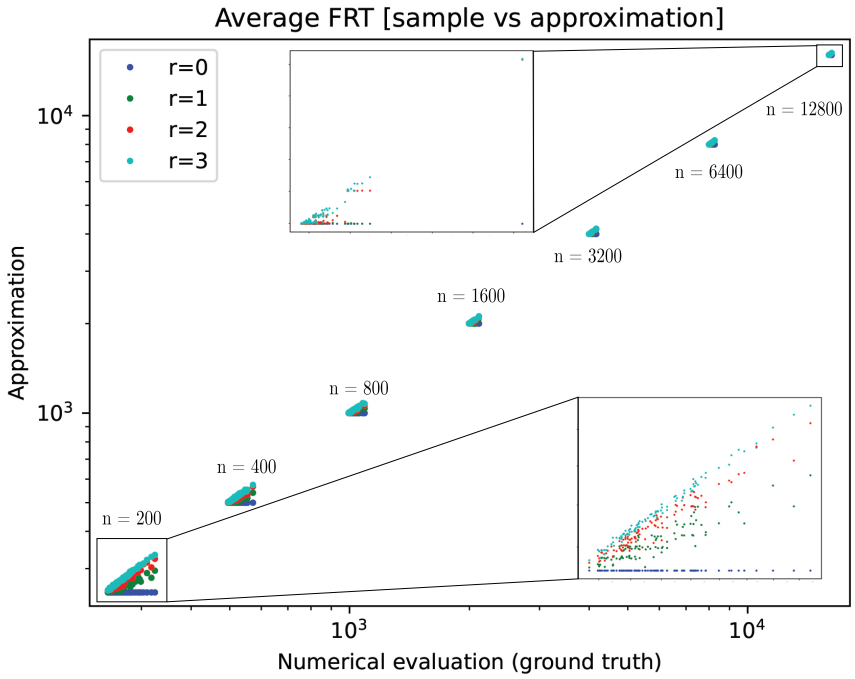


Figure 4.4: First return time distribution approximations against ground truth, for regular stochastic block models of varying size but with a fixed community structure. The exponential tail of randomly chosen nodes in networks of differing sizes was compared to approximations. As the number of nodes increases, the tree approximation ($r = 0$) moves towards the line $y = x$, showing that the approximation is converging to the truth. For the $r = 3$ approximation, the points are close to the line $y = x$ even for small n.

To further investigate how the approximation fails, we look at the case in which the number of communities grows with the size of the network, specifically $N_C = N/20$. In this case, the networks are not locally tree-like, and so it is not surprising that the properties of the random walks are significantly different from those on random graphs. Figure 4.5 shows the results of these experiments. In this case we see that even as n increases, the approximation fails to converge to the ground truth.

For large N, the ground truth and both approximations of Eqs. (4.31) and (4.41) converge to a point on the line $y = x$. However, for smaller N, where there is a significant variance due to finite size effects, only the improved approximation of Eq. (4.41) correctly falls on the line $y = x$. In Fig. 4.5, where the number of communities grows linearly with network size, we observe the discrepancy between the approximation of Eq. (4.31) for all N. In contrast, we see that the approximation of Eq. (4.41) is good for all N.

4.4 Implications on community detection algorithm

The results in the previous section are interesting and quite surprising, as they reinforce the idea that random walks are not affected by the details of the community structure of the graph, and indeed random walks look the same on all classes of locally tree-like graphs.

This is in stark contrast with the use of random walks in community detection methods, so in the remainder of the chapter, we shall investigate this contradiction further. More specifically, in section 4.4.1 we introduce a variation of the SBM model that allows us to conveniently control the community strength in the network, in section 4.4.2 we analyse the first return distribution across the SBM phase transition, and in section 4.4.3 we focus on the analysis of the InfoMap target function, which sheds light on the issues and seems to suggest that the ability of random walks to

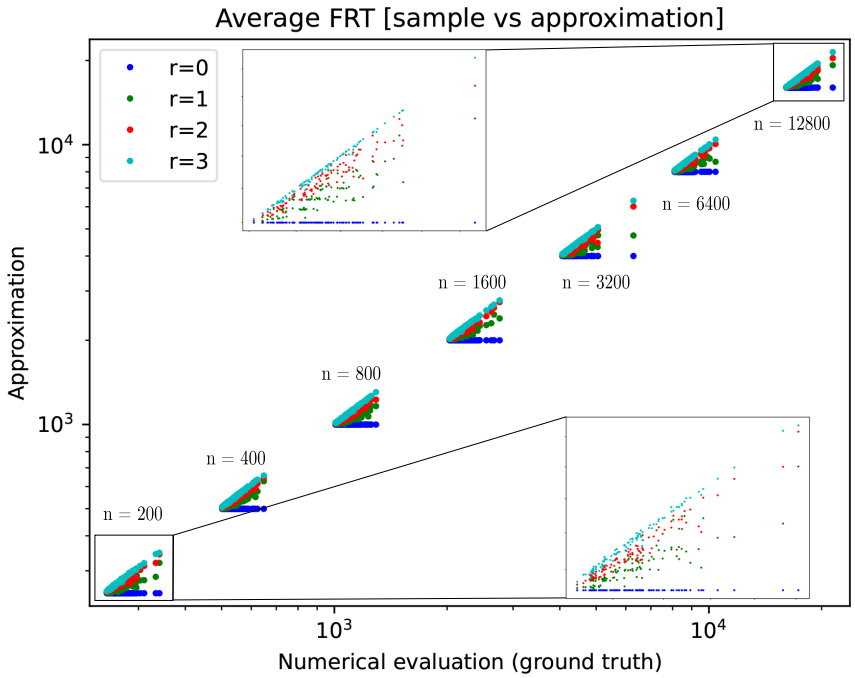


Figure 4.5: First return time distribution approximation against truth for regular stochastic block models of various sizes variable number of communities $N_C = N/20$. The exponential tail of randomly chosen nodes in networks of differing sizes was compared to approximations. The predictions for the tree approximation are not very close to the line $y = x$, even as n increases. Conversely, the predictions for $r = 3$ remain close to $y = x$ for all n.

pick up the community structure is due to a finite size effect.

We shall refer the reader to section 4.5 for further discussion.

4.4.1 Micro-canonical SBM model

In order to test the return effect of the random walks on the community detection methods we want to run the test on a class of networks that offers both a tunable parameter α for the community strength and a fixed average degree $\langle d \rangle$ so that the first return time distributions for the random walks can be compared like-for-like.

We therefore introduce modified version of the standard Block Model (SBM) [77] graph that satisfies both those requests. The original SBM model is a good candidate,

because the planted community structure can be tuned by modifying the p_{in} and p_{out} parameters, controlling the probability of an edge between nodes inside the same community and belonging to different communities, respectively. However, this causes the average degree of each node $\langle d \rangle = \frac{1}{N}\sum_i d_i$ to be distributed as the sum of two independent binomial random variables, one for the generation of infra-community edges and one for inter-community edges. We are interested in comparing the return time distribution between the Erdős-Rényi random graph and graph with a planted community structure. Therefore, we would like to fix the average return time $\tau = \frac{1}{N}\sum_i \tau_i$, where τ_i is the average return time for the node x_i. Using Kac's Lemma,

$$\tau_i = \frac{2E}{d_i} \tag{4.60}$$

we immediately see that this is equivalent to fixing the average node degree $\langle d \rangle$, which in turn is governed, for each size of the network N, by the total number of edges E

$$\langle d \rangle = \frac{2E}{N} \tag{4.61}$$

In our modified version of the SBM model $\langle d \rangle$ (and consequently E) is no longer a random variable, but a known fixed value of the network. The model we propose not only surpasses the problem of E being a random variable, but it uses E as an input of the model, so that it is possible to generate an SBM-like graph with an arbitrary average degree $\langle d \rangle$. This is particularly helpful when generating an ensemble of such networks for statistical purposes, as we no longer need to consider the variance of $\langle d \rangle$ across different realisations.

We call this modified SBM *micro-canonical*, drawing from statistical physics. Just as the canonical ensemble fixes the overall temperature T of the system causing the energy to fluctuate around an average value $\langle E \rangle = F + ST$ (in which F is the Helmholtz free energy and S the Gibbs entropy), the "canonical" SBM fixes the

probabilities p_{in} and p_{out} but leaves the average degree $\langle d \rangle$ free to fluctuate around the average value $\langle d \rangle = \frac{1}{N} \sum_i d_i$ across different graph of the SBM ensemble. By contrast, the micro-canonical ensemble fixes the energy directly and the micro-canonical SBM fixes the total number of edges E, and therefore the average degree $\langle d \rangle$ as a parameter of the model.

More specifically, the micro-canonical SBM takes four parameters: the total number of nodes, N the total number of edges E, the number of communities N_C, and the community strength parameter α. Communities are assumed to be equal size, so the size of each community $C_1, C_2, \ldots, C_{N_C}$ is simply equal to N/N_C. The parameter α is defined by the percentage of edges that are generated between edges belonging to the same community. All other edges are generated by connecting two notes chosen at random but not belonging to the same community. Note that N and N_C are the same parameters that are used in the classic SBM networks, while E and α substitute p_{in} and p_{out}.

The parameter α is the one that controls the strength of the community structure. For example, a choice of $\alpha = 1$ would generate N_C disconnected communities of size N/N_C, while $\alpha = 0$ would yield a N_C-multipartite graph which only contains edges that connect nodes in different communities. In between these two limit cases, the model returns a SBM-like graph with fixed average degree $\langle d \rangle = E/N$ and different strength of the community structure. We first observe that there is a value for the parameter α at which the generates a graph that is indistinguishable from an Erdős-Rényi random graph. We call this value $\alpha^\dagger$ and we can find it by equating the number of infra- and inter-community edges in the modified SBM model and an Erdős-Rényi model with the same average degree and verifying that the two conditions are in agreement.

Firstly, the requirement on the average degree fixes the Erdős-Rényi parameter p

as follows:

$$\langle E \rangle_{ER} = \frac{N(N-1)}{2} p = E \;\Rightarrow\; E = \frac{N(N-1)}{2} p \tag{4.62}$$

Now the total number of infra-community edges in the modified SBM is simply $\alpha^{\dagger} E$ and we need to equate this to the number of infra-community edges in the Erdős-Rényi random graph, which is equal to the total number of edges inside a community of size N/N_C, times the number of communities N_C times the ER parameter p:

$$\frac{\frac{N}{N_C}\left(\frac{N}{N_C}-1\right)}{2} \times N_C \times p = \alpha^{\dagger} E = \alpha^{\dagger} \frac{N(N-1)}{2} p \tag{4.63}$$

Solving this for $\alpha^{\dagger}$ yields:

$$\alpha^{\dagger} = \frac{\frac{N}{N_C} - 1}{N-1} \tag{4.64}$$

We still need to guarantee that the condition on the inter-community edges is compatible with equation (4.64). I other words, we need to show that the choice of $\alpha^{\dagger}$ in equation (4.64) also produces the same number of inter-community edges as we would expect in an ER random graph with parameter p satisfying equation (4.62).

