Topology of Complex Networks: Models and Analysis

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

by

Corrie Jacobien Carstens
Bachelor of Science, Master of Science

School of Mathematical and Geospatial Sciences
College of Science, Engineering and Health
RMIT University

January 2016

RMIT
UNIVERSITY

Declaration

I certify that except where due acknowledgement has been made, the work is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; the content of the thesis/project is the result of work which has been carried out since the official commencement date of the approved research program; any editorial work, paid or unpaid, carried out by a third party is acknowledged; and, ethics procedures and guidelines have been followed.

Corrie Jacobien Carstens

11 January 2016

Dedication

To my parents Paul en Afke, who always encourage me, no matter what I do or where I go. Their optimism and support have allowed me to chase my dreams.

Acknowledgements

I am very grateful to my supervisor Professor Kathy Horadam, who has encouraged and guided me, and who has provided me with invaluable feedback throughout this project. Her energy and enthusiasm are an inspiration to me. I would also like to thank the additional members of my supervising team: Associate Professor Asha Rao for listening to every milestone talk twice, Dr. Miro Kraetzl for his confidence in my capacity, and Darren Boulton for his diligent review of my work.

I have been very lucky to meet, gain advice from and work with several inspiring people throughout this project. In particular I would like to express my gratitude to Assistant Professor Craig Westerland, Dr. Dallas Warren, Dr. Vanessa Robins, Dr. Stephen Davis, Professor Lewi Stone, Dr. Giovanni Strona and Dr. Annabell Berger. I would like to thank the members of RMIT's Complex Networks Journal Club for many interesting discussions and the members of our school's Writing Group for their detailed feedback and collegial support.

Thanks to my friends and family, near and far, for both distracting and encouraging me. Thanks to our dog Freda for strictly enforcing outdoor time and the happiness she brings. Finally I want to thank my dear husband Mike, for encouraging me to undertake this research, for moving to Melbourne with me and for his continuous support and unconditional love. Dankjewel lieverd.

I gratefully acknowledge the financial support of RMIT's School of Graduate Research and the Australian Department of Defence. I have been lucky to be part of the IDTC program which provided me with the opportunity to undertake course work, attend conferences and a supportive nationwide cohort.

Summary

There is a large variety of real-world phenomena that can be modelled and analysed as networks. Part of this variety is reflected in the diversity of *network classes* that are used to model these phenomena. However, the differences between network classes are not always taken into account in their analysis. This thesis carefully addresses how to deal with distinct classes of networks in two different contexts.

First, the switching model is a well-known model that has been used to randomise different classes of networks, and is typically referred to as *the* switching model. We argue that really we should be talking about a *family* of switching models. We show that it is important to distinguish between the switching model with respect to different classes of networks, because ignoring this distinction has lead to biased sampling. Given that the most common use of the switching model is as a null-model, it is critical that it samples without bias. We provide a comprehensive analysis of the switching model with respect to nine classes of networks and prove under which conditions sampling is unbiased for each class.

Recently the Curveball algorithm was introduced as a faster approach to network randomisation. We prove that the Curveball algorithm samples without bias; a position that was previously implied, but unproven. Furthermore, we show that the Curveball algorithm provides a flexible framework for network randomisation by introducing five variations with respect to different network classes.

We then compare the switching models and Curveball algorithms to several other random network models. As a result of our findings, we recommend using the configuration model for multi-graphs with self-loops, the Curveball algorithm for networks without multiple edges or without self-loops and the ordered switching model for directed acyclic networks.

Second, we extend the theory of the popular technique of motif analysis to directed acyclic networks. Directed acyclic networks are an important class of networks. We establish experimentally that there is no difference in the motifs detected by existing motif analysis methods and our customised method. However, we show that there *are* differences in the detected anti-motifs. Hence, we recommend taking into account the acyclic nature of directed acyclic networks when running motif finding experiments.

Network science is a young and active field of research. Most existing network measures originate in statistical mechanics and focus on statistics of local network properties. Such statistics have proven very useful. However, they do not capture the complete structure of a network. In this thesis we present experimental results on two novel network analysis techniques.

First, at the local level, we show that the neighbourhood of a node is highly distinctive and has the potential to match unidentified entities across networks. Our motivation is the identification of individuals across dark social networks hidden in recorded networks.

Second, we present results of one of the first studies of the application of persistent homology to network analysis. This recently introduced technique from topological data analysis offers a new perspective on networks: it describes the mesoscopic structure of a network.

Finally, we used persistent homology for a classification problem in pharmaceutical science. This is a novel application of persistent homology. Our analysis shows that persistent homology is a promising approach for the classification of the phase behaviour of lipid formulations.

Contents

List of Publications Arising

1. C. J. Carstens. Proof of uniform sampling of binary matrices with fixed row sums and column sums for the fast curveball algorithm. *Physical Review E*, 91:042812, 2015

2. A. Hecker, C. J. Carstens, and K. J. Horadam. Neighbourhood distinctiveness: An initial study. In *Complex Networks VI*, volume 597 of *Studies in Computational Intelligence*, pages 99–110. Springer International Publishing, 2015

3. C. J. Carstens. A uniform random graph model for directed acyclic networks and its effect on motif-finding. *Journal of Complex Networks*, 2:419–430, 2014

4. C. J. Carstens. Motifs in directed acyclic networks. In *Signal-Image Technology Internet-Based Systems (SITIS), 2013 International Conference on*, pages 605–611, 2013

5. J. Jeffers, K. J. Horadam, C. J. Carstens, A. Rao, and S. Boztas. Influence neighbourhoods in CiteSeer: A case study. In *Signal-Image Technology Internet-Based Systems (SITIS), 2013 International Conference on*, pages 612–618, 2013

6. C. J. Carstens and K. J. Horadam. Persistent homology of collaboration networks. *Mathematical problems in engineering*, 2013. Article ID 815035, 7 pages, 2013

List of Figures

List of Tables

1. Introduction

Networks have become an increasingly popular way to model a wide range of real world phenomena. For instance, the structure of our brains, the spread of a computer virus and the effect of a species' extinction can all be described using the language of networks. Our brain is made out of interconnected neurons, a computer virus moves on a network of email addresses and the predator-prey relationships between species reveal a network of dependencies. There are countless other examples of the relevance of networks to daily life.

This thesis reviews existing tools and develops new ways to derive meaning from networks. In particular, it presents research on three topics related to network science: *random network models*, the analysis of *local network properties* and the application of tools from *topological data analysis* in network science.

The interdisciplinary nature of network science results in a large variety of networks. Some networks have relations that are symmetric, such as those in a contact network. Others are directed, like predator-prey relations. Some systems are modelled most naturally as weighted networks, for instance the network of air-travel or trade networks. Dealing with different classes of networks is a central theme throughout this thesis.

Random network models

Random network models have played a central role in the development of network science, and they are important in two key respects. On the one hand, they have helped understand and predict typical network phenomena, as observed in both random networks and real networks. On the other hand, they have helped identify and highlight structural features of real networks that do not occur in random networks and can not be explained by randomness. These models have evolved with the field of network science, by incrementally incorporating structural properties observed in real networks.

A substantial part of this thesis is dedicated to random network models that *fix* the degree sequence of a network and, in particular to their use as null-models. It is very important for a null-model to sample without bias. However, it has proven difficult to develop truly unbiased random network models that produce *simple* networks with fixed degree sequence.

The switching model is a random network model that has mainly been used in the motif finding literature and in ecological network studies. However, the definition of the switching model in the literature is imprecise in two respects: the implementation of the model remains ambiguous and no distinction is made between the treatment of different network classes. In several cases, this imprecision has lead to biased sampling.

This thesis addresses both issues by introducing precise definitions for a *family* of switching models. We refer to *the switching model with respect to $\mathcal{G}$* when it is used to randomize any *specific* class of networks $\mathcal{G}$. We give a detailed discussion of the properties of the switching model with respect to nine different network classes, and prove that for some of these classes small adjustments are necessary to ensure unbiased sampling.

The switching model is a Markov chain based approach to network randomisation. A drawback of this type of model is that it is generally unknown how many steps are needed to reach the stationary distribution. We analyse the run-time of the switching model and compare it to the run-time of a range of alternative random network models. Of particular interest is the comparison with the configuration model, a conceptually different method that is much faster since it *constructs* a random network from scratch. We point out, however, that this method introduces features that can be undesirable.

Recently, the Curveball algorithm has been introduced as a fast alternative to the switching model with respect to directed networks. The Curveball algorithm was shown to sample without bias for specific examples, but its general behaviour has not been analysed. Similarly to the switching model, it is important for the Curveball algorithm to sample without bias, since it too is used as a null-model. We reformulate the Curveball algorithm to reveal its underlying Markov chain, and prove that it indeed samples without bias. Furthermore we introduce variations on the Curveball algorithm, providing a more general framework, for the randomisation of additional classes of networks.

Local network properties

A common approach to derive meaning from a network is to analyse and summarize its local properties. For instance, two examples of such local statistics and their global summary are the node degree and degree sequence, and the local and global clustering coefficient. Techniques based on local network properties have proven very useful for the analysis of networks. Network motifs are another example of a popular method for the analysis of networks based on local properties.

Directed acyclic networks form an important class of networks that appear in many applications. The best known example of directed acyclic networks is that of citation networks. Other examples include patent networks, networks of dependencies in software and of lemmas, axioms and theorems in pure mathematics, as well as biological networks such as predator-prey networks. The existing definition of a motif does not take into account the acyclic nature of this class of networks. We extend the theory of motif finding to apply to directed acyclic networks.

Intuitively it is clear that we can learn a lot about a node by studying all the nodes that it is related to. Neighbourhood analysis is a natural approach to the study of local network properties. We are interested in using the neighbourhood of a node to identify the node itself. Our interest is motivated by identifying persons who wish to remain unidentified by analysing their relations. For this purpose, we present experimental results on the distinctiveness of the neighbourhood of a node in different information networks.

Topological data analysis

The analysis of networks based on local properties is very useful. However, local properties do not capture the complete structure of a network. We show that persistent homology, a technique from the field of topological data analysis, is very suited to the analysis of networks and offers an additional measure of the mesoscopic structure of a network. We present experimental results on the use of persistent homology as a measure for weighted networks, in particular we analyse a collection of collaboration networks. We show that persistent homology brings a novel approach to the analysis of weighted networks. Furthermore we show that it has the potential to do the same for bipartite networks and temporal networks.

Finally we discuss a novel application of topological data analysis. We show that persistent homology is a promising approach to the analysis and classification of lipid formulations. Lipid formulations have applications in pharmaceutical science, where they are used to improve the rate of absorbency of certain drugs. They are currently analysed and classified manually: a time consuming and error-prone approach that would benefit from a more automated process of analysis

This thesis is organised as follows. The current chapter introduces terminology and known results needed throughout the rest of the thesis. It discusses topics in network science, some theoretical properties of Markov chains and the field of topological data analysis. Section 1.2 contains original proofs about 'Eulerian' properties of the edge set difference of certain directed graphs.

Chapters 2, 3, and 4 of this thesis are dedicated to random network models. These chapters mostly contain theoretical results.

Chapter 2 presents our definitions and analysis of the switching model with respect to nine classes of networks. Furthermore it presents our proofs of unbiased sampling for the nine (adjusted) versions of the switching model.

Chapter 3 discusses the run-time of the switching model with respect to directed networks. We compare the run-time of the switching model to that of several alternative

Markov chain models, the configuration model and the recently introduced Expand and Contract method.

Chapter 4 proves that the original Curveball algorithm (with respect to directed networks) samples without bias. In addition, we introduce new variations of the Curveball algorithm for the randomisation of five additional network classes. For the most important variations of the Curveball algorithm, we show that sampling is fast and prove that sampling is unbiased.

In the remaining part of the thesis, Chapters 5, 6 and 7, we present mostly experimental work.

In Chapter 5 we analyse how the well-known technique of motif detection can be applied to the class of directed acyclic networks. Furthermore we investigate the distinctiveness of neighbourhoods within communication and information networks.

In Chapter 6 we present our findings on the application of persistent homology to the analysis of weighted social networks. We also present a more general survey of the utility of persistent homology to the analysis of networks.

Chapter 7 presents our work on the analysis and classification of lipid formulations.

Finally, Chapter 8 summarizes the thesis's contributions to the literature and discusses interesting directions for further research.

1.1. Network science definitions

Network science borrows a lot of terminology and notation from graph theory. The terms network and graph are often used interchangeably and describe the same mathematical object. However, the term network is more often used in applied settings, when describing a real-world phenomena, whereas the term graph is more often used in theoretical context. We adhere to this convention within this thesis.

A network is a collection of nodes and links. Nodes correspond to the *parts* of the system under study, whereas *links* correspond to the connections or relations between the parts. When working with networked data, it becomes apparent that there are some subtle and some obvious structural differences between them. For instance, the relationships in a network may have a direction associated with them or be mutual. There may be weights associated to the relationship or there may be multiple connections between nodes. We now introduce some of the main classes of networks that we work with in this thesis.

In graph theory, a network is known as a graph, its nodes as vertices and its links as edges. In the remainder of this section we will use this more mathematical terminology.

Definition 1.1.1. A **graph** is a pair $G = (V, E)$, where V is a set of vertices and E is a set of unordered pairs of vertices $\{u, v\}$, representing the undirected edges of the graph. Similarly, a **directed graph** is a pair $G = (V, E)$, where E is a set of ordered pairs of vertices (u, v). We say that (u, v) is an edge from u to v and call u the **source** or **tail** of the edge and v the **target** or **head**.

All graphs in this thesis have a finite number of vertices, n, and a finite number of edges m. We will write $V = \{v_1, \ldots, v_n\}$ for the set of vertices of a graph and $E = \{e_1, \ldots e_m\}$ for the set of edges of a graph.

Definition 1.1.2. A **self-loop** is an edge from a vertex to itself: $\{v, v\}$ or (v, v). A **simple (directed) graph** is a (directed) graph that does not contain any self-loops.

Informally, a multigraph is a graph where each pair of vertices may be connected by more than one edge.

Definition 1.1.3. A **multigraph** is a triple $G = (V, E, J)$ where V is a set of vertices, E is a set of edges and J is the incidence map. This map sends each edge $e \in E$ to a two-element subset of V corresponding to the endpoints of e. Two edges e and e' are called **parallel** if $J(e) = J(e')$. **Directed multigraphs** are similarly defined.

To summarize, the table below establishes eight network classes that we will be working with throughout the thesis. We follow the conventions used in network science. However, in graph theory, these classes may be known by different terminology as indicated.

Class	Network Science	Graph Theory	Multiple	Directed	Self loops
$\mathcal{G}_1$	Simple graph	Graph	No	No	No
$\mathcal{G}_2$	Graph	Graph/Pseudograph	No	No	Yes
$\mathcal{G}_3$	Simple directed graph	Digraph	No	Yes	No
$\mathcal{G}_4$	Directed graph	Digraph/Pseudodigraph	No	Yes	Yes
$\mathcal{G}_5$	Multigraph, no self-loops	Multigraph	Yes	No	No
$\mathcal{G}_6$	Multigraph	Pseudograph	Yes	No	Yes
$\mathcal{G}_7$	Directed multigraph, no self-loops	Multidigraph	Yes	Yes	No
$\mathcal{G}_8$	Directed multigraph	Pseudodigraph	Yes	Yes	Yes

Table 1.1.: Eight network classes that we will be working with throughout this thesis.

Definition 1.1.4. A **walk** in a graph $G = (V, E)$ is a sequence of edges $e_1, \ldots, e_l$ such that there are vertices $v_0, \ldots, v_l \in V$ with $e_i = \{v_{i-1}, v_i\}$ for each i. A **closed walk** is a walk with $v_0 = v_l$. A **trail** is a walk with no repeated edges. A trail that starts and ends at the same vertex is a **closed trail**. A **path** is a trail where all vertices v_i are distinct. A closed trail with $l \geq 3$ and all vertices distinct (except for $v_0 = v_l$) is called a **cycle**.

For a directed graph, the definitions of a walk, trail, path, and cycle are the same as above, ignoring the directions on the edges. The following definitions take edge direction into account.

Definition 1.1.5. A **directed walk** in a directed graph $G = (V, E)$ is a sequence of edges $e_1, \ldots, e_l$ such that there are vertices $v_0, \ldots, v_l \in V$ with $e_i = (v_{i-1}, v_i)$ for each $i \in \{1, \ldots, l-1\}$. A **directed trail** is a directed walk with no repeated edges. A **directed path** is a directed walk with all vertices distinct and a **directed cycle** is a directed trail with $v_0 = v_l$ and all other vertices distinct.

Definition 1.1.6. A **directed acyclic graph** is a simple directed graph that does not contain any directed cycles.

Definition 1.1.7. A **topological ordering** of a directed graph $G = (V, E)$ is an ordering of its vertices $V = (v_1, \ldots, v_n)$ such that for each edge $(v_i, v_j) \in E$ the indices satisfy $i > j$.

A directed graph allows a topological ordering if it is directed acyclic [116]. This ordering is normally not unique. The converse holds as well: a directed graph that allows a topological ordering is acyclic. Clearly, such a graph can not contain a directed cycle, since it is impossible for all edges (v_i, v_j) in a directed cycle to have the property $i > j$.

Proposition 1.1.8. A directed graph allows a topological ordering if and only if it is directed acyclic. □

We now introduce some of the terminology that is commonly used to describe graphs and that will be used throughout this thesis.

Definition 1.1.9. A graph G is **connected** if for any pair of vertices u, v there is a walk from u to v. A directed graph G is **connected** if the underlying graph is, and is **strongly connected** if for any pair of vertices there is a directed walk from u to v.

Definition 1.1.10. Let $G = (V, E)$ be a graph, $G' = (V', E')$ is a **subgraph** of G if G' is a graph and $V' \subseteq V$ and $E' \subseteq E$.

Definition 1.1.11. Let $G = (V, E)$ be a graph and let $V' \subseteq V$ be a set of vertices of G. The **subgraph induced** by V' or the **induced subgraph** on V' is the subgraph with vertices V' and edges all edges of G between vertices in V'.

Instead of representing a graph as a pair of sets, it can also be represented as a matrix.

Definition 1.1.12. Let $G = (V, E)$ be a graph, an edge is **adjacent** to a vertex $v \in V$ if it is of the form $\{v, x\}$. The **adjacency matrix** A of a graph G is a binary $n \times n$ matrix. For directed graphs A_{ij} equals one if there is an edge from v_i to v_j and equals zero otherwise. For undirected graphs $A_{ij} = A_{ji}$ equals one if there is an edge between i and j and zero otherwise.

Definition 1.1.13. Let $G = (V, E)$ be a graph. Vertices $u, v \in V$ are **neighbours** if there is an edge connecting them. In directed networks a vertex has both **in-neighbours**, vertices connected by incoming edges, and **out-neighbours**, vertices connected by outgoing edges.

Definition 1.1.14. Let $G = (V, E)$ be a graph and let $v \in V$. The **neighbourhood**

of v is the graph induced by the neighbours of v, but excluding v itself, and is denoted by $N(v)$. This subgraph is sometimes referred to as the open neighbourhood of v. The closed neighbourhood of v is the graph induced by v and its neighbours. This graph is sometimes referred to as the **ego-network** of v. There are analogous definitions for the **in-neighbourhood** and **out-neighbourhood** in the directed case.

An important characteristic of a vertex is the number of edges that are adjacent to it.

Definition 1.1.15. Let $G = (V, E)$ be a graph and let $v \in V$. The **degree** of v is the number of edges adjacent to v. The degree of v is denoted by $k(v)$. The sequence $k(v_1), \ldots, k(v_n)$ is called the **degree sequence** of G.

Definition 1.1.16. Let $G = (V, E)$ be a directed graph. The **in-degree**, $k^{in}(v)$ of a vertex $v \in V$ equals the number of edges of which it is the head, that is the number of edges of the form (x, v). Similarly, the **out-degree**, $k^{out}(v)$ of v is the number of edges (v, x) of which it is the tail. Finally the **total degree**, $k(v)$, of the v is the sum of its in-degree and out-degree. The sequences $k^{in}(v_1), \ldots, k^{in}(v_n)$; $k^{out}(v_1), \ldots, k^{out}(v_n)$ and $k(v_1) \ldots k(v_n)$ are called the **in-degree sequence**, **out-degree sequence** and **degree sequence** of G respectively.

The degree sequence of a graph is a sequence of integers. Not all sequences of integers correspond to the degree sequence of a graph.

Definition 1.1.17. A sequence of integers k is called a **graphical degree sequence** if there exists a graph G with degree sequence equal to k. The graph G is called a **realization** of the degree sequence. Similarly a pair of sequences of integers is called graphical if there exists a directed graph with those sequences as its in-degree and out-degree sequence.

We now define a few special classes of graphs.

Definition 1.1.18. The n**-cycle**, denoted by C_n, is the undirected graph with nodes $v_1, \ldots, v_n$ and edges $\{v_i, v_{i+1(\mod n)}\}$ where $i \in \{1 \ldots n\}$.

Definition 1.1.19. The **complete graph** on n nodes, K_n, is the undirected graph where all distinct vertices are adjacent. This graph is also referred to as the n**-clique**.

It contains ${}^{n(n-1)}/_{2}$ edges.

The next two classes of networks that we discuss are both random network. Networks from this class are defined as the outcome of a stochastic process.

Definition 1.1.20. There are two types of undirected **Erdős-Rényi random network models** [39, 40]. The first, $G(n, m)$, produces random networks with n vertices and m edges. Edges are chosen uniformly at random from the collection of all ${}^{n(n-1)}/_{2}$ possible edges. In the second model, $G(n, p)$, it is not the *number* of edges m that is fixed, but the *probability* p of an edge being present between vertices. This model again produces random networks with n vertices, but now an edge is placed between each distinct vertex pair with independent probability p. **Directed Erdős-Rényi random network models** are defined analogously, with $n(n-1)$ potential edges.

1.2. Edge set differences and Eulerian paths

We now introduce the symmetric edge set difference of graphs [21]. We prove that the edge set difference of two simple directed graphs with equal degree sequences has some nice 'Eulerian' properties [16]. These properties will be needed in Chapter 2. The proofs of Lemmas 1.2.2, 1.2.11 and 1.2.12 are original work.

Definition 1.2.1. Let $G = (V, E)$ and $G' = (V', E')$ be two graphs. The **symmetric edge set difference** of G and G' is defined as $E\Delta E' := (E\backslash E') \cup (E'\backslash E)$. We write $G\Delta G'$ for the graph with vertices $V \cup V'$ and edge set $E\Delta E'$.

We will be interested in the symmetric edge set of graphs with the same set of vertices. We may think of $G\Delta G'$ as a *2-edge-coloured* graph, i.e. a graph where each edge has one of two colours corresponding to whether the edge is in $E\backslash E'$ or in $E'\backslash E$. We will refer to these colours as **red** and **blue** as is customary [11] (see Figure 1.1). We will denote the number of red and blue edges adjacent to v by $k_{red}(v)$ and $k_{blue}(v)$ respectively. From now on **edge-coloured** refers to 2-edge-coloured.

We will be interested in the graph $G\Delta G'$ of two simple directed networks G and G'

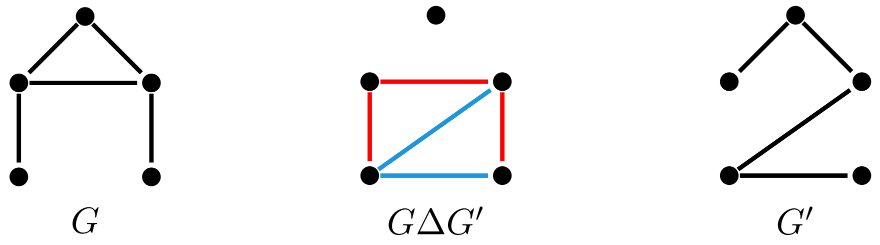


Figure 1.1.: Two graphs G and G' on the same vertex set, and their symmetric edge set difference graph $G\Delta G'$. Edges from G are coloured red and edges from G' blue.

with equal in-degree and out-degree sequence. We first observe the following basic property.

Lemma 1.2.2. Let $G = (V, E)$ and $G' = (V, E')$ be distinct directed networks with equal degree sequences. Let $G\Delta G' = (V, E\Delta E')$. For every vertex $v \in V$ the red and blue in and out degrees are equal in $G\Delta G'$. More precisely $k^{in}_{red}(v) = k^{in}_{blue}(v)$ and $k^{out}_{red}(v) = k^{out}_{blue}(v)$ for all vertices $v \in V$.

Proof. Assume there exists a vertex $v \in V$ with $k^{in}_{red}(v) \neq k^{in}_{blue}(v)$, then the in-degree of v in G is not equal to the in-degree of v in G' which is a contradiction. The case of outgoing edges follows similarly. □

Corollary 1.2.3. Let $G = (V, E)$ and $G' = (V, E')$ be distinct directed networks with equal degree sequences. The symmetric edge set difference $E\Delta E'$ has even cardinality.

Proof.
$$|E\Delta E'| = \sum_i k^{in}(v_i) = \sum_i k^{in}_{red}(v_i) + \sum_i k^{in}_{blue}(v_i) = 2\sum_i k^{in}_{red}(v_i).$$
□

In the remainder of this section we show that any edge-coloured graph with the property, $k^{in}_{red}(v) = k^{in}_{blue}(v)$ and $k^{out}_{red}(v) = k^{out}_{blue}(v)$ for all vertices v allows a type of Eulerian tour.

Definition 1.2.4. An **Eulerian trail** of a graph G is a trail that visits each edge of G (exactly once, of course). An **Eulerian tour** is a closed Eulerian trail. A graph that contains an Eulerian tour is called **Eulerian**. There are analoguous definitions for directed graphs with the additional requirement that the trails are directed trails.

The following theorem about Eulerian graphs is referred to as Euler's theorem. This theorem is a basic result in graph theory and can be found in almost every book on the topic. The theorem as stated below as well as its proof can be found in [62, Theorem 1.3.1].

Theorem 1.2.5. *Let G be a connected multigraph without self-loops. Then the following statements are equivalent: (1) G is Eulerian, (2) each vertex of G has even degree, and (3) the edge set of G can be partitioned into cycles.*

This theorem holds equally well for connected multigraphs since the insertion of self-loops does not change any of the three properties. Inserted self-loops can be added to a Eulerian tour since every vertex has to be visited at least once. A vertex with even degree, still has even degree after adding a self-loop and each self-loop forms a cycle by itself. There is a similar theorem for directed graphs [62, Theorem 1.6.1].

Theorem 1.2.6. *Let G be a connected directed multigraph without self-loops. Then the following statements are equivalent: (1) G is directed Eulerian, (2) each vertex of G has the same in-degree as out-degree, i.e. $k^{in}(v) = k^{out}(v)$ for every $v \in V$, and (3) the edge set of G can be partitioned into directed cycles.*

Again, this theorem holds equally well for directed multigraphs. There is a similar theorem for edge-coloured graphs. To state this theorem, we first need to introduce the concept of an alternating Eulerian trail.