So, the number of edges across different communities in the modified SBM model is simply $(1-\alpha^{\dagger})E$, while in the ER random graph the expected number is equal to the total number of pairs across two different communities, times the number of possible permutations of two communities, times the ER parameter p. Hence:

$$\frac{N^2}{N_C^2} \times \frac{N_C\,(N_C-1)}{2} \times p = (1-\alpha^{\dagger})E \tag{4.65}$$

Again substituting E from equation (4.62) and solving equation (4.65) for $\alpha^{\dagger}$, we find the same value as equation (4.64).

Furthermore, we studied the detectability parameter for the SBM model, which takes the form:

$$p_{in} - p_{out} > \sqrt{2N\,(p_{in} + p_{out})} \tag{4.66}$$

we need to express p_{in} and p_{out}, the probabilities of an infra- and intra-community edge to present, as functions of the parameters of our model. Simply by calculating them as the ration of inter- (infra-) community edges created over the total number of possible inter- (infra-) community edges, we get:

$$p_{in} = \frac{\alpha E}{\frac{\frac{N}{N_C}\left(\frac{N}{N_C}-1\right)}{2} \times N_C} = \frac{2\alpha E}{N\left(\frac{N}{N_C}-1\right)} \tag{4.67}$$

and

$$p_{out} = \frac{(1-\alpha)E}{\frac{N^2}{N_C^2}\frac{N_C(N_C-1)}{2}} = \frac{2EN_C(1-\alpha)}{N^2(N_C-1)} \tag{4.68}$$

We know that four our modified SBM model we have

$$p_{in} = \frac{2\alpha E}{N\left(\frac{N}{N_C}-1\right)} \qquad p_{out} = \frac{2EN_C(1-\alpha)}{N^2(N_C-1)}$$

and we know that the detectability rule for an SBM model is

$$p_{in} - p_{out} > \sqrt{2N\left(p_{in} + p_{out}\right)}$$

We can calculate easily:

$$p_{in} - p_{out} > 0 \Leftrightarrow \alpha > \frac{N - N_C}{NN_C - N_C}$$

In the general case when $N_C \ll N$ we have the condition simplifies to $\alpha > 1/N_C$, which is a correct but very week bound, as we normally work with $N_C = 4$ and therefore $\alpha \in [0.9, 1[$.

4.4.2 FRTD across the SBM phase transition

Our intuition that we briefly introduced in section 4.1 is rooted in the idea that random walks should look the same regardless of the fine details of the community structure. This is what makes it possible for our approximations in section 4.2 to work so well despite only relying on local and global properties of the networks.

Now we want to put this intuition to the test and use our micro-canonical SBM to generate instances of networks with different strengths of community structure with constant average degree. We can then compare the first return time distributions across the phase transition, that is for values of the community strength parameters smaller and greater than the critical value $\alpha^{\dagger}$.

We consider networks with N nodes and N_c communities of equal size. The use of the micro-canonical SBM ensures that the degree distributions are centred around the same average degree $\langle d \rangle$, so by Kac's lemma, equation (4.3), we get a trivial behaviour of the average return time. The full data of the graph testes are reported in table 4.1.

Graph	No. nodes [N]	No. edges [E]	No. communities [N_C]
$\mathcal{G}_1$	200	800	2
$\mathcal{G}_2$	200	800	20
$\mathcal{G}_3$	2000	8000	2
$\mathcal{G}_4$	2000	8000	20

Table 4.1: Table of the parameters (N, E, N_C) used in the simulations to .

We calculate the full return time distribution with the method described in section 4.3.1 for the graphs in table 4.1. We report the first two momenta of the return time distribution.

Then we move on to study the dependence of the return time distribution on the connectivity parameter p, the main interest of this study. Since we know the cutoff time T, we can focus on simulating many more values of the connectivity p. The chosen values ranged from $p = 0.1$ up to $p = 0.975$, bracketing well below and above

the critical value $\alpha^{\dagger}$ of the SBM phase transition.

For graphs with two communities, we do not observe significant differences in the variance when changing the strength of community structure[1]. The size of the network also appears to be irrelevant for the distribution of the variance. However, we can see that in graphs with more communities, the variance does shift significantly depending on the community strength, and this effect is more pronounced in smaller networks, suggesting that the ratio N_C/N is crucial in

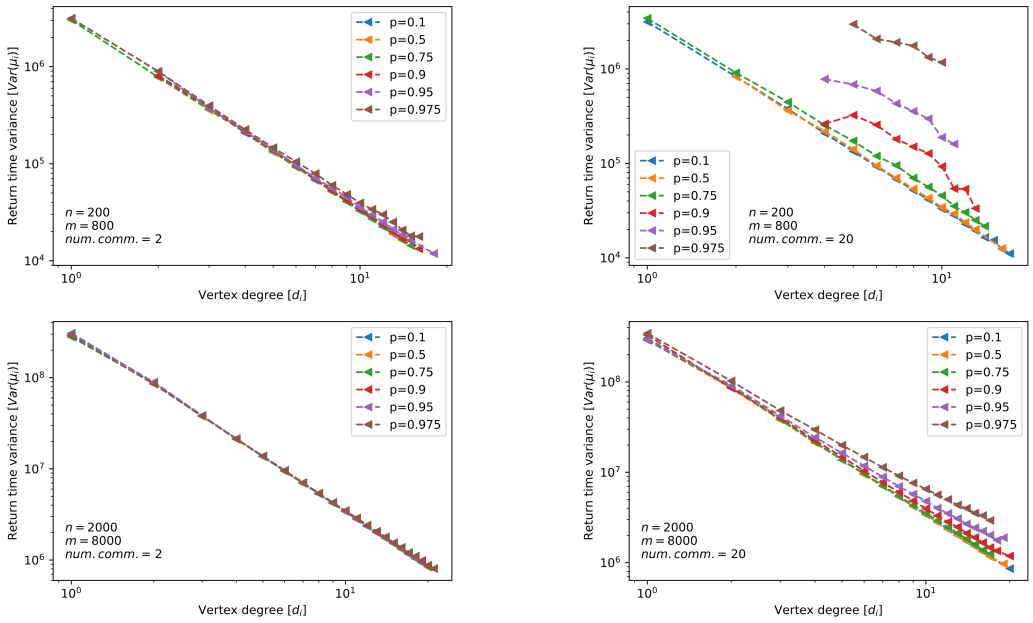


Figure 4.6: Variance of the return time distribution, for different values of the α parameter, as described in section 4.4.1. Other parameters are the graph size (n, m) and the number of communities. The baseline is the Erdős-Rényi graph, for which the variance decays exponentially as a function of the node degree. The results for the SBM deviate more strongly when the number of communities is higher, even for equal values of the r parameter.

We perform simulations in order to test the hypothesis we formulated in the previous section. We claim, in short, that the distribution of the return times of

[1]Note here the convenience of the micro-canonical SBM model, which allows to tune the community strength with a single parameter while leaving untouched the average degree. This allows for the data in figure 4.6 to be potted against the node degree and obtaining comparable results across the four graphs in table 4.1. Using a traditional SBM method here would have meant recalculating all the in- and out- probabilities in order to keep the average degree fixed, while simultaneously varying the community strength.

random walkers on a strongly connected time has the following properties:

- the average is independent on the community structure of the network, in accordance with Kac's lemma
- the higher momenta - and the variance specifically - exhibit a more complex behaviour, but this is driven by whether the network is locally-tree like rather than the strength of the community structure.

The second point is crucial in our discussion and seems to support our intuition that random-walk based community detection methods should not work. The discerning factor for random walks is whether or not the graph is locally-tree like. We will develop this analysis more thoroughly in the coming section.

4.4.3 Analytical analysis of the InfoMap function

We test the clustering results of the InfoMap clustering methods (see section 2.4.2. We have run the simulations with the following procedure way: we generated several micro-canonical SBMs with various community strengths parameters and various sizes of the networks. For each network, we have the planted community structure and we calculate the InfoMap clusters, fixing the number of search clusters as the same number of planted communities we have. We score the InfoMap result using the normalised mutual information, see section 2.4.3. The normalised mutual information is a standard measure of the quality of clustering, and it is widely used in the literature [78]. However, it has been shown to fail when the network presents too many clusters [79]. To avoid this issue, we have kept the number of planted community small, and we have also fed the InfoMap algorithm the N_C parameter, which is used to suggest InfoMap the correct number of clusters to find and will penalise solutions that returns a number of clusters far from N_C.

We report the Normalised Mutual Information score between the two clustering in figure 4.7.

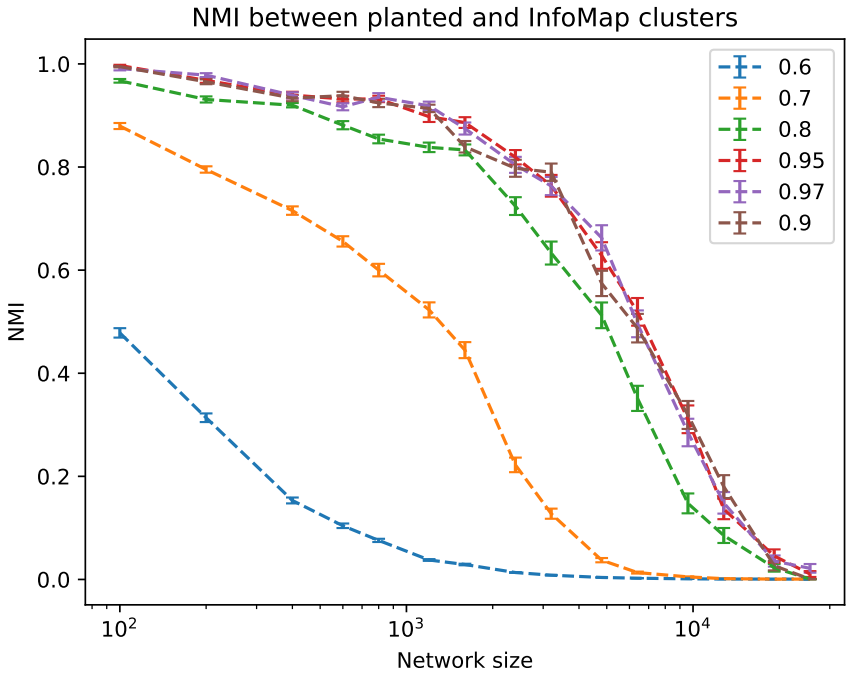


Figure 4.7: Normalised mutual information between the planted clusters of a micro-canonical SBM model and the clusters returned by the InfoMap algorithm, as a function of the network size N. Each data point has been averaged over 200 iterations of the micro-canonical SMB model with the same parameter, and the error bar indicate one standard deviation.

The data show that InfoMap fails in the limit of large, sparse graphs. More precisely, we express the claim as follows. *Let $\mathcal{C} = NMI(C, C')$ be the normalised mutual information between the planted community assignment and the InfoMap Result. No matter the strength of the community structure (i.e., the value of the α parameter in the micro-canonical SBM) we have:*

$$\lim_{N\to\infty} \mathbb{P}(\mathcal{C} < 1 - \epsilon) = 1 \tag{4.69}$$

In other words, no matter the community strength, the probability of InfoMap returning the correct clusters goes to zero when the size of the network grows. We

hypothesize that this is due to the locally tree-like structure of both the ER and SBM models in the limit of $N \to \infty$. For further disucussion on this hypothesis, please refer to section 4.5.