Definition 1.2.7. An **alternating trail** in an edge-coloured graph $G = (V, E)$ is a trail $(e_1, \ldots, e_l)$ where the colour of e_i differs from the colour of e_{i+1} for all $i \in 1, \ldots, l-1$. An **alternating Eulerian trail** of an edge-coloured graph G is an alternating trail that visits every edge exactly once, and an **alternating Eulerian tour** is a closed alternating Eulerian trail where e_1 and e_l have different colours.

A more general version of the following theorem was proved in [13, Theorem 2].

Theorem 1.2.8. *An edge-coloured simple graph G has an alternating Eulerian tour if and only if G is connected and $k_{red}(v) = k_{blue}(v)$ for all vertices v.* □

In the setting that we are interested in, i.e. in that of a simple directed edge-coloured graph with $k^{in}_{red}(v) = k^{in}_{blue}(v)$ and $k^{out}_{red}(v) = k^{out}_{blue}(v)$ for all vertices v, the type of Eulerian path defined next is most natural.

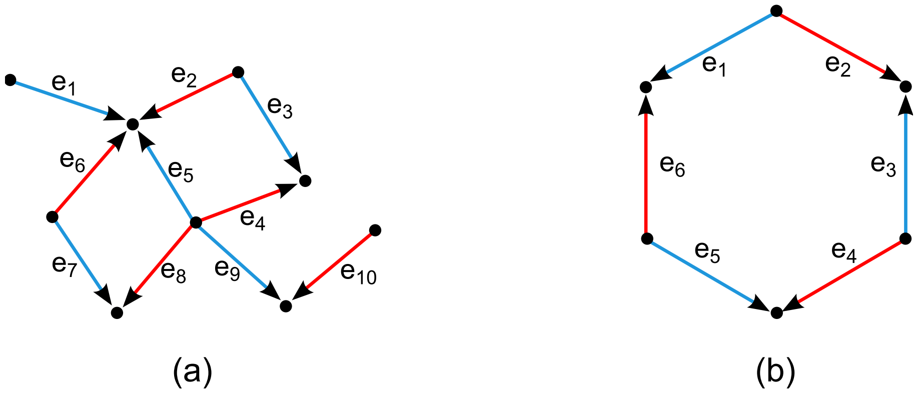


Figure 1.2.: (a) An alternating, direction-alternating trail: $e_1, \ldots, e_{10}$. (b) A closed alternating, direction-alternating trail: $e_1, e_2, e_3, e_4, e_5, e_6$.

Definition 1.2.9. An **alternating direction-alternating trail** in a directed edge-coloured graph $G = (V, E)$ is a trail $e_1, \ldots, e_l$ where the colour and the direction of e_i differs from the colour and the direction of e_{i+1} for all $i \in 1, \ldots, l-1$. An **alternating direction-alternating Eulerian tour** is a closed alternating direction-alternating trail that visits all edges of G and that starts and ends with edges of different colour and direction.

Notice that an alternating direction-alternating trail in an edge-coloured graph traverses all red edges in one direction and all blue edges in the opposite direction (see Figure 1.2).

Definition 1.2.10. A closed alternating direction-alternating trail $e_1, \ldots, e_l$ is **minimal** if its only proper closed alternating direction-alternating sub-trail $e_i, \ldots, e_{i+k}$, is the trail itself.

Minimal closed alternating direction-alternating trails have the following property.

Lemma 1.2.11. A minimal closed alternating direction-alternating trail in a directed edge-coloured graph traverses each vertex at most twice and vertices that are traversed twice are left and re-entered by edges of the same colour (or equally are left and re-entered in opposite directions).

Proof. Let C be a closed alternating trail that traverses a vertex v twice, i.e. $C = e_1$, $\ldots$, e_{i-1}, e_i, $\ldots$, e_{j-1}, e_j, $\ldots$, e_l with e_{i-1}, e_i, e_{j-1} and e_j all adjacent to v. If e_i and e_{j-1} have different colours, then C contains two proper alternating subcycles: $C' = e_1$, $\ldots$, e_{i-1}, e_j, $\ldots$, e_l and $C'' = e_i$, $\ldots$, e_{j-1} and hence C is not minimal. If a vertex v is traversed more than twice, then there always is a sequence of edges in C starting with a red edge from v and ending with a blue edge to v, and hence by the argument above C is not minimal. Thus a minimal closed alternating direction-alternating trail can traverse each vertex at most twice. □

We can now state the following lemma about closed alternating direction-alternating Eulerian trails in edge-coloured graphs, analogous to properties (2) and (3) of Theorem 1.2.5.

Lemma 1.2.12. Let $G = (V, E)$ be a connected edge-coloured simple directed graph. The following statements are equivalent.

1. For each vertex of G both $k^{in}_{red}(v) = k^{in}_{blue}(v)$ and $k^{out}_{red}(v) = k^{out}_{blue}(v)$.

2. The edge set of G can be partitioned into minimal closed alternating direction-alternating trails.

Proof. (1) ⇒ (2): Define a simple edge-coloured undirected graph $\bar{G} = (\bar{V}, \bar{E})$ as follows. Let $V' = \{v \in V | k^{out}(v) > 0\}$ and let $V'' = \{v \in V | k^{in}(v) > 0\}$. Let $\bar{G}$ be the bipartite graph with vertex set the disjoint union of V' and V'' and edges $\{v', v''\}$, $v' \in V'$ and $v'' \in V''$ if $(v', v'') \in E$. Let $p : E \to \bar{E}$ be the map that sends $e = (v_i, v_j)$ to $\bar{e} = \{v'_i, v''_j\}$. By construction, p is a bijection. Let the edges of $\bar{G}$ be coloured using the colours of E and this bijection. Figure 1.3 is an illustration of this construction.

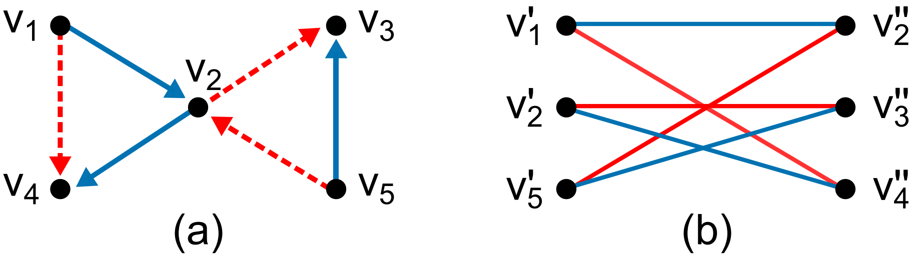


Figure 1.3.: (a) A simple 2-edge coloured graph G with $k^{in}_{red}(v) = k^{in}_{blue}(v)$ and $k^{out}_{red}(v) = k^{out}_{blue}(v)$. (b) The bipartite graph $\bar{G}$ with $k_{red}(v) = k_{blue}(v)$.

For each vertex $v' \in V'$ we find $k_{red}(v') = k^{out}_{red}(v) = k^{out}_{blue}(v) = k_{blue}(v')$. Similarly each vertex $v'' \in V''$ has $k_{red}(v'') = k_{blue}(v'')$. Hence, by Theorem 1.2.8 each connected component of $\bar{G}$ contains a closed alternating Eulerian trail. Let $\bar{C}$ be a closed alternating trail in $\bar{G}$, using the function p^{-1} we obtain a closed alternating trail C in G. The trail $\bar{C}$ alternates between vertices in V' and V'', which implies that C is direction-alternating in G, and thus C is an alternating direction-alternating cycle. This implies that the edge set of G can be partitioned into closed alternating direction-alternating trails. By induction, we may split these trails to obtain a partition of E into minimal closed alternating direction-alternating trails.

(2) $\Rightarrow$ (1): If the edge set of G partitions into closed alternating direction-alternating cycles, then we can find the in-degrees and out-degrees of each vertex by counting the traversals in each of these cycles. For each closed alternating direction-alternating trail we find that a traversal of a vertex v either increases both $k^{in}_{red}(v)$ and k^{in}_{blue} or $k^{out}_{red}(v)$ and k^{out}_{blue} by one. Hence we find that $k^{in}_{red}(v) = k^{in}_{blue}$ and $k^{out}_{red}(v) = k^{out}_{blue}$ for each vertex $v \in V$. □

Combining Lemmas 1.2.2 and 1.2.12 we obtain the following corollary.

Corollary 1.2.13. The symmetric edge set difference of two simple directed graphs with equal in-degree and out-degree sequences can be partitioned into minimal closed alternating direction-alternating trails. □

Notice that such a partitioning is not necessarily unique as shown in Figure 1.4.

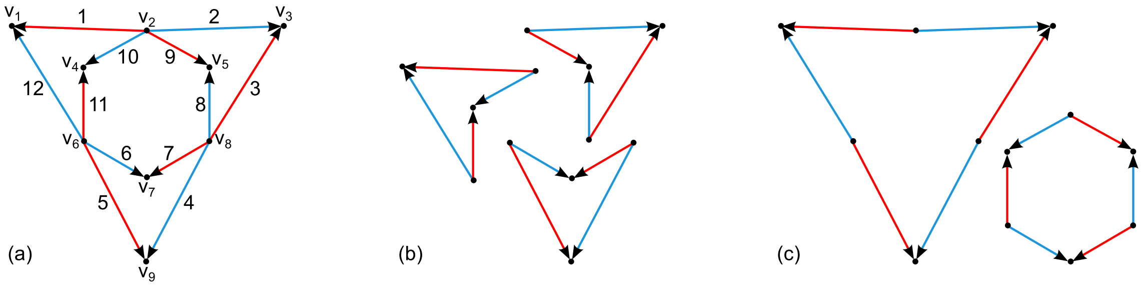


Figure 1.4.: (a) A closed alternating direction-alternating trail $C = (v_1, v_2, v_3, v_8, v_9, v_6, v_7, v_8, v_5, v_2, v_4, v_6, v_1)$. (b)-(c) Two distinct partitions of C into minimal closed alternating direction-alternating trails.

1.3. Markov chains

The random network models to be described in Chapter 2 and Chapter 3 correspond to finite discrete-time Markov chains. In this section we present some well-known basic results about finite discrete-time Markov chains. For a more in depth treatment of the subject we refer the reader to [83].

Definition 1.3.1. A **discrete stochastic process** is a collection of random variables X_t, indexed by a discrete time variable t, that develop over time according to probabilistic rules. A stochastic process has the **Markov property** or **memoryless property** if the probability of each state in the sequence only depends on its immediate predecessor, that is if

$$P(X_{t+1} = x | X_t = x_t, \dots, X_1 = x_1) = P(X_{t+1} = x | X_t = x_t).$$

A **discrete-time Markov chain** is a discrete stochastic process with the Markov property. The random variables X_t can take on a set of possible different values. This set is called the **state space** of the Markov chain. A Markov chain is called **finite** if its state space is.

Definition 1.3.2. Let $\{X_t\}$ be a finite discrete-time Markov chain. For each pair of states x_i and x_j in the state space, the **transition probability** p_{ij} from x_i to x_j is the probability of moving to state x_j given we are in state x_i. That is p_{ij} equals $P(X_{t+1} = x_j | X_t = x_i)$. The Markov property implies that the **transition matrix**, $P = [p_{ij}]$, defines the Markov chain. The transition matrix is a $s \times s$ matrix where s is the size of the state space.

Figure 1.5 shows a simple example of a Markov chain with only four states in its state space. The vertices correspond to the possible states and edge labels indicate the transition probabilities between states. This representation of the Markov chain as a graph is referred to as the **state graph** of a Markov chain.

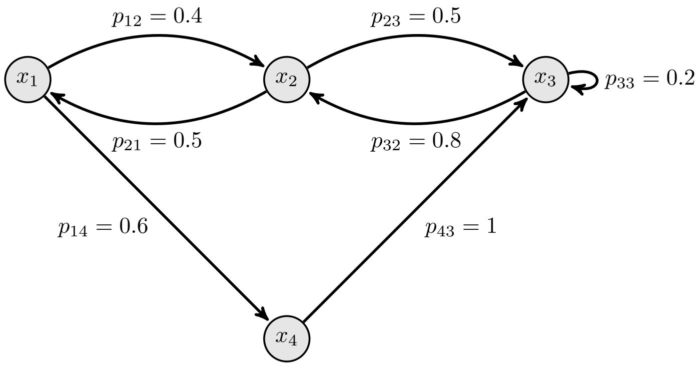


Figure 1.5.: The state graph of a finite Markov chain on four states: x_1, x_2, x_3, x_4.

The transition matrix can be used to find the limiting behaviour of a Markov chain. For a chain starting at state x_i the probability of being at state x_j after k steps is $(e_i P^k)_j$, with $e_i = (0, \dots, 1, \dots, 0)$ the i-th unit vector. If the limit $\lim_{N\to\infty} P^N$ exists then the limiting behaviour is described by $e_i \lim_{N\to\infty} P^N$. The transition matrix of the Markov chain in Figure 1.5 and its limit are

$$P = \begin{pmatrix} 0 & 0.4 & 0 & 0.6 \\ 0.5 & 0 & 0.5 & 0 \\ 0 & 0.8 & 0.2 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \quad \lim_{N\to\infty} P^N = \begin{pmatrix} 5/28 & 5/14 & 5/14 & 3/28 \\ 5/28 & 5/14 & 5/14 & 3/28 \\ 5/28 & 5/14 & 5/14 & 3/28 \\ 5/28 & 5/14 & 5/14 & 3/28 \end{pmatrix}.$$

This Markov chain thus converges to the distribution $(5/28, 5/14, 5/14, 3/28)$; it is most likely to end up in state x_2 or x_3, no matter what the initial state.

Most of the Markov chains that we will discuss, have a state space that is too large to compute the corresponding transition matrix. Fortunately we can make use of a well-known theoretical result to derive the stationary distributions of these Markov chains, without explicitly finding the transition matrix. We now introduce some definitions needed to state this result, i.e. to state Theorem 1.3.6.

Definition 1.3.3. A Markov chain is **irreducible** if its state graph is strongly connected.

Definition 1.3.4. A state x in a Markov chain is **k-periodic** if the greatest common divisor of the length of walks starting and ending at x equals k. A Markov chain is **k-periodic** if every state is.

Definition 1.3.5. A Markov chain is called **aperiodic** iff all its states are 1-periodic.

Notice that a finite irreducible Markov chain is aperiodic if one of its states is (see Figure 1.6).

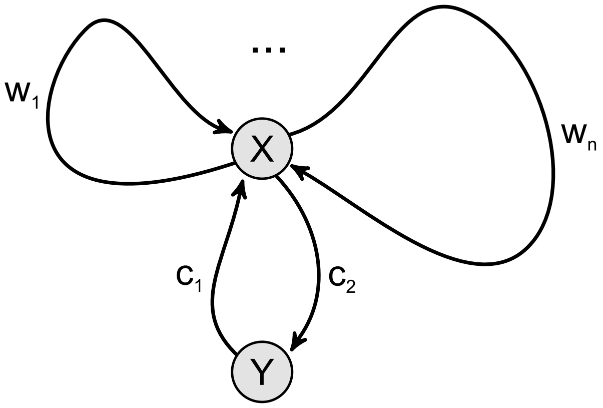


Figure 1.6.: Let X be an aperiodic state in an irreducible state graph. There are closed walks $w_1, \ldots, w_n$ starting and ending at X of length $l_i = |w_i|$ such that $\gcd(l_1, \ldots, l_n) = 1$. For any other state, Y there are walks c_1 from Y to X and c_2 from X to Y. Let $l = |c_1| + |c_2|$ be the sum of their lengths, then $\gcd(l, l + l_1, \ldots, l + l_n) = 1$, and hence Y is also an aperiodic state.

A proof of the following theorem can be found in [83, Theorem 7.10].

Theorem 1.3.6. *A finite irreducible and aperiodic Markov chain converges to a **unique** stationary distribution. If there exists a probability distribution π on its state space such that the **detailed balance equations***

$$\pi_i p_{ij} = \pi_j p_{ji} \text{ for all } i, j, \tag{1.1}$$

are satisfied, then π is this unique stationary distribution.

In Chapters 2 and 4 we will be interested in Markov chains that converge to the uniform distribution. Notice that this is the case under the following conditions.

Corollary 1.3.7. A finite irreducible and aperiodic Markov chain converges to the uniform distribution if the **simplified detailed balance equations** hold

$$p_{ij} = p_{ji} \text{ for all } i, j. \tag{1.2}$$

There are finite irreducible aperiodic Markov chains that converge to the uniform distribution for which the simplified detailed balance equations do not hold. An example is the Markov chain on three states with transition probabilities $p_{11} = p_{12} = p_{22} = p_{23} = p_{31} = p_{33} = {}^1\!/_2$, and $p_{13} = p_{21} = p_{32} = 0$.

The mixing time of a Markov chain quantifies the number of steps needed for the chain to get close to its stationary distribution. We use total variational distance to measure how close a Markov chain is to its stationary distribution.

Definition 1.3.8. The **total variation distance** between two discrete probability distributions p and q on X is given by

$$d_{var}(p,q) = \frac{1}{2}\sum_{x \in X} |p(x) - q(x)|$$

Definition 1.3.9. The **mixing time**, $\tau(\epsilon)$, of a Markov chain with stationary distribution π is defined as follows. For each state x_i in the Markov chain, find the minimum number of steps $\tau_i(\epsilon)$ such that $d_{var}(e_i P^{\tau_i(\epsilon)}, \pi) < \epsilon$. Then $\tau(\epsilon)$ is defined as $\max_i \tau_i(\epsilon)$.

The following informal definition of a rapidly mixing Markov chain can be found in [18]. The idea was first introduced in [2].

Definition 1.3.10. A Markov chain is called **rapidly mixing** if the size of its state space is exponential in some input data size, whereas the mixing time is bounded by a polynomial.

1.4. Topology

Topology and networks are closely related. Both the field of topology and graph theory are sometimes said to have originated from the famous 'bridges of Köningsberg' paper by Euler in 1736 [42]. The problem that Euler solves in this paper is the following: Köningsberg is a city with a central island, which is connected to other parts of the city by bridges (see Figure 1.7[1]). The citizens of Köningsberg want to organise a parade through their city that crosses every bridge exactly once, however they do not seem to be able to find such a route. Euler proves that this is in fact impossible.

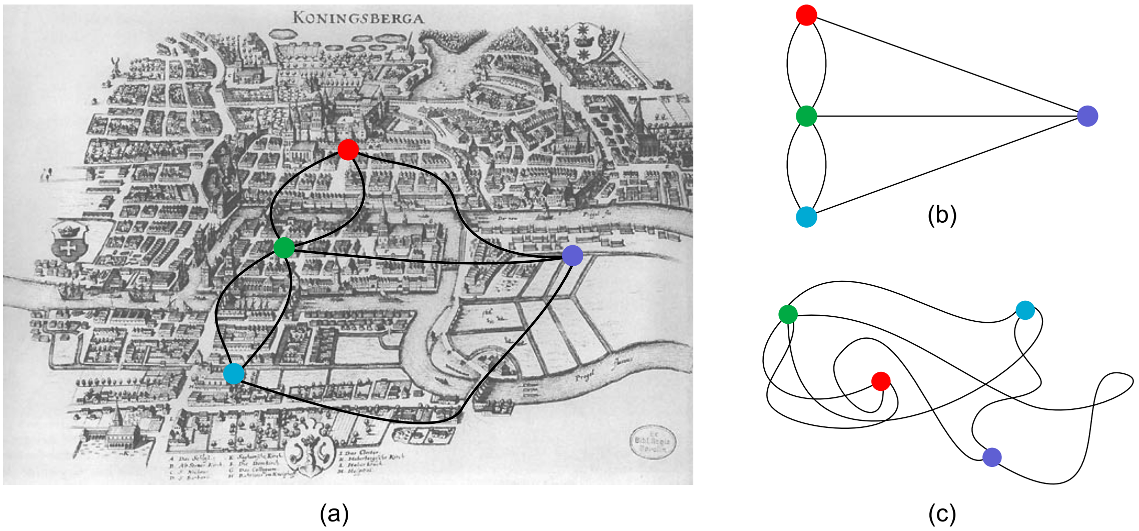


Figure 1.7.: (a) The city of Köningsberg in 1651 and the corresponding network representation. (b)-(c) Two different embeddings of the same network.

Euler realised that in order to solve this problem, it is very useful to represent the problem as what we now know as a multigraph (Definition 1.1.3). He represented land masses as vertices, and bridges as edges between them. He found that the size of the land masses, the length of the bridges and the exact location of the bridges were all irrelevant to the problem. The only thing that *did* matter, was which pairs of land masses where connected and by how many bridges, i.e. *exactly* the information represented in a multigraph.

[1] map by Merian-Erben 1652 http:\\www.preussenchronik.de\bild_jsp\key=bild_kathe2.html

He proved that a 'parade' through the city would only be possible if this multigraph has either no, or exactly two vertices of odd degree. In the latter case the parade has to start and end at the vertices of odd degree. The reason for this is that whenever we visit a landmass during the parade, we have to arrive and leave via different bridges, hence an even number of bridges is 'used' whenever we visit a land mass. These 'routes' that visit every edge in a graph exactly once are now referred to as Eulerian trails, which we have discussed in detail in Section 1.2.

The crucial point here is that the existence or non-existence of a Eulerian trail *only depends* on the connectivity information of the network. That is, it only depends on the way in which vertices are connected by edges, regardless of the *geometry* of the underlying problem. The *topology* of a network refers exactly to the structural properties that are inherent to its combinatorial structure. A network is a purely topological object, since the only information that is stored is this connectivity information.

Notice that for some real-world networks, nodes *do* have a physical location. For instance, the position of neurons in brain networks, and the coordinates of intersections in traffic networks. Networks where each node has coordinates associated with it are referred to as *spatial networks*. Most of the networks analysed in this thesis are not (considered as) spatial networks. However, in Chapter 7, where we construct networks based on coordinates of molecules, the geometry of the data plays an important role.

For non-spatial networks, it can be misleading to plot networks in the plane. When doing so, a layout algorithm decides where to position the nodes of the network. However, a network does not prescribe positions for its nodes, it only contains information about the edges between them. When visualizing a network, technically we are embedding it in the plane $\mathbb{R}^2$ and hence giving it a geometry. That is, we assign a map $e : G \to \mathbb{R}^2$ to the network G. The geometric information, such as the position of the nodes and the distance between the nodes, is not intrinsic to the graph, it depends on the choice of the embedding e.

Topological properties of a network however, are intrinsic. No matter how the network is drawn, its connectivity remains the same. For instance, the topology of the networks in Figure 1.7(b) and (c) is equal whereas their geometry is quite different.

1.5. Topological data analysis

Topological data analysis (TDA) is a relatively new field of research. It builds on the rich theory from algebraic topology to understand and classify data. In this section we discuss one specific tool from TDA, persistent homology, and how it can be used to analyse point clouds. A point cloud is data that is represented as a set of points in $\mathbb{R}^n$.

There are several good reasons to expect that (algebraic) topology may be useful in the analysis of point clouds. Firstly, topology is the branch of mathematics that deals with qualitative geometric information; it describes the connectivity of spaces. In data analysis we are often interested in such qualitative information. For instance, finding the number of clusters in a data set is a classical problem. Secondly, the notion of distance between data points is often constructed in some intuitive rather than strictly theoretical way. It is thus very useful that topological properties are much less sensitive to the choice of a metric than purely geometrical properties. Thirdly, the chosen coordinates for a point cloud are often unnatural. Topology measures properties that do not depend on the choice of coordinates. Finally, summaries are more valuable than individual parameter choices, and the functorality that is central to algebraic topology allows us to map the relation between geometric objects to a relation between the objects we use to describe them [22].

The remainder of this section is organised as follows. We first give the definition of simplicial homology, a homology theory that is combinatorial in nature and hence very suitable for computations. We then give the definition of persistent homology. Finally we discuss how point cloud data can be analysed using persistent homology.

1.5.1. Simplicial homology

One of the main goals of algebraic topology is constructing algebraic invariants of topological spaces. That is, associating algebraic structures, such as groups, rings and modules, to topological spaces in such a way that homeomorphic spaces have the same algebraic structure associated to them. The homology groups of a topological space are such invariants. There are several ways to define the homology groups of a topological space that result in the same groups [50].

In general, it is hard to compute the homology groups of a topological space. In this thesis, we only use simplicial homology, which is a variant of homology that is suitable for computations. Simplicial homology is only defined for topological spaces that have a simplicial structure, that is, it is only defined for simplicial complexes.

Simplicial homology assigns a group $H_i(X)$ to a simplicial complex X for $i \in \mathbb{N} \cup \{0\}$. To compute these groups we do not need to know the topology of a simplicial complex, it is sufficient to know its *combinatorial* structure. Therefore, to simplify things, we will only deal with *abstract* simplicial complexes. From now on we will use the terms simplicial complex and abstract simplicial complex interchangeably.

Definition 1.5.1. An **abstract simplicial complex** is a collection Δ of finite sets such that for each $\sigma \in \Delta$ and $\tau \subset \sigma$, the set τ is also in Δ. An element $\sigma \in \Delta$ is called a **simplex**. The dimension of a simplex σ is $|\sigma| - 1$. The union of all simplices of Δ is called the vertex set V of Δ. The **dimension** of a complex is the supremum of the dimension of its simplices.

A 0-simplex, short for 0-dimensional simplex, is a set of size one. We think of a 0-simplex as a vertex. A 1-simplex is a set of two vertices, and we think of it as an edge. Similarly a 2-simplex is a set of three vertices and can be thought of as a triangular surface and a 3-simplex can be thought of as a solid tetrahedron. In this thesis we will only deal with finite abstract simplicial complexes.

Definition 1.5.2. Let Δ be a simplicial complex. Its k**-skeleton**, Δ_k, is the simplicial complex that contains all simplices $\sigma \in \Delta$ of dimension smaller than or equal to k.

Example 1.5.3. A simple undirected graph $G = (V, E)$ is equivalent to a 1-dimensional simplicial complex $\Delta = V \cup E$.

Homology groups $H_i(X)$ are defined with integer coefficients. However, it is possible to define homology groups $H_i(X; G)$ with coefficients in any group G. In this thesis we always use $G = \mathbb{Z}_2$ and write $H_i(X)$ instead of $H_i(X; \mathbb{Z}_2)$.

Definition 1.5.4. Let X be an abstract simplicial complex. We define $C_p(X)$ as the vector space over $\mathbb{Z}_2$ with basis the p-simplices of X: $\{\sigma_i\}_{i \in I}$. Elements in $C_p(X)$ are called p**-chains** and are formal linear sums $\sum_{i \in I} n_i \sigma_i$ with $n_i \in \mathbb{Z}_2$. They can simply be written as $\sum_{j \in J} \sigma_j$ with $J = \{i \in I | n_i = 1\}$. We define a **boundary map** $\partial_p : C_p(X) \to C_{p-1}(X)$ on the basis elements of C_p

$$\partial_p(v_0, \ldots, v_p) \mapsto \sum_{i=0}^{p} (v_0, \ldots, \hat{v}_i, \ldots, v_p)$$

where the hat indicates removing vertex v_i to obtain a face of the simplex. A key property of the boundary map is that $\partial \circ \partial = 0$ as derived below.