Now we want to further investigate the issue that we are seeing in figure 4.7 and our hypothesis above. To do so, we now study the analytic target function of InfoMap $L(M)$ and show that, in the same limit, the value of the function is arbitrarily small in the case of a planted community structure and a random graph. If that is the case, this result is strengthening our hypothesis.

We first report the InfoMap target function as seen in [36]:

$$L(M) = q_{\curvearrowright} H\left(\mathcal{Q}\right) + \sum_{i=1}^{m} p^i_{\circlearrowright} H\left(\mathcal{P}^i\right) \tag{4.70}$$

We first calculate the average (expected) value of equation (4.70) in the case of an SBM. For simplicity, we will start with a regular SBM, in which all vertices have degree d.

In the case of only one module, the first term is zero, as $q_{\curvearrowright} = \sum_i q_{i\curvearrowright}$ and the probability $q_{i\curvearrowright}$ that the random walker exits module i at each step is zero, as there is no other module to exit to. So, we need to compute the second term, for which the complicated part is really just computing $H\left(\mathcal{P}^i\right)$. Using [36] we have:

$$H\left(\mathcal{P}^i\right) = \frac{q_{i\curvearrowright}}{q_{i\curvearrowright} + \sum_{v\in i} p_v} \log\left(\frac{q_{i\curvearrowright}}{q_{i\curvearrowright} + \sum_{v\in i} p_v}\right) \tag{4.71}$$

$$+ \sum_{w\in i} \frac{p_w}{q_{i\curvearrowright} + \sum_{v\in i} p_v} \log\left(\frac{p_w}{q_{i\curvearrowright} + \sum_{v\in i} p_v}\right) \tag{4.72}$$

Now equation (4.71) is equal to $0 \log 0$ for the same observation made above. Here we assume such quantity to be zero from the observation that the first term is equal to the argument of the logarithm and from the standard limit $\lim_{x\to 0} x \log x = 0$.

Hence, equation (4.72) is the only nonzero term, yielding:

$$H\left(\mathcal{P}^i\right) = \sum_{w \in i} \frac{p_w}{\sum_{v \in i} p_v} \log\left(\frac{p_w}{\sum_{v \in i} p_v}\right) \tag{4.73}$$

Now using the assumption of regular graph, we get:

$$H\left(\mathcal{P}^i\right) = \sum_{v \in i} \frac{1/N}{N_C/N} \log\left(\frac{1/N}{N_C/N}\right) = \sum_{v \in i} \frac{C}{N} \log\left(\frac{C}{N}\right) = \log(N) \tag{4.74}$$

In which we have used the fact that there is only one module, $C = 1$. The last term left to compute is $p^i_{\circlearrowleft}$, which we can write as:

$$p^i_{\circlearrowleft} = q_{i\curvearrowright} + \sum_{v \in i} p_v = \sum_{v \in i} p_v = 1/C = 1 \tag{4.75}$$

Putting everything together, we can estimate the target function:

$$L(M) = \log(N) \tag{4.76}$$

In the case of a regular SBM with only one module.

Now, we want to calculate the same quantity in the case of a regular SBM with a fixed size N, with a planted community structure of C communities of size $N_i = N/C$ and community strength parameter α.

In order do so, it is convenient to express the target function with the following equivalent expression(which is just a regrouping of the terms that appear in equa-

tion (4.70), see [36]). We have:

$$L(M) = q_{\curvearrowright} H\left(\mathcal{Q}\right) + \tag{4.77}$$

$$- \sum_{v=1}^{N} p_v \log\left(p_v\right) + \tag{4.78}$$

$$+ \sum_{i=1}^{C} \left(q_{i\curvearrowright} + \sum_{v \in i} p_v \right) \log \left(q_{i\curvearrowright} + \sum_{v \in i} p_v \right) \tag{4.79}$$

Let's start calculating $q_{i\curvearrowright}$, the probability of leaving module C_i. This can happen in two ways, either by crossing an edge that connect two different communities, or by teleportation. The relative probability of the two events happening is controlled by the teleportation parameter τ. Now, the probability of switching module via the standard random walk rule is just α, the community strength parameter, as we defined it as just the proportion of edges that run across community. The chances of teleporting to another module only depend on the number of modules C, as teleportation is assumed uniform to all nodes and all communities are of the same size. Putting everything together, we obtain:

$$q_{i\curvearrowright} = (1 - \tau)\,\alpha + \tau \left(\frac{C-1}{C} \right) \tag{4.80}$$

Next, the sum of the p_v – the ergodic node visit frequency at node v — can be simplified as:

$$\sum_{v \in i} p_v = \sum_{v \in i} \frac{d(v,\tau) + d(v)}{N(d(v,\tau) + d(v))} = N_C \frac{d(\tau) + d}{N(d(\tau) + d)} = \frac{N_C}{N} = 1/C \tag{4.81}$$

in which we have used the graph regularity to simplify $d(v) \mapsto d$, the uniformity of teleportation to simplify $d(v,\tau) \mapsto d(\tau)$, and the fact that all communities have the same size to conclude the last equality.

Let's discuss each term that makes up the target function $L(M)$ separately. Start-

ing from equation (4.77), we can rewrite it as follows [36]:

$$q_{\curvearrowright} \times H\left(\mathcal{Q}\right) = \left[\sum_{i=1}^{C} q_{i\curvearrowright}\right] \times \left[\sum_{i=1}^{C} \frac{q_{i\curvearrowright}}{\sum_{j=1}^{C} q_{j\curvearrowright}} \log\left(\frac{q_{i\curvearrowright}}{\sum_{j=1}^{C} q_{j\curvearrowright}}\right)\right] \tag{4.82}$$

which, together with equation (4.80) shows that the first term in equation (4.77) is constant in N.

Now equation (4.78) is a relatively straightforward analysis due to the hypotheses we have made: p_v is the ergodic node visit frequency, which would be a function of the node degree $d(v)$ and the teleportation parameter τ. In a regular graph, this is just a constant and by normalization we must have $p_v = 1/N$. Hence:

$$-\sum_{v=1}^{N} p_v \log\left(p_v\right) = \log N \tag{4.83}$$

Finally, equation (4.79) also does not depend on N, which can be seen by combining equation (4.80) and equation (4.81). This shows that the InfoMap target function reaches asymptotically the same value regardless of the number of communities in the graph. We can now derive a detectability threshold by studying the dependence of the two correction terms in equation (4.77) and equation (4.79). The correction ΔL will be a function as a function of the number of communities C, the community strength parameter α, and the teleportation parameter τ.

$$\Delta L(C, \alpha, \tau) = q_{\curvearrowright} H\left(\mathcal{Q}\right) + \tag{4.84}$$

$$+ \sum_{i=1}^{C} \left(q_{i\curvearrowright} + \sum_{v \in i} p_v\right) \log\left(q_{i\curvearrowright} + \sum_{v \in i} p_v\right) \tag{4.85}$$

Let's start by calculating $q_{\curvearrowright}$, the probability of switching module altogether at

each step. We will make use of the result in equation (4.80) to obtain:

$$q_{\curvearrowright} = \sum_{i=1}^{C} q_{i\curvearrowright} = (1-\tau)\, C\, \alpha + \tau\, (C-1) \tag{4.86}$$

Calculating $H(\mathcal{Q})$ is relatively straightforward, as we only need to substitute equation (4.80) and equation (4.82). When substituting we obtain:

$$H(\mathcal{Q}) = \sum_{i=1}^{C} \frac{q_{i\curvearrowright}}{\sum_{j=1}^{C} q_{j\curvearrowright}} \log\left(\frac{q_{i\curvearrowright}}{\sum_{j=1}^{C} q_{j\curvearrowright}}\right) = \tag{4.87}$$

$$= \log\left(\frac{q_{i\curvearrowright}}{\sum_{j=1}^{C} q_{j\curvearrowright}}\right) = \log(1/C) = -\log(C) \tag{4.88}$$

So, we obtain the expression for equation (4.84) by putting together equations (4.86) and (4.88). We obtain:

$$q_{\curvearrowright} H\left(\mathcal{Q}\right) = -(1-\tau)\, \alpha\, C \log(C) - \tau(C-1)\log(C) \tag{4.89}$$

Now we have pre-calculated all terms to compute equation (4.85). We obtain:

$$\sum_{i=1}^{C} \left(q_{i\curvearrowright} + \sum_{v\in i} p_v\right) \log\left(q_{i\curvearrowright} + \sum_{v\in i} p_v\right) = \sum_{i=1}^{C} \left(q_{i\curvearrowright} + \frac{1}{C}\right) \log\left(q_{i\curvearrowright} + \frac{1}{C}\right) \tag{4.90}$$

Since we know that $q_{i\curvearrowright}$ is independent of i due to the simplification introduced (see equation (4.80)), we obtain:

$$\sum_{i=1}^{C} \left(q_{i\curvearrowright} + \sum_{v\in i} p_v\right) \log\left(q_{i\curvearrowright} + \sum_{v\in i} p_v\right) = (q_{\curvearrowright} + 1) \log\left(q_{i\curvearrowright} + \frac{1}{C}\right) \tag{4.91}$$

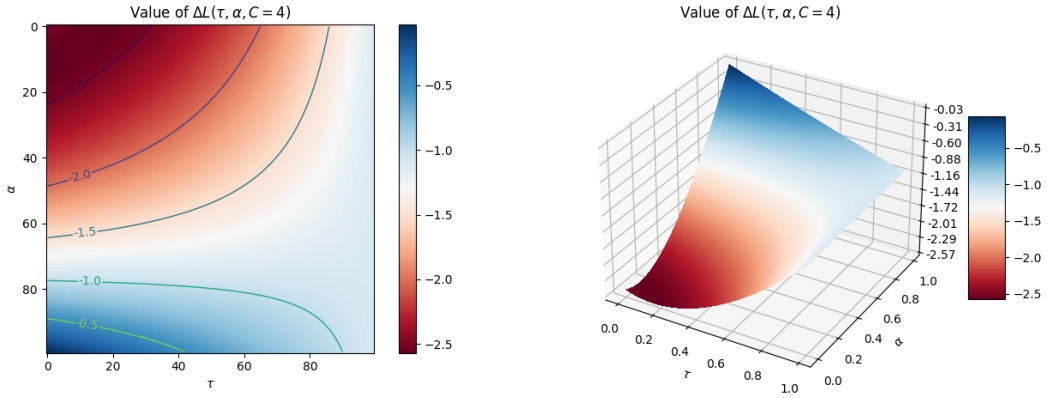
Substituting equation (4.80) we get equation (4.85), which together with equation (4.84)

produces:

$$\Delta L(C, \alpha, \tau) = -\log(C) + \tag{4.92}$$

$$+ \Big((1-\tau)\alpha C + \tau(C-1) + 1\Big) \log\left[(1-\tau)\alpha + \tau\left(\frac{C-1}{C}\right) + \frac{1}{C}\right] \tag{4.93}$$

A plot of the correction term ΔL is provided in figure section 4.4.3



(a) 2D plot with level lines (b) 3D plot

Figure 4.8: Plots of $\Delta L(\tau, \alpha, C)$ as a function of τ, α and with $C = 4$. Note how the correction term is negative but independent on the network size N.