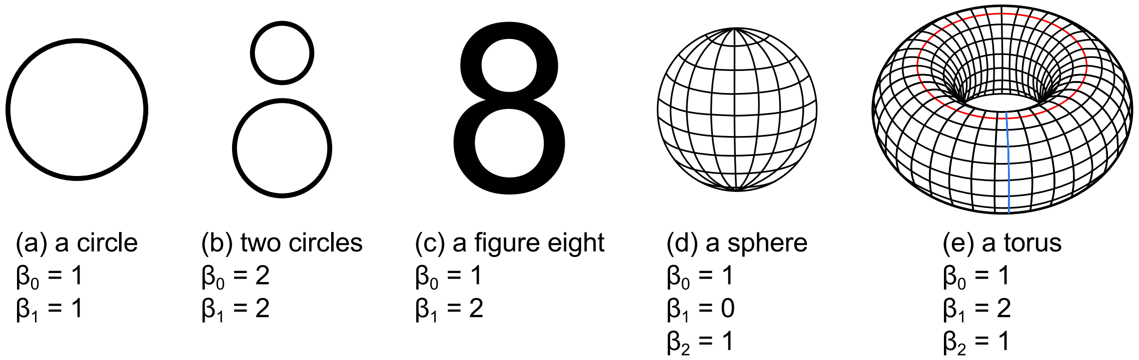
$$\begin{aligned}
\partial_{p-1}\partial_p(v_0, \ldots, v_p) &= \sum_{j<i} \sum_{i} (v_0, \ldots, \hat{v}_j, \ldots, \hat{v}_i, \ldots, v_p) + \sum_{j>i} \sum_{i} (v_0, \ldots, \hat{v}_i, \ldots, \hat{v}_j, \ldots, v_p) \\
&= 2 \sum_{j<i} \sum_{i} (v_0, \ldots, \hat{v}_j, \ldots, \hat{v}_i, \ldots, v_p) \\
&= 0.
\end{aligned}$$

In particular, $\partial \circ \partial = 0$ implies that the image of ∂_{p+1} is a subset of the kernel of ∂_p. These subsets play such an important role in homology that they are given names; the cycles $Z_p(X) = Ker\, \partial_p$ and the boundaries $B_p(X) = Im\, \partial_{p+1}$. We just showed that $B_p(X) \subseteq Z_p(X) \subseteq C_p(X)$. The p-th homology group of X is then defined by

$$H_p(X) = Z_p(X)/B_p(X).$$

By definition ∂ is a linear map, and hence both $B_p(X)$ and $Z_p(X)$ are subspaces of $C_p(X)$. This implies that $H_p(X) = \mathbb{Z}_2^{\beta_p}$, the integer β_p is called the **p-th Betti number** of X.

The homology groups $H_i(X)$ of a d-dimensional simplicial complex are trivial for $i > d$, since $C_i(X) = 0$ for $i > d$. Homology is a functor, this means that a map $f : X \to Y$ induces a homomorphism in homology $f_* : H(X) \to H(Y)$.



(a) a circle
$\beta_0 = 1$
$\beta_1 = 1$

(b) two circles
$\beta_0 = 2$
$\beta_1 = 2$

(c) a figure eight
$\beta_0 = 1$
$\beta_1 = 2$

(d) a sphere
$\beta_0 = 1$
$\beta_1 = 0$
$\beta_2 = 1$

(e) a torus
$\beta_0 = 1$
$\beta_1 = 2$
$\beta_2 = 1$

Figure 1.8.: The zeroth Betti number of a space corresponds to its number of connected components. (a) A circle has one connected component and contains one loop, going around the circle. Hence $\beta_0 = 1$ and $\beta_1 = 1$. (b) For two circles there are two connected components and each circle contains one loop, hence $\beta_0 = 2$ and $\beta_1 = 2$. (c) A figure eight has one connected component and two cycles, thus $\beta_0 = 1$ and $\beta_1 = 2$. (d) A sphere (hollow on the inside), consists of one connected component and encloses a void, namely the inside of the sphere. There are no non-trivial loops, since any loop drawn on the surface can be continuously deformed to a point on the surface of the sphere. Hence $\beta_0 = 1$, $\beta_1 = 0$ and $\beta_2 = 1$. (e) Finally, a torus has one connected component, two non-trivial loops as indicated in red and blue and it encloses a void, hence $\beta_0 = 1, \beta_1 = 2$ and $\beta_2 = 1$.

It is well known that the zeroth Betti number, β_0, corresponds to the number of connected components of a space [50]. Informally, the i-th dimensional Betti number counts the number of i-dimensional holes in a space. We should think of these holes as the space enclosed by a i-dimensional sphere S^i. The 1-dimensional sphere S^1 is a circle, and 1-dimensional homology classes are often discussed in terms of loops in the space. The first Betti number, β_1 roughly counts the number of loops in a space. The 2-dimensional sphere, S^2, is the space that we often just refer to as the sphere, it encloses a void. The second Betti number, β_2 roughly counts the number of voids in a space. There are higher dimensional analogues, but in this thesis we focus on low-dimensional homology. Figure 1.8 gives some examples of spaces and their Betti numbers.

There are algorithms available to compute simplicial homology groups. In [86, Chapter 1.§11] an algorithm for computing simplicial homology with integer coefficients is discussed. Computing simplicial homology with coefficients in a field such as $\mathbb{Z}_2$ is easier. Whether computing simplicial homology with integer coefficients or with field coefficients, in both cases the linear boundary maps are represented as matrices with respect to bases which are the p-simplexes of X. Using a reduction algorithm these matrices are brought into normal form. The homology groups can then be read off these matrices.

When taking homology with coefficients in a field $\mathbb{F}$, the reduction algorithm corresponds to Gaussian elimination to obtain the reduced row echelon form. Now the p-th Betti number, β_p, equals $\dim(Ker\,\partial_p) - \dim(Im\,\partial_{p+1})$. We can find $\dim(Ker\,\partial_p)$ by counting the number of zero-columns in the reduced row echelon form of ∂_p and $\dim(Im\,\partial_{p+1})$ equals the number of non-zero rows of the reduced row echelon form of ∂_{p+1}.

Example 1.5.5. As a simple example we compute the Betti numbers for a triangle (see Figure 1.9) with coefficients in $\mathbb{Z}_2$. The corresponding chain groups and boundary maps are given below.

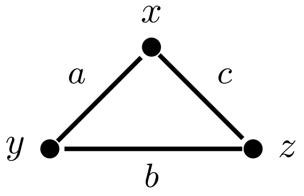


Figure 1.9.: A triangle with labelled simplices.

$$0 \xrightarrow{\partial_2=0} \mathbb{Z}_2\langle a,b,c\rangle \xrightarrow{\partial_1=\left(\begin{smallmatrix}1&1&0\\0&1&1\\1&0&1\end{smallmatrix}\right)} \mathbb{Z}_2\langle x,y,z\rangle \xrightarrow{\partial_0=0} 0$$

We use Gaussian elimination to obtain the reduced row echelon form of ∂_1.

$$\begin{bmatrix}1&1&0\\0&1&1\\1&0&1\end{bmatrix}^{(x,y,z)}_{(a,b,c)} = \begin{bmatrix}1&1&0\\0&1&1\\0&1&1\end{bmatrix}^{(x,y,x+z)}_{(a,b,c)} = \begin{bmatrix}1&0&1\\0&1&1\\0&0&0\end{bmatrix}^{(x+y,y,x+y+z)}_{(a,b,c)} = \begin{bmatrix}1&0&0\\0&1&0\\0&0&0\end{bmatrix}^{(x+y,y,x+y+z)}_{(a+c,b+c,c)}.$$

We thus find

$$\begin{aligned}\beta_0(T) &= \dim(Ker\partial_0) - \dim(Im\partial_1) = 3-2 = 1\\ \beta_1(T) &= \dim(Ker\partial_1) - \dim(Im\partial_2) = 1-0 = 1.\end{aligned}$$

1.5.2. Persistent homology

Instead of computing the homology of a single simplicial complex, persistent homology computes the homology groups for a filtration of a simplicial complex X. The definitions in this and following sections are based on Carlsson's paper [22] and the book by Edelsbrunner and Harer [35].

Definition 1.5.6. A **filtration** $\{X_i\}$ of a space X is a sequence of spaces, $X_0, X_1, \dots X_n$ such that $X_0 = \emptyset$, $X_n = X$ and X_{i-1} is a subspace of X_i for all i. We denote the inclusion maps between these spaces by $\iota_i : X_i \hookrightarrow X_{i+1}$.

In each dimension i we obtain a sequence of vector spaces $H_p(X_i)$ and homomorphisms induced by the inclusion maps

$$0 \xrightarrow{\iota_*} H_p(X_1) \xrightarrow{\iota_*} H_p(X_2) \xrightarrow{\iota_*} \dots \xrightarrow{\iota_*} H_p(X_{n-1}) \xrightarrow{\iota_*} H_p(X_n).$$

Notice that for any pair of indices $i < j$ there is an inclusion map $\iota^{i,j}$ from X_i to X_j.

Definition 1.5.7. Let $\{X_i\}_{i=0}^n$ be a filtration of X. The **p-th persistent homology groups** $H_p^{i,j}(X)$ are the images of the homomorphisms induced by the inclusion map, that is $H_p^{i,j}(X) = Im\, \iota_*^{i,j}$, for $0 \leq i < j \leq n$. The corresponding **p-th persistent Betti numbers** are the ranks of these groups, $\beta_p^{i,j} = \text{rank}\, H_p^{i,j}(X)$.

Notice that $H_p^{i,i}(X) = H_p(X_i)$. The persistent homology groups $H_p^{i,j}(X)$ consist of the classes in $H_p(X_i)$ that are still 'alive' i.e., that are nonzero in X_j. An equivalent definition for the k-th persistent homology groups is $H_p^{i,j} = Z_p(X_i)/(B_p(X_j) \cap Z_p(X_i))$, in other words, the p-dimensional cycles in X_i that have not become a boundary in the filtration from X_i to X_j. This is well-defined since both $B_p(X_j)$ and $Z_p(X_i)$ (by inclusion) are subspaces of $C_p(X_j)$.

For a homology class a in $H_p(X_i)$, we say that it is **born** at i if it is not in $H_p^{i-1,i}(X)$, in other words, if it is not in the image of $H_p(X_{i-1})$. For a class a that is born at i, we say it **dies** at j if it merges with an older class in X_j. That is, if $\iota_*^{i,j-1}(a) \notin H_p^{i-1,j-1}(X)$ but $\iota_*^{i,j}(a) \in H_p^{i-1,j}(X)$, see Figure 1.10. The rule that we say that the younger class dies when two classes merge, is called the **Elder rule** [35].

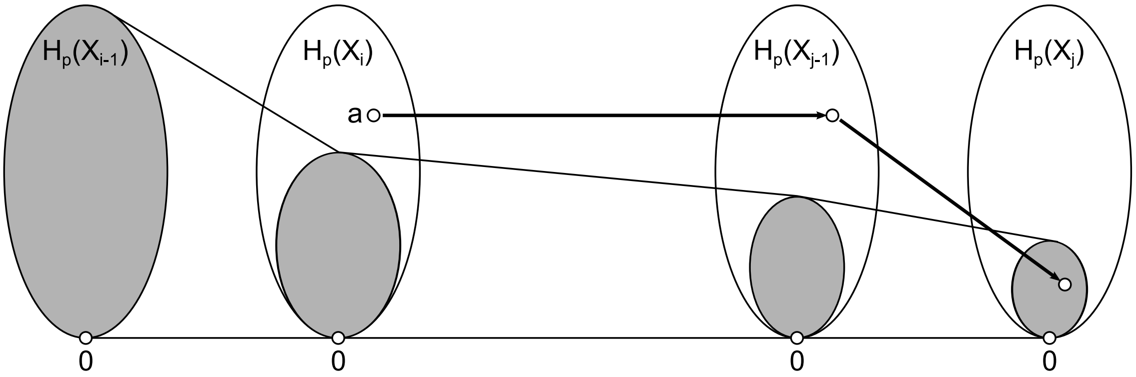
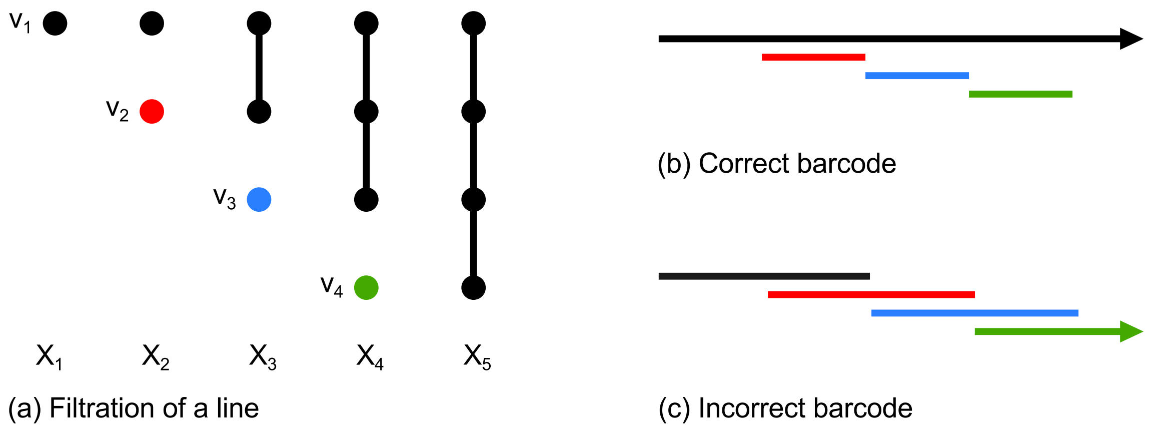


Figure 1.10.: Each ellipse represents a vector space, the shaded area shows the image of $H_p(X_{i-1})$ under the maps $\iota_*^{i,-}$. The class a is born at i, since it does not lie in the image of $H_p(X_{i-1})$. The class a dies at j since this is the first time it does lie in the image of $H_p(X_{i-1})$.