As expected, the correction term ΔL does not depend on N, so the InfoMap target function L asymptotically cannot distinguish the case of one or more communities. Indeed, as the graph size N grows, the correction term ΔL stays the same while L grows logarithmically as $\log(N)$. Hence, we do expect InfoMap to fail even for large, finite graphs.

4.5 Discussion

In this chapter, we have investigated the interplay between communities and random walks on graphs. Specifically, how the former influence the latter, and the ability to use random walks to detect communities in a random graph.

Fristly, we have studied the distribution of the first return times as a way to measure the influence of the community structure on random walks. The average of such distribution is independent of the global community structure, as expected via Kac's lemma. More surprisingly, this is also true for the variance. We obtained a similar result for the skewness and the kurtosis. Hence, the distribution as whole is not significantly affected by the community structure, or at least it is only in its very high momenta. This is quite counter-intuitive given the premises that we discussed in the previous section.

This seems to further corroborate our initial observation that random walks behave similarly on random graphs with and without communities. We have developed the following heuristic explanation for this phenomenon. Specifically, in the limit of large number of nodes, both SMBs and Erdős-Rényi graphs become locally tree-like. Because random walks are a local phenomenon, they aren't able to distinguish between the two, no matter the strength of the planted community structure of the SBM model. This is a quite surprising conjecture: even for those graphs with a community structure so strong that a human could distinguish them from Erdős-Rényi graph by visual inspection, any method based on random walks would fail is the size of the graph is large enough.

We want to be careful in stating this result. The stronger version would read as follows:

Conjecture 1. *Random-walks-based community detection method are a byproduct of finite size effects and cannot retrieve the correct planted community structure in the limit of large random graphs.*

We haven't quite proven that, and indeed there are a few technical details that make proving this claim very hard, if not outright impossible.

For example, even if we proved that all community detection methods based on random walks fail in the large size limit, we would then need to prove that such

failure is due to the nature of the problem rather than a limitation of the algorithm. Other problems relating to the very definition of communities — which are normally ignored in more standard applications — also get in the way. For example, what if there are many possible partitions with the same *min bisection* cost, but they are not correlated. Would it make sense to consider a planted community structure in such case?

Partly because of the delicacy of the question we started off with, we proceeded to show that the InfoMap method fails to retrieve the correct planted community structure in the limit of large random graphs. This is shown in Figure 4.2.

This can be explained by an argument similar to the one above but that doesn't pose the more technical questions of the previous claim. We formulate the argument in the following form:

Proposition 7. *For any finite value of the teleportation parameter τ and for any community strength $\alpha < 1$, in the limit of large graphs, $N \to +\infty$, the InfoMap method fails to retrieve the planted community structure.*

This is supported by the data in figure 4.7, which shows experimental evidence of this phenomenon. We explain this result is as follows. Given the teleportation parameter τ, we expect to look at chains of at most $\frac{1}{\tau}$ steps before teleporting in the network. Hence, using random walks only allows us to measure properties that are local, i.e. that manifest at a scale $\frac{1}{\tau}$ in the network. For a community to be detected, we need to at least leave the community during those steps. If we can't leave the community in $\frac{1}{\tau}$ steps, we would not be able to distinguish it from and Erdős-Rényi graph. This is because the subgraph of a community is itself isomorph to an Erdős-Rényi graph.

The relevant test to measure whether or not the random walk can escape the community is to compare the diameter of the graph with $\frac{1}{\tau}$. As the diameter of the graph scales like the logarithm of the number of nodes, as the graph size increase

the graph diameter will overgrow the average number of steps performed by random walks before teleportation. Hence InfoMap will fail on any large enough graph, and the same must be true for all other random-walk based methods.

This idea is connected to the concept of Markov stability, which measures the persistence of clusters in a network across multiple timescales[80] and mirrors our detectability claim. In this comparison, our teleportation parameter τ acts as a cut-off for the timescale used to analyse random walks used in the definition of Markov stability. Both the teleportation parameter and the time-scale cut-off constrain the scope of the analysis to an increasingly local subset of the graph. Our claim that the random walk teleportation affects the detectability of the communities in the limit of large graphs mirrors the loss in resolution in the work by Delvenne et.al. Both these approaches therefore suggest that the effectiveness of random-walk-based methods is inherently tied to the scale at which the network's structure is probed. This, in return, confirms our intuition to explore hybrid approaches that combine random-walk methods with global structural insights.

Chapter 5

Random Cluster Similarity

Similarity matrices are a broad class of methods that is used to study indirect connections between pairs of nodes that are very close but not directly connected. They have successfully been applied to tasks as diverse as clustering [81], network visualisation [82] and graph matching [83].

They can be thought of as a generalisation of the adjacency matrix A_{ij}. Similarly, as A_{ij} measures the direct connectedness between pairs of nodes, a similarity matrix S_{ij} captures the proximity or similarity between pairs of nodes in the network that are not necessarily adjacent. Sometimes referred to in the data science community as *kernels*, they have become an essential part of the data science toolkit. For many algorithms, the first step is the definition of a similarity matrix that can then be used to solve a task. This strategy is at the core of many methods for link prediction, by identifying pairs of nodes that are very similar yet not directly connected [84], but also for other tasks such as anomaly detection [85]and label propagation [86].

In this chapter, we introduce a new similarity measure for the nodes in the network that is rooted in the global structure of the network and is based on edge percolation (see section 2.6) and the concept of connected components. The fundamental idea is to assign high values of similarity to pairs of nodes that are in the same connected

component after an edge percolation process has taken place.

More formally, consider an undirected, unweighted network. We want to define a similarity matrix S_{ij} using the intuition described above. Using the percolated adjacency matrix $A_{ij}^{(p)}$, in which each element of the matrix A_{ij} is set to 0 with some uniform independent probability p, is not a good option as it only accounts for direct connectivity and misses out on longer range connections between the nodes. Especially when graphs are sparse, this can yield a poor result on the similarity measure. Several approaches have been proposed to bypass this limitation, including the shortest-path distance, the effective resistance [87], but also different functions of the adjacency matrix, for example the absorption n-th power matrix described in section 4.3.1.

Here, instead, we define the random cluster similarity matrix P_{ij}^{q} between two nodes as the probability that they are in the same connected component after the random removal of a fraction $1-p$ of the edges. This intuition is based on the random cluster model, which is itself a generalisation of percolation, where each cluster is weighted by a factor of q. In other words, the parameter q controls the relative probability weight of each configuration taken from the ensemble of the original graph. More explicitly, while p controls the removal of the edges, each resulting instance of the graph is weighted differently when producing the final random cluster similarity measure, according to q. Once the edge removal process terminates, the resulting graph configuration $\mathcal{G}_\omega$ contributes to the average (i.e., the random cluster similarity measure) with a weight of $q^{C(\mathcal{G}_\omega)}$, in which $C(\mathcal{G}_\omega)$ is the number of connected components in the graph. What this means in practice is that for different values of q, different weights are given to different graphs according to the structures of the connected components. For example, when $q = 1$, the random cluster model reduces to a simple percolation, as each instance of the ensemble is weighted equally. When $q \to 0$, graph realisations that have more than one connected component become

vanishingly irrelevant in calculating the random cluster similarity.

The random cluster similarity can be written as a similarity function on the random cluster model, which itself is the analytical continuation of the q-states Potts model. This is because the random cluster similarity is defined as:

$$P_{ij}^{q} = \left[1 + q^{-1}(\mu_q^{-1}(i \leftrightarrow j) - 1)\right]^{-1} = \frac{q\,\mu_q(i \leftrightarrow j)}{1 + (q-1)\mu_q(i \leftrightarrow j)} \tag{5.1}$$

where the quantity $\mu_q(i \leftrightarrow j)$ represents the probability of nodes v_i and v_j being connected in the random cluster model, which is calculated as:

$$\mu_q(i \leftrightarrow j) = 1 - \frac{q}{1-q}\left(\frac{\partial}{\partial L_{ii}} + \frac{\partial}{\partial L_{jj}} - \frac{\partial}{\partial L_{ij}} - \frac{\partial}{\partial L_{ji}}\right) \ln Z_{RC} \tag{5.2}$$

Here, L is the weighted graph Laplacian, $L_{ij} = \sum_k A_{ik}\delta_{ij} - A_{ij}$, and Z_{RC} is the partition function of the random cluster model:

$$Z_{RC} = \sum_{\omega \in \Omega}\left[\prod_{e \in E} p_e^{\omega_e}(1-p_e)^{1-\omega_e}\right] q^{c(\omega)} \tag{5.3}$$

The partition function depends on the Bernoulli variables $\omega_e \in \{0, 1\}$, which are equal to one when the edge e is present (or *free*, in percolation language) and zero when e is absent (or *occupied*). Each realisation $\omega = (\omega_1, \omega_2, \cdots, \omega_{|E|})$, sampled from the binary space $\Omega = \{0, 1\}^{|E|}$ corresponds to a graph in the ensemble of random graphs on the node set $V(\mathcal{G})$. The weight factor $q^{c(\omega)}$ formalises the intuitive behaviour of the random cluster similarity as a function of q described above.

To see how equation (5.2) is obtained from equation (5.3), we recall that the connectivity probability $\mu_q(i \leftrightarrow j)$, can be expressed as:

$$\mu_q(i \leftrightarrow j) = \frac{Z_{RC}^{i \leftrightarrow j}}{Z_{RC}} \tag{5.4}$$

where $Z_{RC}^{i\leftrightarrow j}$ is the restricted partition function summing only over configurations in which i and j are connected. Then, to avoid explicitly summing over configurations we use the derivatives of $Z_{RC}^{i\leftrightarrow j}$ and $\ln Z_{RC}$ with respect to the interaction parameters expressed by the Laplacian $L_{ij} = \sum_k A_{ik}\delta_{ij} - A_{ij}$. The connectivity probability is given by

$$\mu_q(i \leftrightarrow j) = 1 - \frac{q}{1-q}\left(\frac{\partial \ln Z_{RC}}{\partial L_{ii}} + \frac{\partial \ln Z_{RC}}{\partial L_{jj}} - 2\frac{\partial \ln Z_{RC}}{\partial L_{ij}}\right). \tag{5.5}$$

which can be symmetrized as in equation (5.2).

5.1 The similarity matrix

An obvious question that arises from the definition of the random cluster similarity is which pairs of nodes have a high similarity in the network. In other words, what network properties cause higher values of random cluster similarities between certain pairs on nodes. In the numerical analysis performed in this Chapter, we investigate the competing effects of two possible mechanisms:

- On one hand, in the case of modular networks, pairs of nodes in the same community are expected to have, on average, a higher random cluster similarity than pairs that of nodes belonging to different communities. Recall the random cluster similarity can be interpreted as the probability that the nodes in the pair are in the same connected component after the random removal of a fraction p of the edges. Since nodes are more tightly connected within communities, nodes within the same community will have, on average, a higher number of independent paths connecting them, hence a higher probability of a path connecting them surviving the edge percolation removal, hence higher random cluster similarity.

- Secondly, in the case of heterogeneous networks, and scale-free networks in particular, nodes with a higher degree are expected to possess higher random cluster similarity. This is caused by the fact that during the edge percolation procedure, each edge is removed with fixed probability p, so nodes that have higher degree at the beginning of the process will, on average, retain their advantage at the end. In addition, high degree nodes in random networks also tend to have more k-neighbours, which increases their probability of belonging to large clusters, and hence to have a high similarity to many other nodes (especially high-degree ones, for the very same reason). Overall, this means that highly connected nodes will retain a higher number of connections with a higher number of nodes. Thus, on average, highly connected nodes will be more likely to belong to the same connected component with respect to more nodes in the network, in turns yielding higher random cluster similarity.

In the following, our objective is to quantify the relative importance of the two phenomena on the random cluster similarity. In other words, both pairs of nodes with high degree (which we will refer to as pair of *nodes with the "degree property"*) and nodes belonging to the same community as pair of (*nodes with the "community property"*) are expected to have higher-than-average random cluster similarity. But which of the two properties has the strongest influence? Note that, in principle, the answer to this question is a function of the two random cluster similarity parameters p, q.

In order to study the relative influence of the degree property and the community property on the random cluster similarity, we proceed as follows. We calculate the random cluster similarity for a number N_S of sample pairs of the nodes in the graph. We then order them and take only the ones in the top $\mathcal{P}$ percentile. For each value of $\mathcal{P}$, we consider all pairs of nodes whose random cluster similarity is above the threshold and we check whether the two nodes v_1, v_2 have the same community

assignment $C(v_1) = C(v_2)$ and whether their degree is higher than the average of the graph $d(v_1) + d(v_2) \geq 2\langle d \rangle$. We then define the following two sets, $\mathcal{C}$ is the set of all pairs that satisfy the community condition:

$$\mathcal{C} = \{(v_1, v_2) \text{ s.t. } C(v_1) = C(v_2),\ RCM(v_1, v_2) \geq P_{100-\mathcal{P}}\} \tag{5.6}$$

and $\mathcal{D}$ is the sets of all pairs that satisfy the degree condition:

$$\mathcal{D} = \{(v_1, v_2) \text{ s.t. } d(v_1) + d(v_2) \geq 2\langle d \rangle,\ RCM(v_1, v_2) \geq P_{100-\mathcal{P}}\} \tag{5.7}$$

We remark that the two sets $\mathcal{C}$ and $\mathcal{D}$ both depend on the random cluster similarity and the choice of the percentile $\mathcal{P}$, so they are functions of the random cluster similarity parameters p, q as well as $\mathcal{P}$. In a scale-free network, the degree property will be satisfied whenever the pair of nodes (v_1, v_2) is made up by at least one hub node, and it is not satisfied only when both nodes are "leaf" nodes, i.e. both have a lower-than-average degree. This is a direct result of the power law degree distribution $P(d) \propto d^{-\gamma}$, for which:

$$\langle d \rangle = \sum_{d=0}^{\infty} kP(k) = \frac{1}{\gamma - 2} \tag{5.8}$$

and in which the degree of hub nodes d_{hub} is often several times larger than $\langle d \rangle$. More explicitly, hub-hub pairs and hub-leaf pairs[1] will be in the set $\mathcal{D}$, while the leaf-leaf pairs will not. This heuristic observation will become useful in the later discussion of the results, see section 5.2.

We then classify each node pair (v_1, v_2) depending on whether it belongs to the first set $\mathcal{C}$, the second set $\mathcal{D}$, both sets or neither. We define the two quantities A and B as the number of pairs of nodes that belong to exactly one of the two sets, $\mathcal{C}$ and

[1]Here "leaf" is used in a loose sense, meaning *a node with degree lower than the average of the power law distribution* $\langle d \rangle$ rather than in the traditional sense of node with unitary degree.

$\mathcal{D}$, respectively.

$$A(p,q) = |\mathcal{D} \setminus \mathcal{C}| \qquad B(p,q) = |\mathcal{C} \setminus \mathcal{D}| \tag{5.9}$$

The idea here is to study which property contribute more strongly to a higher random cluster similarity. If two nodes are in either $\mathcal{C}$ or $\mathcal{D}$, they have a higher-than-average random cluster similarity, but if they are in both sets this may be due to either them being in the same community or them having a higher-than-average degree. So, we only compare the number of pairs for which exactly one of the two property is true, which is the meaning of A and B. More explicitly, A is the number of pairs of nodes that have random cluster similarity in the top $\mathcal{P}$ percentile, have a higher-than-average joint degree but are not in the same community. On the other hand, B is the number of pairs of nodes that have random cluster similarity in the top $\mathcal{P}$ percentile, have the same community assignment but have smaller than average joint degree.

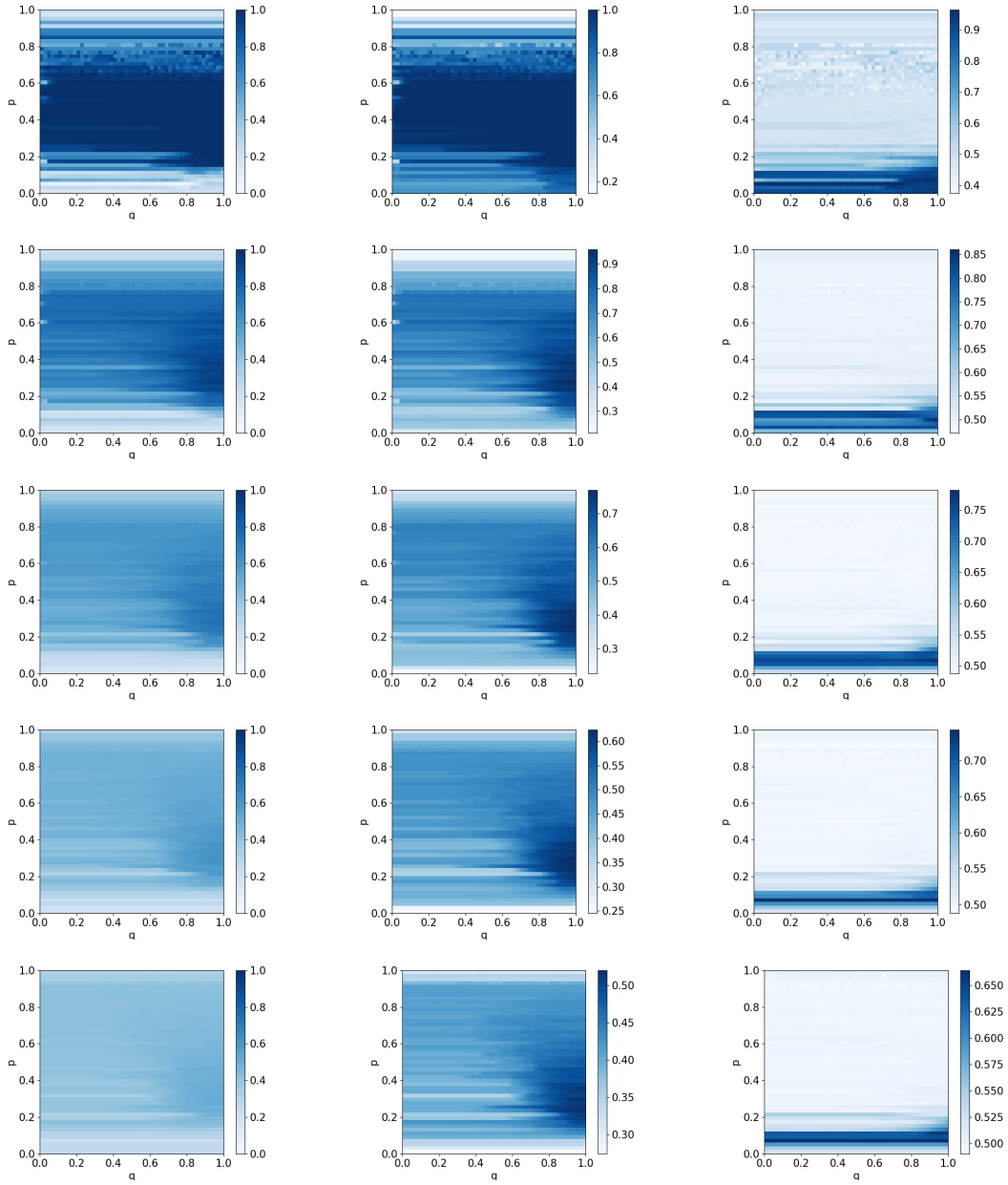


Figure 5.1: Similarity matrix study for a degree-corrected SBM with power law degree distribution. The graph has been generated the parameters $N = 200$ and $\gamma = 3$, and has two communities of equal size. We report (by column) the value of $A(p, q)$, the share of pairs in the upper $\mathcal{P}$ percentile having the degree and community property (that is, $\frac{|\mathcal{C}|}{|\mathcal{P}|}$. Different values for the percentile $\mathcal{P}$ are reported (by row).