A homology class that is born at i and dies at j has **persistence** $j-i$. If a class never dies it is said to persist, and its persistence is infinity. We may associate a **birth-death pair**, (i, j) or (i, ∞), to each homology class that appears in a filtration. The fundamental lemma of persistent homology tells us that all information about the persistent homology groups in a filtration are given by these birth-death pairs. A major advantage is that these birth-death pairs are much easier to visualize and understand than the collection of persistent homology groups $H_p^{i,j}$. One way of representing the birth-death pairs is the persistence barcode.

Definition 1.5.8. The persistence barcode of a filtration $\{X_i\}$ is a collection of intervals (b_j, d_j) with $b_j \in \mathbb{N}, d_j \in \mathbb{N} \cup \{\infty\}$, corresponding to the births and deaths of homology classes in the persistent homology groups of $\{X_i\}$.

Example 1.5.9. Let $\{X_i\}$ be the filtration illustrated in Figure 1.11(a). This filtration only has homology in dimension zero. The homology class corresponding to the connected component v_1 is born at $i = 1$ and persists throughout the filtration. At $i = 2$ a second homology class corresponding to the connected component of v_2 is born. This homology class merges with the the older class v_1 at $i = 3$. Similarly there is a class born at $i = 3$ and $i = 4$, and these die at $i = 4$ and $i = 5$ respectively. The corresponding barcode is shown in Figure 1.11(b). Without the Elder rule, the barcode associated to this filtration would not be uniquely defined. For instance, the barcode in Figure 1.11(c) would offer an alternative.



(a) Filtration of a line

(b) Correct barcode

(c) Incorrect barcode

Figure 1.11.: (a) A simple filtration of a line. (b) The unique persistence barcode of this filtration. (c) An alternative barcode that does not obey the Elder rule.

An alternative way of representing birth-death pairs is the persistence diagram.

Definition 1.5.10. The persistence diagram of a filtration $\{X_i\}$ is a multiset of points $(b_i, d_i) \in \bar{\mathbb{R}}^2$, where $\bar{\mathbb{R}} = \mathbb{R} \cup \{\pm\infty\}$. These points correspond to the births and deaths of homology classes in the persistent homology groups of $\{X_i\}$. The multiplicity of (b_i, d_i) is given by the number of distinct homology classes that are born at b_i and die at d_i. For technical reasons all points on the diagonal are included with infinite multiplicity. The persistence diagram is visualised as a collection of points in the plane and the diagonal (x, x).

Both the barcode and the persistence diagram illustrate the same information: the birth-death pairs (b_i, d_i) corresponding to the persistent homology groups of a filtration of a space X.

An advantage of the barcode representation is that if there are multiple classes with the same birth and death pair, they are represented as bars of equal length, whereas this results in overlapping points in the persistence diagram. Long bars in the barcode representation correspond to classes that *persist* for a long time, which are the most important classes. In the persistence diagram, important classes are slightly harder to spot, since they correspond to points with a large vertical distance to the line $y = x$.

The barcode representation can become rather cumbersome when there are a lot of

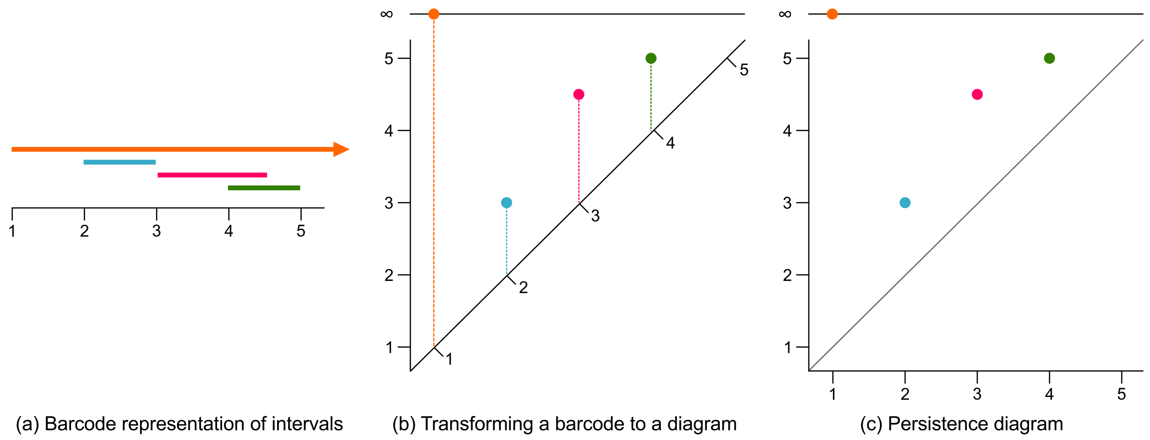


Figure 1.12.: Different visualizations of the collection of intervals $\{(1,\infty),(2,3),(3,4.5),(4,5)\}$. (a) In the barcode representation each interval is represented as a bar with x-coordinates its birth and death. The longer a bar the more important the corresponding topological feature. (b) We obtain the persistence diagram by rotating each bar by 90 degrees around its starting point and shifting it vertically such that its y-coordinate corresponds to the birth, the top of each bar then corresponds to the point (b_i, d_i). (c) The persistence diagram of a collection of intervals displays each interval as the point (b_i, d_i) in $\mathbb{R}^2$. The further the vertical distance of a point from the line $y = x$ the more important the corresponding topological feature.

barcodes, since it expands vertically for each additional homology class. An advantage of the persistence diagram in this case is that it always takes up a fixed amount of space. Furthermore, the persistence diagram always looks exactly the same for a given set of birth- and death-pairs, whereas the barcode looks different depending on the order in which we draw the bars. It is often easier to detect a pattern using the persistence diagram.

We finish this section by discussing a more complicated persistent homology computation.

Example 1.5.11. Let $\{X_i\}$ be the filtration depicted in Figure 1.13. The four vertices a, b, c and d in X_1 correspond to four 0-dimensional homology classes. Only one persists to the end of the filtration, the others die at X_2 where all components are merged into one. No other 0-dimensional classes appear during the filtration, since there is only one connected component in all the following simplicial complexes.

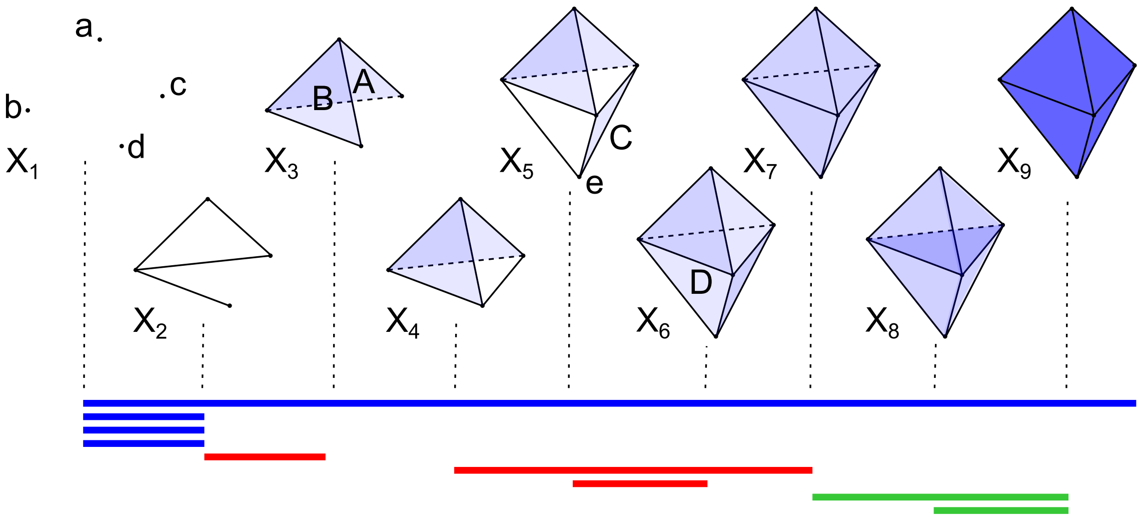


Figure 1.13.: A filtration of a simplicial complex and its persistent homology intervals represented as a barcode. Blue bars correspond to 0-dimensional homology classes, red bars to 1-dimensional homology classes and green bars to 2-dimensional homology classes.

At X_2 a 1-dimensional homology class is born. This class can be represented by the cycle $[a, b] + [a, c] + [b, c]$. This cycle dies at X_3 where it becomes the boundary of face A. At X_4 another cycle is born which can be represented by $[b, c] + [b, d] + [c, d]$. It may look as though there is a second homology class that can be represented by the cycle, $[a, c] + [a, d] + [c, d]$. In fact, these cycles respresent the same homology class, since one can be obtained from the other by adding boundaries of faces. That is, $[b, c]+[b, d]+[c, d] = [a, c] + [a, d] + [c, d] + \partial(A) + \partial(B)$, where $A = [a, b, c]$ and $B = [a, b, d]$.

Another 1-dimensional homology class is born at X_5, it can be represented by $[b, d] + [b, e]+[d, e]$. It appears as though a class represented by $[b, e]+[e, c]+[c, b]$ is also born at X_5. However this class is equivalent to the sum of the two previously mentioned classes. That is, $[b, c] + [b, d] + [c, d] + [b, d] + [b, e] + [d, e] + \partial([c, d, e]) = [b, c] + [c, d] + [b, e] + [d, e] + [c, d] + [d, e] + [e, c] = [b, c] + [b, e] + [e, c]$.

The homology class represented by $[b, d] + [b, e] + [d, e]$ dies at X_6 when it merges with $[b, c] + [b, d] + [c, d]$ since $[b, d] + [b, e] + [d, e] = [b, c] + [b, d] + [c, d] + \partial(C) + \partial(D)$. In X_7 this 1-dimensional homology class dies.

In X_7, a 2-dimensional homology class is born, corresponding to the hole inside. In X_8 another 2-dimensional homology class is born, since the inside of X_8 is now divided into two compartements by the triangle $[b, c, d]$. Both these classes die in X_9 where the tetrahedra $[a, b, c, d]$ and $[b, c, d, e]$ are added to the space.

1.5.3. Common constructions of filtrations

The construction of a filtration depends on the problem at hand. There are two common ways in which filtrations are obtained: one method constructs a filtration of simplicial complexes from a point cloud whereas the other method constructs a filtration from a simplicial complex and a suitable real valued function on the simplicial complex. We now discuss these methods in more detail.

When constructing a filtration of simplicial complexes from a discrete set of points, the idea is to recover some of the underlying structure of the point cloud. For example, we would like to be able to detect that the point cloud in Figure 1.14(a) contains one loop and that the point cloud in Figure 1.14(b) consists of three connected components.

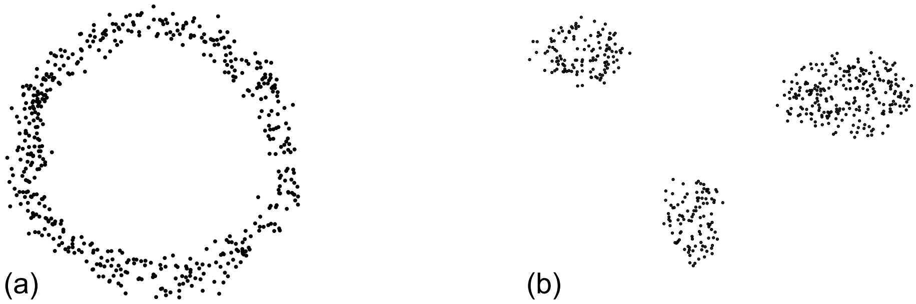


Figure 1.14.: Two point clouds.

The following two constructions are often used to obtain a simplicial complex from a point cloud that reflects the underlying topological structure: the Cech complex and the Vietoris Rips complex.

Definition 1.5.12. The Cech complex $C(X, \epsilon)$ of a metric space (X, d) is the simplicial complex Δ with vertices X and simplices $\{x_k, \ldots, x_l\} \in \Delta$ if $\cap_{i=k}^{l} B(x_i, \epsilon) \neq \emptyset$, where

$B(x, \epsilon)$ is the open ball of radius ϵ centred at x with respect to the metric d.

The following theorem [22, Theorem 2.4] tells us that the Cech complex is a very good candidate to find the topology of the space underlying a point cloud.

Theorem 1.5.13. *Let M be a compact Riemannian manifold. Then there is a positive number e such that $C(M, \epsilon)$ is homotopy equivalent to M whenever $\epsilon \leq e$. Furthermore, for every $\epsilon \leq e$, there is a finite subset $V \subset M$ such that the subcomplex $C(V, \epsilon) \subseteq C(M, \epsilon)$ is also homotopy equivalent to M.*

Unfortunately the Cech complex is computationally expensive. The Vietoris-Rips complex does not approximate the underlying space as well, but is much easier to compute.

Definition 1.5.14. The Vietoris-Rips complex $VR(X, \epsilon)$ of a metric space (X, d) is the simplicial complex with vertices X and simplices $\sigma_i = \{x_{i_1}, \ldots, x_{i_d}\}$ if the pairwise distance $d(x_{i_k}, x_{i_l})$ is less than or equal to ϵ for all x_{i_k} and x_{i_l}.

Notice that the 1-dimensional subcomplex of the Vietoris-Rips complex *defines* the whole complex. We only need to compute the pair-wise distance of each pair of points of the point cloud to be able to construct the complete simplicial complex. Notice that a 1-dimensional subcomplex of a simplicial complex is a network.