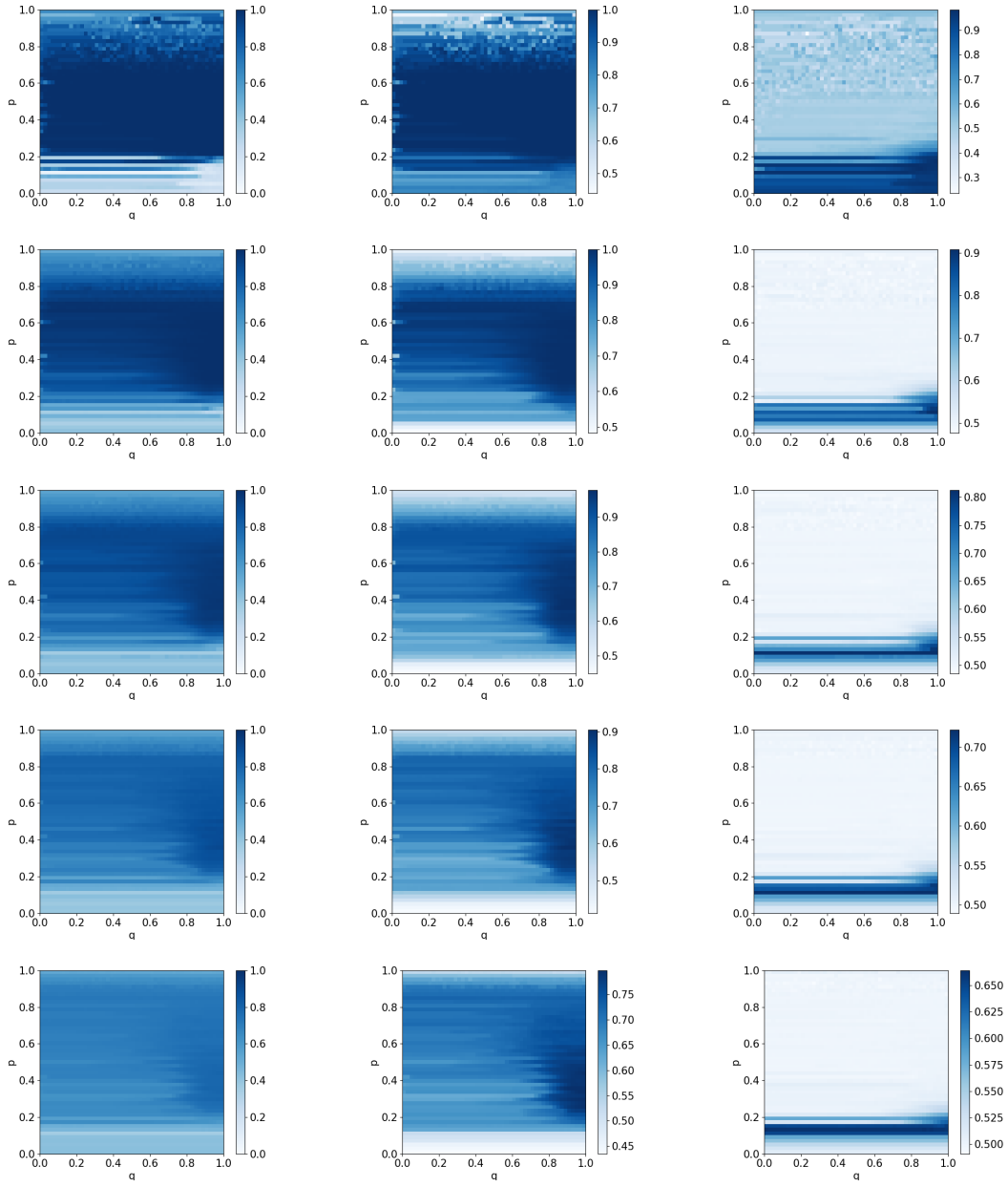


Figure 5.2: Similarity matrix study for a degree-corrected SBM with power law degree distribution. The graph has been generated using the parameters $N = 300$ and $\gamma = 3$, and has two communities of equal size. We report (by column) the value of $A(p, q)$, the share of pairs in the upper $\mathcal{P}$ percentile having the degree and community property (that is, $\frac{|\mathcal{C}|}{|\mathcal{P}|}$. Different values for the percentile $\mathcal{P}$ are reported (by row).

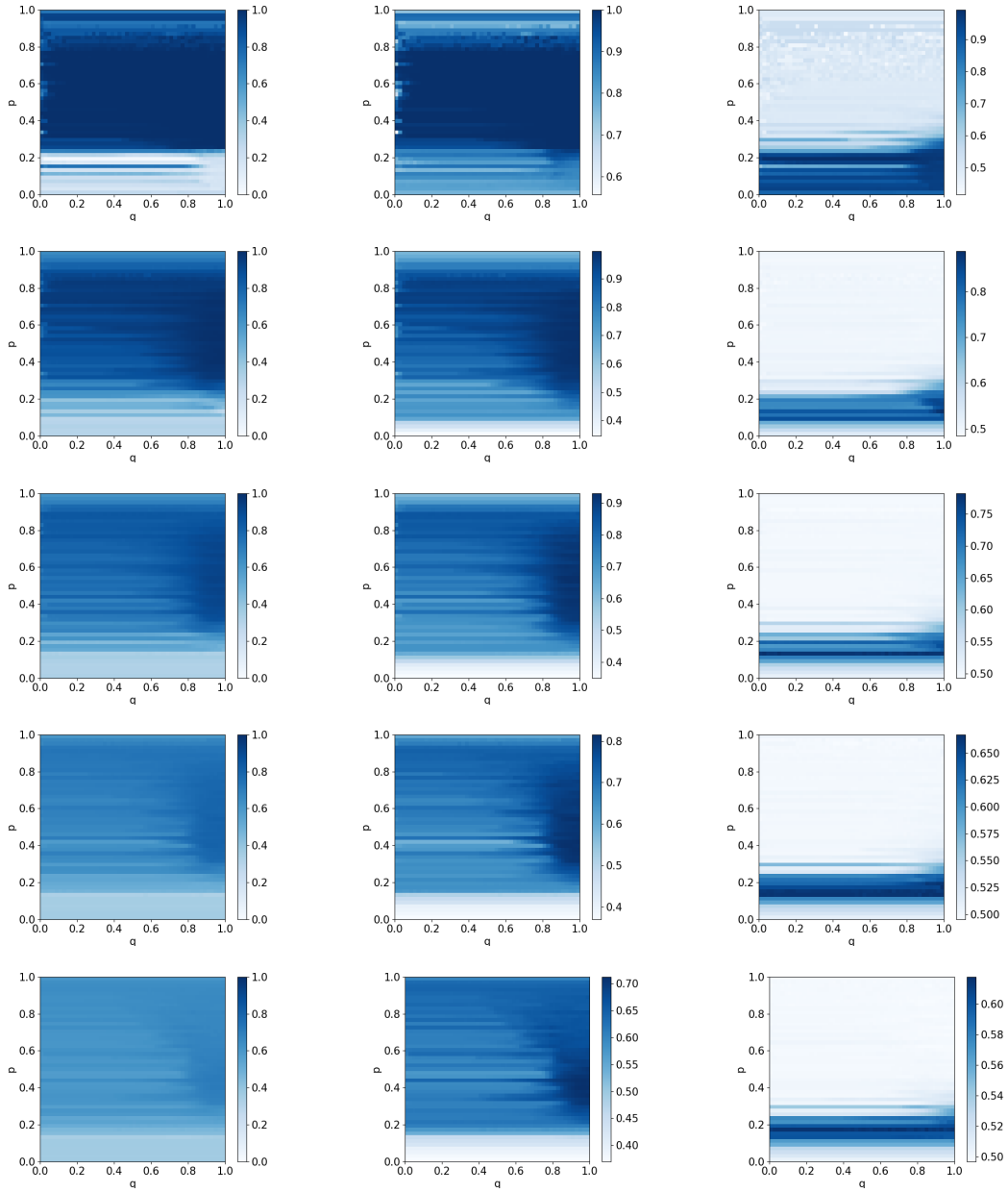


Figure 5.3: Similarity matrix study for a degree-corrected SBM with power law degree distribution. The graph has been generated using the parameters $N = 400$ and $\gamma = 3$, and has two communities of equal size. We report (by column) the value of $A(p, q)$, the share of pairs in the upper $\mathcal{P}$ percentile having the degree and community property (that is, $\frac{|\mathcal{C}|}{|\mathcal{P}|}$. Different values for the percentile $\mathcal{P}$ are reported (by row).

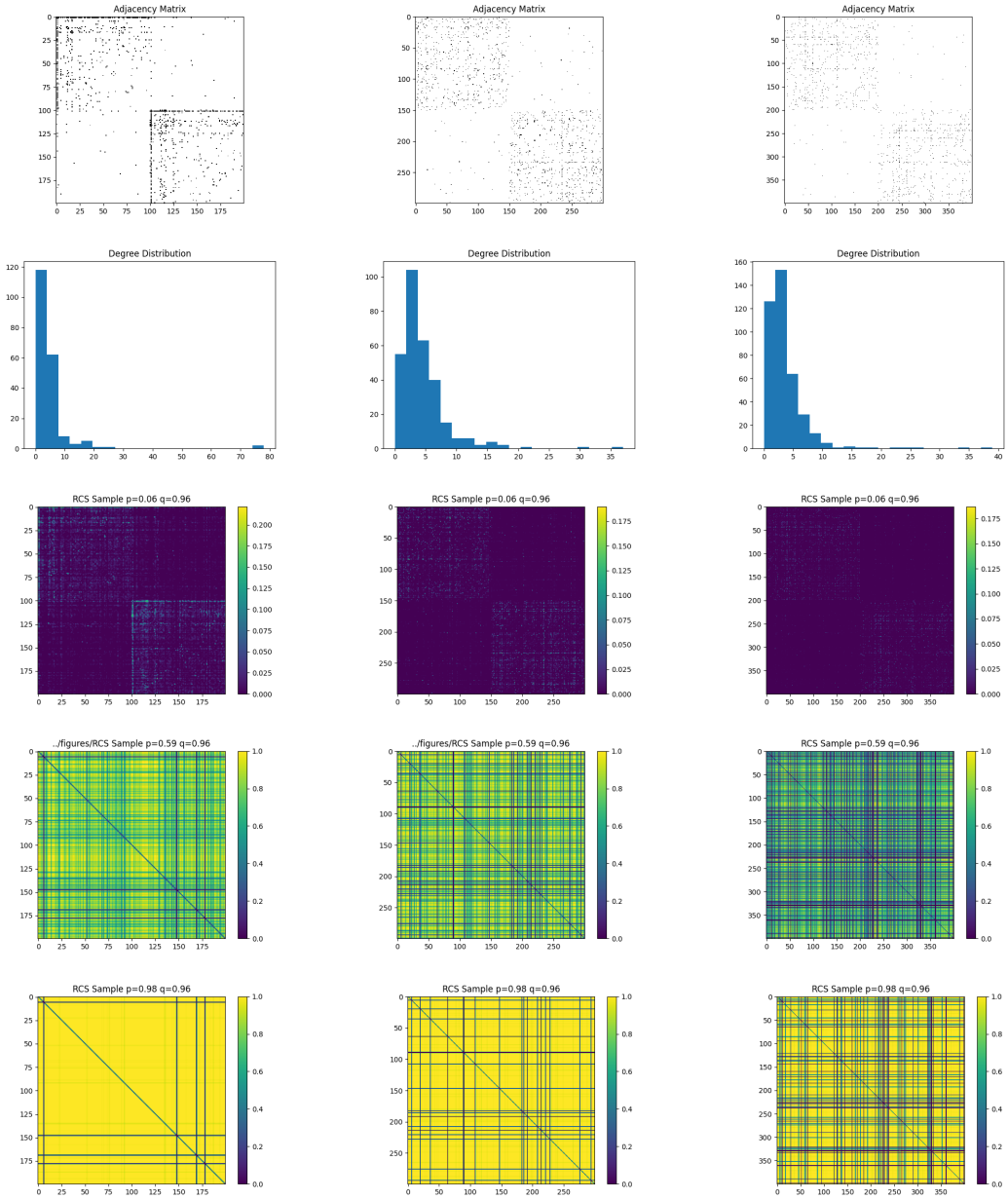


Figure 5.4: Analysis of the three graphs studied in figures 5.1 to 5.3, By row, adjacency matrix, degree distribution, and three samples of the similarity matrix from different values of decreasing p at fixed q.

5.2 Discussion

In the regime of low p, the community property dominates as the primary driver for pairs with high random cluster similarity. This can be explained by the behaviour of the random cluster similarity. The parameter p controls how many edges, on average, are left in the graph after percolation (as a proportion of the total number of edges originally present in the graph). For low p, only a few edges are left in the network, so the probability of nodes that already started with a low degree to be left with any connection at all is vanishingly small. Hence, the remaining edges are prevalently connecting node pairs of the hub-hub or hub-leaf type. We therefore expect the number of pairs in the upper $\mathcal{P}$ percentile with the degree property to become more and more prevalent with respect to the number of pairs having the community property.

In the high p regime, the graph is left untouched (that is, very few edges are occupied in the percolation process) and, in the limit of $p \to 1$, all pairs of nodes have random cluster similarity identically equal to 1. Hence, whatever the choice of the percentile $\mathcal{P}$, taking the pairs with highest random cluster similarity is equivalent to simply performing a random selection of pairs of nodes in the graph. Because leaf nodes make up the majority of nodes, the probability of extracting nodes that satisfy the degree condition of the set $\mathcal{D}$ decreases. More specifically, if we indicate with N_h the number of hubs (nodes with high-than-average degree) and with N_l the number of leaves, the share of leaf-leaf nodes extracted by the percentile procedure will be proportional to the probability of randomly extracting two leaf nodes, that is:

$$\lim_{p\to 1} A(p,q) = \mathbb{P}(v_1 \text{ leaf}, v_2 \text{ leaf}) \approx \mathbb{P}\big(d(v_1) < \langle d\rangle\big)\mathbb{P}\big(d(v_2) < \langle d\rangle\big) \tag{5.10}$$

in which the approximation is due to the fact that the choice of the two nodes is not strictly independent, because we cannot choose the same node twice. However, the

error committed in the approximation is negligible in the limit of large N, scaling down as $O(1/N)$. Under this approximation, we can calculate the limit exactly in expectation as a function of the degree distribution power law parameter γ only:

$$\mathbb{P}\big(d(v) < \langle d \rangle\big) \approx 1 - \mathbb{P}\big(d(v_1) > \langle d \rangle\big) = 1 - \frac{\langle d \rangle^{1-\gamma}}{\gamma - 1} \tag{5.11}$$

in which the approximation is exact as long as $\langle d \rangle$ is non-integer, which happens for all choices of $\gamma \neq 2 + \frac{1}{n}$, for all $n \in \mathbb{N}$. Plugging section 5.2 into equation (5.10) yields:

$$\lim_{p \to 1} A(p, q) = \left[1 - \frac{\langle d \rangle^{1-\gamma}}{\gamma - 1}\right]^2 + O\left(\frac{1}{N}\right) \tag{5.12}$$

uniformly in q.

Chapter 6

Conclusion and outlook

The aim of this final chapter is twofold. Firstly, it contains a brief overview of the results proposed in this thesis. Secondly – and perhaps more importantly – it collates such results and presents a unified forward look into how they might be further developed and improved. While the projects are fairly independent, they all gravitate on the same investigation of the relationship between the structural properties and the dynamical quantities of complex networks, and here we highlight their commonalities and how their developments might interconnect.

Finally, we have collected a series of open questions about the topics we have covered that we believe are particularly interesting and worth investigating.

- The Ergodic Sets Compression Algorithm (ESCA) that we introduced in chapter 3 is a simple yet powerful tool to reduce the dimensionality of directed networks while preserving the long-term dynamical properties of the network. The first step of the algorithm relies on the coarse-graining of the ergodic sets to prepare for the SVD compression of the graph core. In principle, there is nothing stopping us from applying a SVD compression directly to the entire graph. This would mean that we'd lose out on the black-box structure shown in figure 3.9 and the dynamical factorisation in three distinct regimes separated

by a hitting time, as seen in equation (3.23). It would be beneficial to investigate what degree of performance deterioration – if any at all – would be caused by skipping the first ESCA step (in other words, how much more effective is the ESCA algorithm in comparison to a direct SVD compression of the whole network).

- A related question involves a modified version of the ESCA algorithm leading to a higher yet lossy graph compression. The presentation of the algorithm focused on the lossless compression, in which the rank of the compressed mixing matrix is $k = \min(N_{bw}, N_{fw})$. Further investigation would focus on reducing the rank of the singular value decomposition of the mixing matrix k in order to improve the network compression at the cost of an error in the estimation of the infinite-time transition probabilities. It would be worth investigating how far this lossy compression could be taken and the benefits and drawbacks of such extension of the ESCA.

- Collectively, proposition 3, proposition 4, and proposition 5 create a hierarchy of structures defined on directed networks based of the random walk dynamics. Other similar hierarchies have been introduced on complex networks, such as trophic levels [88] and communities [28]. While we do not expect different hierarchical structures to be encapsulated across different hierarchies, it could be interested to explore their relation in expectation.

- the fast nature of the compression algorithm makes it a suitable basis for a heuristic approach that is commonplace in numerical simulations, particularly in numerical analysis. The compression factor can be controlled by a single

parameter n, which is the total number of principal SVD values retained in the second step of the ESCA algorithm. We want to look into an iterative approach that maximises the network compression while keeping the information loss under a specified threshold. Idea is to first step to first compress the network and measure the resulting loss with the results in section 3.6, then if the precision level is high enough, iterate the first step with a smaller set of singular values by decreasing n, if instead if the lossy compression leads to a precision loss higher than the threshold.

- A natural development of the results in chapter 4 for this project is to better formalise conjecture 1. However, it is not obvious how to do so. It is not clear how to properly distinguish the intrinsic error that the algorithm performs (due to the convergence to locally optimal community partitions) and the claimed error caused by the finite size effect. The basis of such proof could be to "suggest" to InfoMap the ground truth partition and show that the algorithm is still unable to converge to a community structure correlated to the correct one. We would need further investigations on whether this idea is viable and effective.

- A possible extension of the result in equation (5.12) would be a similar closed form for the similarity matrix in the q parameter, uniformly in p. This would involve a more "statistical mechanics" approach, as the roles of p and q are fundamentally distinct. While averaging over the parameter p involves a simple resampling over the uniform distribution of picking the two nodes v_1 and v_2, sampling over q involves the canonical sampling over the Boltzmann distribution that generates the partition function in equation (5.3). While this analysis would depart from network science strictly interpreted, it would be useful to

improve our understanding of the asymptotic behaviour of the random cluster similarity.

- In chapter 5 we have focused on the random cluster similarity as defined on the classical percolation, that is, a percolation process in which each edge is non-occupied with the same, independent probability p. A natural extension would be to study the random cluster similarity behaviour under a broader family of percolating phenomena. This would allow to extend the basic definitions to a much broader class of phenomena, highlighting different graph structures with different biased edge removals.

- Similarly, further generalisations could involve repeating the analysis of the random cluster similarity based on a random cluster models on graphs with the scale-free property and exponential degree distribution, to align the prediction with real-world networks, like social networks, internet infrastructure, or biological networks.

To conclude, this thesis contains a broad overview on different ways the interplay between the network structure and the network dynamics can be exploited to obtain useful information on the analysis of complex networks. We hope that this work shall be useful to anyone working on adjacent questions and that it may foster further research into this fascinating topic.

Bibliography

[1] "Opte project." https://www.opte.org/the-internet. Accessed: 2024-08-11. iii, 4

[2] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener, "Graph structure in the web," *Computer Networks*, vol. 33, p. 309–320, 2000. iii, 46, 74

[3] N. Papanikolaou, G. Vaccario, E. Hormann, R. Lambiotte, and F. Schweitzer, "Consensus from group interactions: An adaptive voter model on hypergraphs," *Physical Review E*, vol. 105, no. 5, p. 054307, 2022. 2

[4] R. Pastor-Satorras and A. Vespignani, "Epidemic spreading in scale-free networks," *Physical review letters*, vol. 86, no. 14, p. 3200, 2001. 7

[5] P. Pons and M. Latapy, "Computing communities in large networks using random walks," in *Computer and Information Sciences-ISCIS 2005: 20th International Symposium, Istanbul, Turkey, October 26-28, 2005. Proceedings 20*, pp. 284–293, Springer, 2005. 7

[6] S. Janson, T. Luczak, and A. Rucinski, *Random graphs*. John Wiley & Sons, 2011. 13

[7] K. Pearson, "The problem of the random walk," *Nature*, vol. 72, no. 1867, pp. 342–342, 1905. 17

[8] R. N. Mantegna and H. E. Stanley, "Scaling behaviour in the dynamics of an economic index," *Nature*, vol. 376, no. 6535, pp. 46–49, 1995. 17

[9] J.-P. Bouchaud and M. Potters, *Theory of financial risk and derivative pricing: from statistical physics to risk management*. Cambridge university press, 2003. 17

[10] A. Einstein *et al.*, "On the motion of small particles suspended in liquids at rest required by the molecular-kinetic theory of heat," *Annalen der physik*, vol. 17, no. 549-560, p. 208, 1905. 17, 42

[11] M. Kimura, "Diffusion models in population genetics," *Journal of Applied Probability*, vol. 1, no. 2, pp. 177–232, 1964. 17, 42

[12] R. Motwani and P. Raghavan, "Randomized algorithms," *ACM Computing Surveys (CSUR)*, vol. 28, no. 1, pp. 33–37, 1996. 17

[13] K. Avrachenkov, D. Nemirovsky, and K. S. Pham, "A survey on distributed approaches to graph based reputation measures," in *1st International ICST Workshop on Tools for solving Structured Markov Chains*, 2010. 17

[14] F. Göbel and A. Jagers, "Random walks on graphs," *Stochastic processes and their applications*, vol. 2, no. 4, pp. 311–336, 1974. 18

[15] R. Burioni and D. Cassi, "Random walks on graphs: ideas, techniques and results," *Journal of Physics A: Mathematical and General*, vol. 38, no. 8, p. R45, 2005. 18

[16] J. R. Norris, *Markov chains*. Cambridge series in statistical and probabilistic mathematics, Cambridge University Press, 1998. 18, 22, 24, 64

[17] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," *Computer Networks and ISDN Systems*, vol. 30, no. 1, pp. 107 – 117,

1998. Proceedings of the Seventh International World Wide Web Conference. 22, 42

[18] W. Feller, *An introduction to probability theory and its applications, Volume 2*, vol. 81. John Wiley & Sons, 1991. 24

[19] N. Biggs, N. L. Biggs, and B. Norman, *Algebraic graph theory*. Cambridge university press, 1993. 26

[20] A.-L. Barabási, "Network science," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 371, no. 1987, p. 20120375, 2013. 26

[21] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE signal processing magazine*, vol. 30, no. 3, pp. 83–98, 2013. 27

[22] M. E. Newman, "Modularity and community structure in networks," *Proceedings of the national academy of sciences*, vol. 103, no. 23, pp. 8577–8582, 2006. 27, 30, 32

[23] R. Kondor and J.-P. Vert, "Diffusion kernels," *kernel methods in computational biology*, pp. 171–192, 2004. 27

[24] D. K. Hammond, Y. Gur, and C. R. Johnson, "Graph diffusion distance: A difference measure for weighted graphs based on the graph laplacian exponential kernel," in *2013 IEEE global conference on signal and information processing*, pp. 419–422, IEEE, 2013. 27

[25] D. A. Spielman and S.-H. Teng, "Spectral partitioning works: Planar graphs and finite element meshes," in *Proceedings of 37th conference on foundations of computer science*, pp. 96–105, IEEE, 1996. 27

[26] G. Strang, *Introduction to linear algebra*. SIAM, 2022. 28, 29

[27] M. T. Schaub, J.-C. Delvenne, M. Rosvall, and R. Lambiotte, "The many facets of community detection in complex networks," *Applied network science*, vol. 2, no. 1, pp. 1–13, 2017. 29