Both these constructions associate a simplicial complex to a point cloud and the parameter ϵ. Furthermore, whenever $\epsilon < \epsilon'$ the Cech and Vietoris-Rips complex corresponding to ϵ are subcomplexes of the Cech and Vietoris-Rips complex corresponding to ϵ' respectively. Thus, for an ascending sequence of values $\epsilon_1 < \epsilon_2 <, \ldots, < \epsilon_n$ we obtain filtrations:

$$\emptyset \subset C(X, \epsilon_1) \subseteq C(X, \epsilon_2) \subseteq \cdots \subseteq C(X, \epsilon_n)$$

and

$$\emptyset \subseteq VR(X, \epsilon_1) \subseteq, VR(X, \epsilon_2) \subseteq \cdots \subseteq VR(X, \epsilon_n).$$

Figure 1.15 shows a simple Vietoris-Rips filtration based on this construction.

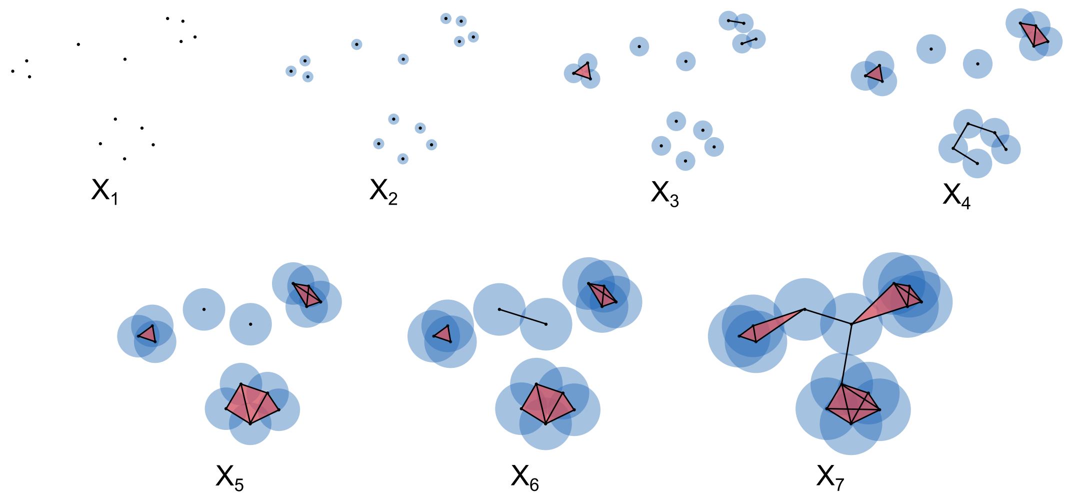


Figure 1.15.: Vietoris-Rips complex filtration of the point cloud X_1, using the Euclidean metric.

1.5.4. Persistence landscapes

Persistent homology gives us qualitative insight into the topology of a space. However in order to compare or classify spaces, we need to develop some statistical tools for persistent homology. There are several distance measures to compare persistence barcodes and diagrams. The two distance measures that are most commonly used are the Bottleneck distance and Wasserstein distance [35].

In this thesis we instead use the landscape distance between persistence landscapes [20, 19]. The reason we use this measure, is that persistence landscapes allow basic operations like subtraction, addition and calculating the mean. Efforts are being made to define similar concepts for the persistence diagram [118]. However, the approach via the persistence landscape appears more straightforward.

Definition 1.5.15. Let $B = \{(b_i, d_i)\}$ be a collection of birth-death pairs. For each pair let $f_{(b_i,d_i)}$ be the piecewise linear function $f_{(b_i,d_i)} : \mathbb{R} \to [0, \infty]$ defined by

$$f_{(b_i,d_i)}(x) = \begin{cases} 0 & \text{if } x \notin (b, d) \\ x - b & \text{if } x \in (b, \frac{b+d}{2}) \\ -x + d & \text{if } x \in (\frac{b+d}{2}, d) \end{cases}$$

These piecewise linear functions are used to define the persistence landscape of a collection of birth-death pairs.

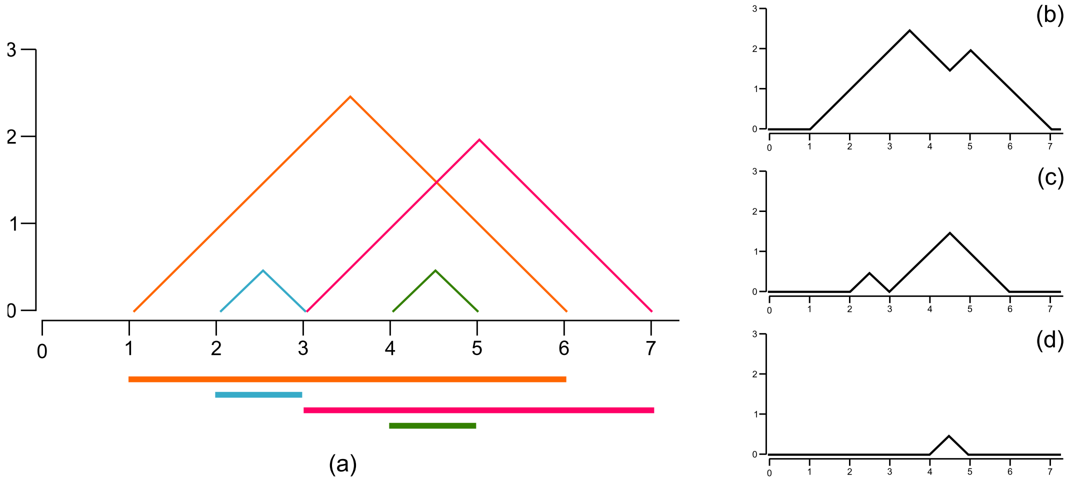


Figure 1.16.: (a) A persistence barcode and the corresponding piecewise linear functions for each birth-death pair. (b)-(d) The corresponding landscape functions λ_1, λ_2 and λ_3.

Definition 1.5.16. The **persistence landscape** of a collection of birth-death pairs $\{[b_i, d_i]\}$ is a set of functions $\lambda_k : \mathbb{R} \to \mathbb{R}$ such that $\lambda_k(x)$ is the k-th largest value of $\{f_{(b_i,d_i)(x)}\}$ and zero if the k-th largest value does not exist.

In Chapters 6 and 7 we make use of **average persistence landscapes** and the L^p **persistence landscape distance**. For their respective definition we refer to [19].

2. Switching models

Random network models have played a central role in the development of network science. The first thorough study of the topic was by Erdős and Rényi in two theoretical papers around 1960 [39, 40]. The class of random networks analysed in these papers is now referred to as Erdős-Rényi random graphs. With great foresight, they already mention that their random network model may be used as a crude model of real-world networks:

> "In fact, the evolution of graphs may be considered as a rather simplified model of the evolution of certain real communication-nets (railway, road or electric network systems, etc.) of a country or some other unit. (Of course, if one aims at describing such a real situation, one should replace the hypothesis of equiprobability of all connections by some more realistic hypothesis.)" [40]

There are several reasons why random network models have been a popular area of research. Firstly, they are an interesting object of study in themselves and provide a rich theory as established by Erdős and Rényi . Secondly, and perhaps more interestingly in the context of network science, random networks can be used to model real networks and provide us with explanations of complex phenomena. Finally, random networks can be used as a null-model: by comparing real networks to random networks important properties of real networks may be revealed.

Before we continue, it is about time to give a definition of a random network model.

Definition 2.0.1. A **random network model** or **random graph** is a graph sampled according to a given probability distribution from a collection of graphs. This collection of graphs is called the **graph ensemble**.

The simple Erdős-Rényi random network model successfully explains certain phenomena observed in real-world networks, however, generally speaking it turns out not to be a very realistic model of real-world networks. Two big discoveries in network science show how real networks differ from Erdős-Rényi random graphs. Firstly, many real networks have the small-world property [120] and secondly their distribution of vertex degrees is scale-free [12]. After these discoveries a lot of research has gone into incorporating these properties into random graph models [95, 84, 93, 6].

Here, we focus on random network models that incorporate the scale-free property, because this property has a big impact on the structure of a network. The models that we discuss in fact *fix the degree sequence* of random networks. Our main motivation to study such random network models, is to use them as a null-model while analysing networked data. For this reason, we are not just interested in a method that can generate networks with a fixed degree sequence, we require a method that does so in an unbiased way.

It is hard to generate truly unbiased samples of networks with fixed degree sequences. Existing random network models that achieve this fall into two categories: the 'fill methods' and methods based on Markov chains. Fill methods *construct* a network, starting with just nodes and adding edges one at a time until reaching the desired in-degree and out-degree distribution [112, 84, 95]. These methods are generally fast. However they either produce a biased sample or only rarely produce a network that is not a multigraph [69, 25].

Methods based on Markov chains are known under many different names, such as the switching model, re-wiring model, swap model and trade model. These methods do not construct a network, instead they *randomize* a given network by repeatedly making small changes. Care must be taken when implementing these models, as it has been shown that certain versions of these models sample with bias [103, 6, 69, 81, 79, 70]. However, when correctly implemented, a method known as *the switching model* has been shown to sample uniformly [6, 79]. The disadvantage of these methods is that they are generally much slower than the 'fill methods'. The speed of the switching algorithm is discussed in more detail in Chapter 3.

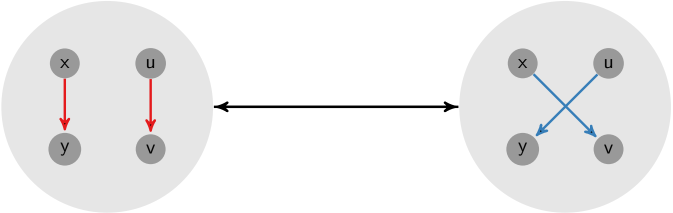


Figure 2.1.: For directed networks, a switch consists of removing two edges (x, y) and (u, v) and inserting two edges (x, v) and (u, y).

The switching model randomizes a network by repeatedly switching edges. For directed networks, a switch consists of the removal of two edges, say (x, y) and (u, v) and the insertion of two new edges (x, v) and (u, y) (see Figure 2.1). This operation changes the topology of a network, while maintaining the in-degree and out-degree of each node. The switching model repeatedly switches edges, until a network is ‘sufficiently randomised’, resulting in a random network with fixed degree sequence. *In the remainder of this chapter, all networks are assumed to contain at least two edges.*

This simple description of the switching model is similar to the description found in the literature [78, 82]. The lack of preciseness in this definition causes inconsistencies. Firstly, ambiguities in this definition lead to different implementations of the model. And secondly, the switching model has been used for different classes of networks, without properly addressing fundamental differences between them.

This chapter discusses several Markov chain based methods for randomizing networks with fixed degree sequence. In Section 2.1 we discuss different interpretations of the switching model with respect to simple directed networks, and give a precise definition of the preferred method. In Section 2.2 we present an overview of the family of switching models for several network classes. In Section 2.3 we discuss a problem in the implementation of switching models for undirected networks. In Section 2.4 we introduce the ordered switching model for the class of directed acyclic networks. We finish the chapter with our conclusion in Section 2.5.

My contributions to the literature in this chapter are the following.

- Introducing a more precise definition for the switching model.

- Compiling an overview of the properties of the switching model with respect to eight different network classes, and the derivation of the stationary distribution for some of these classes.

- Adjusting the switching model with respect to certain classes of networks, to ensure convergence to the uniform distribution.

- Detecting an error in the implementation of the switching model with respect to undirected networks in MFinder (Appendix B.2.5). MFinder is the software package that accompanies the seminal paper on network motifs by Milo et al. [82].

- Introducing the ordered switching model for directed acyclic networks, and proving that it produces unbiased samples. These results are published in [25].

2.1. Switching models for simple directed networks

Throughout this section all networks will be simple directed networks. We discuss the switching model with respect to these networks in Markov chain terminology (see Section 1.3). Theorem 1.3.6 is used to analyse the convergence properties of the switching model. Before starting this analysis, we discuss the ambiguity in the current definition of the switching model, and introduce a more precise definition. We define a directed switch as follows.

Definition 2.1.1. There exists a **directed switch** from network $G_i = (V_i, E_i)$ to $G_j = (V_j, E_j)$ iff $V_j = V_i$, $E_i \neq E_j$ and there are two edges (x, y) and (u, v) in E_i such that $E_j = (E_i \backslash \{(x, y), (u, v)\}) \cup \{(x, v), (u, y)\}$.

The symmetric edge set difference (see Definition 1.2.1) of two networks G_i, G_j that differ by a directed switch is thus $E_i \Delta E_j = \{(x, y), (u, v)\} \cup \{(x, v), (u, y)\}$. Figure 2.2 illustrates the directed switch between two networks, as a diagram of their symmet-

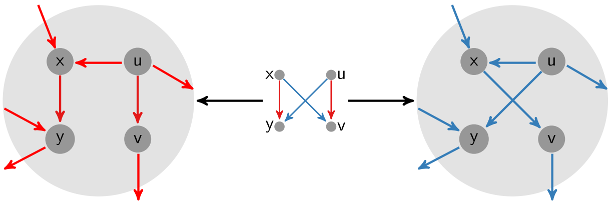


Figure 2.2.: The network on the right is obtained by removing the two edges (x, y) and (u, v) from the network on the left and inserting the two edges (x, v) and (u, y), in other words by applying a directed switch. Similarly the network on the left can be obtained by applying the reversed switch. The diagram in the middle illustrates this switch by indicating which edges have to be removed from a network (in the colour of the network itself) and which edges have to be inserted into the network (in the colour of the other network) to obtain the other network.

ric edge set difference. This type of diagram will be used repeatedly throughout this chapter.

As discussed in the introduction, the ambiguity in the definition of 'the' switching model leads to two common interpretations [6, 79]. These interpretations correspond to two different Markov chains defined on the same state space, but that differ in their transition probabilities. To make this statement concrete, we now analyse in detail a small example.