[28] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of statistical mechanics: theory and experiment*, vol. 2008, no. 10, p. P10008, 2008. 30, 32, 61, 83, 87, 144

[29] E. Abbe, A. S. Bandeira, and G. Hall, "Exact recovery in the stochastic block model," *IEEE Transactions on information theory*, vol. 62, no. 1, pp. 471–487, 2015. 32

[30] E. Abbe and C. Sandon, "Community detection in general stochastic block models: Fundamental limits and efficient algorithms for recovery," *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pp. 670–688, 2015. 32

[31] U. N. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, vol. 76, no. 3, p. 036106, 2007. 32

[32] A. Ng, M. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," *Advances in neural information processing systems*, vol. 14, 2001. 32

[33] I. Derényi, G. Palla, and T. Vicsek, "Clique percolation in random networks," *Physical review letters*, vol. 94, no. 16, p. 160202, 2005. 32

[34] M. Rosvall, D. Axelsson, and C. T. Bergstrom, "The map equation," *The European Physical Journal Special Topics*, vol. 178, no. 1, pp. 13–23, 2009. 33, 43, 89

[35] M. Rosvall and C. T. Bergstrom, "Multilevel compression of random walks on networks reveals hierarchical organization in large integrated systems," *PloS one*, vol. 6, no. 4, p. e18209, 2011. 33

[36] M. Rosvall and C. T. Bergstrom, "Maps of random walks on complex networks reveal community structure," *Proceedings of the National Academy of Sciences*, vol. 105, no. 4, pp. 1118–1123, 2008. 33, 42, 89, 121, 123, 124

[37] Y. Yao, "Information-theoretic measures for knowledge discovery and data mining," *Entropy measures, maximum entropy principle and emerging applications*, pp. 115–136, 2003. 34

[38] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," *Computer Networks and ISDN Systems*, vol. 30, no. 1-7, pp. 107–117, 1998. 35

[39] M. Mézard and A. Montanari, *Information, Physics, and Computation*. Oxford University Press, 2009. 36, 38, 97

[40] K. Murphy, Y. Weiss, and M. I. Jordan, "Loopy belief propagation for approximate inference: An empirical study," *arXiv preprint arXiv:1301.6725*, 2013. 38

[41] W. H. Stockmayer, "Theory of molecular size distribution and gel formation in branched-chain polymers," *The Journal of chemical physics*, vol. 11, no. 2, pp. 45–55, 1943. 39

[42] B. Bollobás, "The evolution of random graphs," *Transactions of the American Mathematical Society*, vol. 286, no. 1, pp. 257–274, 1984. 40

[43] M. Piraveenan, M. Prokopenko, and L. Hossain, "Percolation centrality: Quantifying graph-theoretic impact of nodes during percolation in networks," *PloS one*, vol. 8, no. 1, p. e53095, 2013. 40, 41

[44] F. Black and M. Scholes, "The pricing of options and corporate liabilities," *Journal of political economy*, vol. 81, no. 3, pp. 637–654, 1973. 42

[45] R. M. Anderson and R. M. May, *Infectious diseases of humans: dynamics and control*. Oxford university press, 1991. 42

[46] E. A. Leicht and M. E. Newman, "Community structure in directed networks," *Physical review letters*, vol. 100, no. 11, p. 118703, 2008. 42

[47] F. Bloch, M. O. Jackson, and P. Tebaldi, "Centrality measures in networks," *Available at SSRN 2749124*, 2019. 42

[48] K. Berahmand, E. Nasiri, S. Forouzandeh, and Y. Li, "A preference random walk algorithm for link prediction through mutual influence nodes in complex networks," *Journal of king saud university-computer and information sciences*, vol. 34, no. 8, pp. 5375–5387, 2022. 42

[49] R.-H. Li, J. X. Yu, L. Qin, R. Mao, and T. Jin, "On random walk based graph sampling," in *2015 IEEE 31st international conference on data engineering*, pp. 927–938, IEEE, 2015. 42

[50] M. Bianchini, M. Gori, and F. Scarselli, "Inside pagerank," *ACM Transactions on Internet Technology (TOIT)*, vol. 5, no. 1, pp. 92–128, 2005. 42

[51] D. F. Gleich, "Pagerank beyond the web," *Siam Review*, vol. 57, no. 3, pp. 321–363, 2015. 42

[52] N. Rodgers, P. Tiňo, and S. Johnson, "Strong connectivity in real directed networks," *Proceedings of the National Academy of Sciences*, vol. 120, no. 12, p. e2215752120, 2023. 44

[53] S. Fortunato, "Community detection in graphs," *Physics reports*, vol. 486, no. 3-5, pp. 75–174, 2010. 44

[54] R. S. MacKay, S. Johnson, and B. Sansom, "How directed is a directed network?," *Royal Society open science*, vol. 7, no. 9, p. 201138, 2020. 44

[55] D. Pauly, A. Trites, E. Capuli, and V. Christensen, "Diet composition and trophic levels of marine mammals," *ICES journal of Marine Science*, vol. 55, no. 3, pp. 467–481, 1998. 44

[56] S. Pimm and J. Lawton, "Number of trophic levels in ecological communities," *Nature*, vol. 268, no. 5618, pp. 329–331, 1977. 44

[57] C. De Bacco, D. B. Larremore, and C. Moore, "A physical model for efficient ranking in networks," *Science advances*, vol. 4, no. 7, p. eaar8260, 2018. 44

[58] R. Tarjan, "Depth-first search and linear graph algorithms," *SIAM journal on computing*, vol. 1, no. 2, pp. 146–160, 1972. 58

[59] E. Nuutila and E. Soisalon-Soininen, "On finding the strongly connected components in a directed graph," *Information processing letters*, vol. 49, no. 1, pp. 9–14, 1994. 58

[60] C. Eckart and G. Young, "The approximation of one matrix by another of lower rank," *Psychometrika*, vol. 1, no. 3, pp. 211–218, 1936. 65

[61] F.-Y. Wang and D. Liu, "Networked control systems," *Theory and Applications*, 2008. 70

[62] Y. Tipsuwan and M.-Y. Chow, "Control methodologies in networked control systems," *Control engineering practice*, vol. 11, no. 10, pp. 1099–1111, 2003. 70

[63] J. Mourier, S. Planes, and N. Buray, "Trophic interactions at the top of the coral reef food chain," *Coral Reefs*, vol. 32, no. 1, pp. 285–285, 2013. 72

[64] D. Teodorović, *Transportation networks: a quantitative treatment*. Routledge, 1986. 72

[65] B. Alberts, "The cell as a collection of protein machines: preparing the next generation of molecular biologists," *cell*, vol. 92, no. 3, pp. 291–294, 1998. 72

[66] R. Guimera, S. Mossa, A. Turtschi, and L. N. Amaral, "The worldwide air transportation network: Anomalous centrality, community structure, and cities' global roles," *Proceedings of the National Academy of Sciences*, vol. 102, no. 22, pp. 7794–7799, 2005. 72

[67] B. A. Huberman and L. A. Adamic, "Growth dynamics of the world-wide web," *Nature*, vol. 401, no. 6749, pp. 131–131, 1999. 72

[68] V. A. Traag, L. Waltman, and N. J. Van Eck, "From louvain to leiden: guaranteeing well-connected communities," *Scientific reports*, vol. 9, no. 1, p. 5233, 2019. 83, 87

[69] B. D. Hughes and M. Sahimi, "Random walks on the Bethe lattice," *Journal of Statistical Physics*, vol. 29, pp. 781–794, 1982. 98

[70] G. Polya, "Bemerkung über die mittag-lefflerschen funktionen ea (z)," *Tohoku Mathematical Journal, First Series*, vol. 19, pp. 241–248, 1921. 98

[71] G. T. Cantwell and M. E. J. Newman, "Message passing on networks with loops," *Proceedings of the National Academy of Sciences*, vol. 116, no. 47, pp. 23398–23403, 2019. 100

[72] G. T. Cantwell, A. Kirkley, and F. Radicchi, "Heterogeneous message passing for heterogeneous networks," *Physical Review E*, vol. 108, no. 3, p. 034310, 2023. 100

[73] R. J. Gallagher, J.-G. Young, and B. F. Welles, "A clarified typology of core-periphery structure in networks," *Science Advances*, vol. 7, no. 12, p. eabc9800, 2021. 109

[74] M. Newman, *Networks*. Oxford university press, 2018. 109

[75] A. Decelle, F. Krzakala, C. Moore, and L. Zdeborová, "Inference and phase transitions in the detection of modules in sparse networks," *Physical Review Letters*, vol. 107, no. 6, p. 065701, 2011. 109

[76] T. P. Peixoto, "The graph-tool python library," *figshare*, 2014. 110

[77] P. W. Holland, K. B. Laskey, and S. Leinhardt, "Stochastic blockmodels: First steps," *Social Networks*, vol. 5, no. 2, pp. 109–137, 1983. 112

[78] L. Danon, A. Diaz-Guilera, J. Duch, and A. Arenas, "Comparing community structure identification," *Journal of statistical mechanics: Theory and experiment*, vol. 2005, no. 09, p. P09008, 2005. 119

[79] A. Mahmoudi and D. Jemielniak, "Proof of biased behavior of normalized mutual information," *Scientific Reports*, vol. 14, no. 1, p. 9021, 2024. 119

[80] J.-C. Delvenne, S. N. Yaliraki, and M. Barahona, "Stability of graph communities across time scales," *Proceedings of the national academy of sciences*, vol. 107, no. 29, pp. 12755–12760, 2010. 129

[81] A. Fritsch and K. Ickstadt, "Improved criteria for clustering based on the posterior similarity matrix," 2009. 130

[82] S. Parveen and J. Sreevalsan-Nair, "Visualization of small world networks using similarity matrices," in *International Conference on Big Data Analytics*, pp. 151–170, Springer, 2013. 130

[83] L. A. Zager and G. C. Verghese, "Graph similarity scoring and matching," *Applied mathematics letters*, vol. 21, no. 1, pp. 86–94, 2008. 130

[84] D. Liben-Nowell and J. Kleinberg, "The link prediction problem for social networks," in *Proceedings of the twelfth international conference on Information and knowledge management*, pp. 556–559, 2003. 130

[85] L. Akoglu, M. McGlohon, and C. Faloutsos, "Oddball: Spotting anomalies in weighted graphs," in *Advances in Knowledge Discovery and Data Mining: 14th Pacific-Asia Conference, PAKDD 2010, Hyderabad, India, June 21-24, 2010. Proceedings. Part II 14*, pp. 410–421, Springer, 2010. 130

[86] Z.-W. Zhang, X.-Y. Jing, and T.-J. Wang, "Label propagation based semi-supervised learning for software defect prediction," *Automated Software Engineering*, vol. 24, pp. 47–69, 2017. 130

[87] K. Devriendt, "Effective resistance is more than distance: Laplacians, simplices and the schur complement," *Linear Algebra and its Applications*, vol. 639, pp. 24–49, 2022. 131

[88] 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*, vol. 69, no. 1, pp. 1–15, 2000. 144