Example 2.1.2. Let G_1 be a directed network with five vertices $V = \{1, 2, 3, 4, 5\}$ and four edges $E_1 = \{(1, 4), (2, 3), (4, 2), (4, 5)\}$ (Figure 2.3 top-left). It has in-degree sequence $k^{in} = (0, 1, 1, 1, 1)$ and out-degree sequence $k^{out} = (1, 1, 0, 2, 0)$. There are exactly five simple directed networks with these degree sequences. Figure 2.3 shows these networks and the directed switches (if any) between them.

In the first interpretation of the switching model, the Markov chain changes state in every step. If G_i is the current state, the next state is a neighbour of G_i chosen at random. The transition probability p_{ij} from state G_i to G_j is ${}^{1}/_{k(G_i)}$, where $k(G_i)$ is the degree of G_i (Figure 2.4(a)). This Markov chain can be described by the matrix of transition probabilities P below, and the limiting distribution can be found by taking

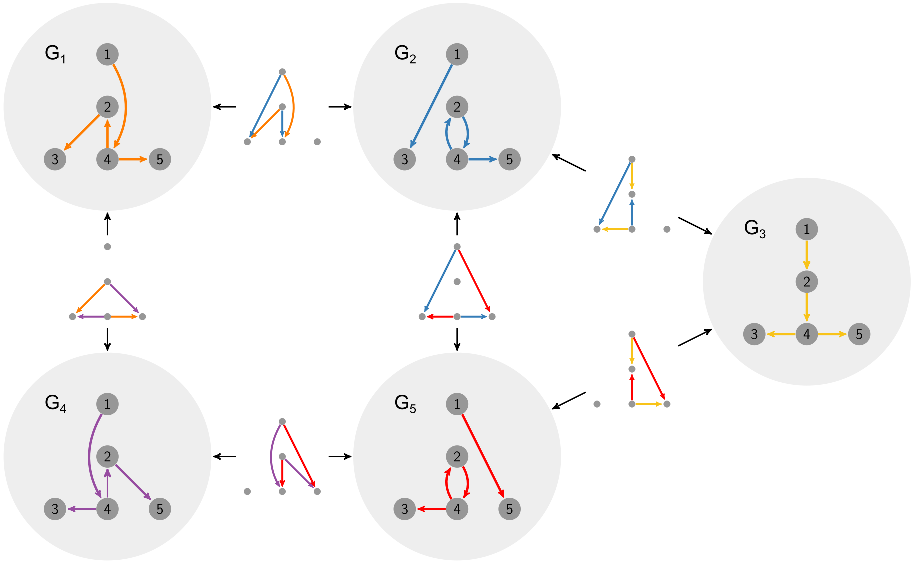


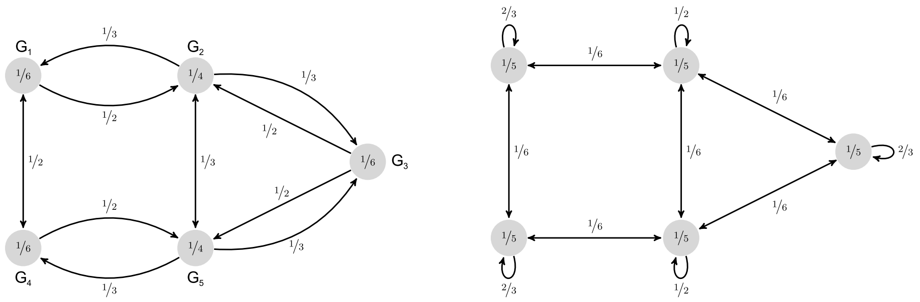
Figure 2.3.: There are exactly five simple networks realizing the degree sequences $k^{in} = (0,1,1,1,1)$ and $k^{out} = (1,1,0,2,0)$. Neighbouring realizations differ by a switch, as indicated by the diagram on the edge.

the limit of P^N.

$$P = \begin{pmatrix} 0 & 1/2 & 0 & 1/2 & 0 \\ 1/3 & 0 & 1/3 & 0 & 1/3 \\ 0 & 1/2 & 0 & 0 & 1/2 \\ 1/2 & 0 & 0 & 0 & 1/2 \\ 0 & 1/3 & 1/3 & 1/3 & 0 \end{pmatrix}, \quad \lim_{N\to\infty} P^N = \begin{pmatrix} 1/6 & 1/4 & 1/6 & 1/6 & 1/4 \\ 1/6 & 1/4 & 1/6 & 1/6 & 1/4 \\ 1/6 & 1/4 & 1/6 & 1/6 & 1/4 \\ 1/6 & 1/4 & 1/6 & 1/6 & 1/4 \\ 1/6 & 1/4 & 1/6 & 1/6 & 1/4 \end{pmatrix}.$$

This Markov chain converges to the distribution $\pi = (1/6, 1/4, 1/4, 1/6, 1/6)$. Thus sampling using this Markov chain is not uniform: it is more likely to sample networks G_2 and G_5 than the other networks.

In the second interpretation, the Markov chain contains repeated states. At each step, a pair of edges of G_i is randomly selected; if switching these edges results in another simple directed network, then the resulting network is the next state, else G_i is repeated. In this case the probability of transitioning to a neighbouring state is given by the probability of selecting the edge pair corresponding to a directed switch. Thus, if G_i and G_j differ by a directed switch the transition probability equals $1/M$ where M equals the number of edge pairs in G_i, i.e. $M = m(m-1)/2$. The probability of staying at G_i then is $(M-k(G_i))/M$ (Figure 2.4(b)). This Markov chain does converge to the uniform distribution, as can



(a) Transition probabilities no repeated states

(b) Transition probabilities repeated states

Figure 2.4.: (a) The state graph corresponding to the first interpretation of the switching model in Figure 2.3. (b) The state graph corresponding to the second interpretation of the switching model. The edge weights correspond to the transition probability from source to target state. The stationary distribution is displayed on the nodes.

be seen by taking the limit of the transition matrix.

$$P = \begin{pmatrix} 2/3 & 1/6 & 0 & 1/6 & 0 \\ 1/6 & 1/2 & 1/6 & 0 & 1/6 \\ 0 & 1/6 & 2/3 & 0 & 1/6 \\ 1/6 & 0 & 0 & 2/3 & 1/6 \\ 0 & 1/6 & 1/6 & 1/6 & 1/2 \end{pmatrix}, \qquad \lim_{N\to\infty} P^N = \frac{1}{5}\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Since the switching model is often used as a null-model, it is desirable to obtain a uniform sample from the network ensemble. In the example above, the Markov Chain that includes repeated states achieves this, and is the preferred model. The issue of non-uniform sampling for the switching model without repeated states has been addressed repeatedly in the literature [6, 79, 107]. When writing code to implement the switching model, it is important to be aware of this issue. From an algorithmic point of view, a particularly useful way of describing the difference between the two implementations is in terms of fixing the performed number of *switches* (no repeated states) versus fixing the performed number of *attempted switches* (repeated states) [79]. The pseudo-code below shows how little difference there is between these two implementations.

```
# Switching model excluding repeated states
# The number of switches is fixed to be N,
# we only increment i when a switch is made.
for(i in 1:N){
  edgePair = getRandomEdgePair(G)
  if(switchIsAllowed(G, edgePair)){
    G = switch(G, edgePair)
    i = i+1
  }
}
```

```
# Switching model including repeated states
# The number of attempted switches is fixed
# to be N, we increment i regardless of
# whether a switch is made or not.
for(i in 1:N){
  edgePair = getRandomEdgePair(G)
  if(switchIsAllowed(G, edgePair)){
    G = switch(G, edgePair)
  }
  i = i+1
}
```

From now on, when discussing switching models for simple directed networks, we will mean the switching model that includes repeated states. To avoid any ambiguity, we define the switching model as follows.

Definition 2.1.3. Let G be a simple directed network. The **switching model for G with respect to simple directed networks** is defined by a Markov chain starting at G. The states of this Markov chain are all the simple directed networks that have the same degree sequences as G. If there exists a directed switch between two states G_i and G_j then the transition probability p_{ij} is the probability of selecting the corresponding unique edge pair. There are $M = {}^{m(m-1)}/_{2}$ edge pairs to choose from, with m the total number of edges of G. The resulting probabilities are

$$p_{ij} = \begin{cases} \frac{1}{M} & \text{if there exists a directed switch between } G_i \text{ and } G_j \\ 1 - \frac{k(G_i)}{M} & \text{if } j = i \\ 0 & \text{otherwise} \end{cases} .$$

Example 2.1.2 shows that this Markov chain converges to the uniform distribution for a particular choice of G. A question that arises naturally is whether this switching model converges to the uniform distribution for any simple directed network? The remainder of this section addresses this question.

We show that the switching model as defined here almost always has the uniform distribution as its stationary distribution. This can be proven using Theorem 1.3.6, i.e. by checking that this finite Markov chain is irreducible, aperiodic and satisfies the simplified detailed balance equations.

2.1.1. Irreducibility

Irreducibility of a Markov chain is equivalent to the corresponding state graph being strongly connected (Definition 1.3.3). Figure 2.5 shows the most basic example of a network for which the switching model with respect to simple directed networks has a reducible Markov chain. The two realizations of the degree sequences $k^{in} = (1, 1, 1)$ and $k^{out} = (1, 1, 1)$ contain no edge pair that is allowed to be switched, since switching any edge pair produces a self loop. The probability of staying in each state equals 1 and the corresponding state graph is disconnected.

In [17] graphical degree sequences for which the Markov chain is irreducible are defined as *arc-swap sequences*. An algorithm is given to find whether a given degree sequence is an arc-swap sequence or not. Furthermore, an additional switch may be introduced to obtain a irreducible Markov chain. This will be discussed in detail in Section 2.2.1.

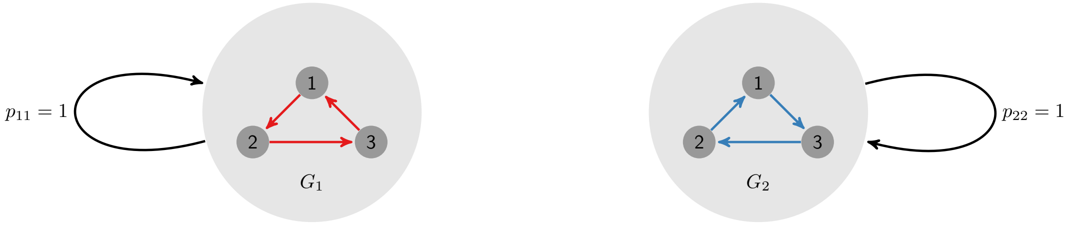


Figure 2.5.: The two simple directed networks that realize $k^{in} = (1, 1, 1)$ and $k^{out} = (1, 1, 1)$. Each network contains three pairs of edges, but no pair can be switched, since a switch would introduce a self-loop. The Markov chain corresponding to these sequences is reducible: its state graph is disconnected. Thus the switching model does not sample uniformly at random.

2.1.2. Aperiodicity

The class of simple directed networks for which the switching model has a periodic Markov chain is pathological. A Markov chain is trivially aperiodic if states have a non-zero probability to be repeated. This is the case for states in the switching model of most simple directed networks G. It is enough to require that G contains at least one vertex with total degree at least two. Lemma 2.1.4 below shows that this condition is necessary and sufficient.

Lemma 2.1.4. Let G be a simple directed network. The Markov chain of the switching model for G with respect to simple directed networks is aperiodic if and only if G contains a vertex v with total degree at least two.

Proof. We first show that the Markov chain is aperiodic when G contains a vertex v of total degree at least two. We show that the probability p_{ii} of repeating any state G_i is larger than zero. This is the probability of selecting an edge pair in G_i such that the network resulting from the corresponding directed switch, G_j, either equals G_i or is not simple. Since all G_i have the same degree sequence as G, v has degree at least two in G_i. If v has an incoming and an outgoing edge then switching this edge pair results in a network that contains a self-loop and thus is not simple. Otherwise, if v has either two incoming or two outgoing edges, the corresponding directed switches result in G_i.

To prove the reverse claim, we use proof by contrapositive: we show that if G does not contain a vertex v with total degree at least two, then the Markov chain is periodic. If G does not contain a vertex v with total degree at least two then G is a disjoint union of single vertices and single edges. We ignore the single vertices since the switching model leaves these invariant. Thus the interesting part of G is a collection of single edges $\{(s_1, t_1), (s_2, t_2), \ldots, (s_m, t_m)\}$. We can represent G as the ordered tuple $T = (t_1, t_2, \ldots, t_m)$. The set of simple directed networks with the same degree sequences corresponds to all permutations of T. For instance, if $(t_{i_1}, t_{i_2}, \ldots, t_{i_m})$ is a permutation of T, then the network with edge set $\{(s_1, t_{i_1}), (s_2, t_{i_2}), \ldots, (s_m, t_{i_m})\}$ is simple directed and has the same degree sequence as G.

A directed switch corresponds to a transposition of two elements in $(t_1, t_2, \ldots, t_m)$. The identity is an even permutation and can thus only be obtained as the composition of an even number of transpositions.

This precisely means that any sequence of networks in the Markov chain starting and ending at a network G_i has to be of even length. The chain is periodic with period two. □

An example of a simple directed network with vertex degree at most one and thus with 2-periodic state graph is shown in Figure 2.6.

2.1.3. Detailed balance equations

The transition probabilities of the switching model with respect to simple directed networks always satisfy the simplified detailed balance equations. This can easily be seen from the symmetry in the definition: if there exists a directed switch between networks G_i and G_j then $p_{ij} = {}^1\!/\!_M = p_{ji}$. In fact the construction of this chain corresponds exactly to the Metropolis chain for uniform sampling [83, 48].

To summarize: the Markov chain corresponding to the switching method with respect to simple directed networks *always* satisfies the simplified detailed balance equations and is

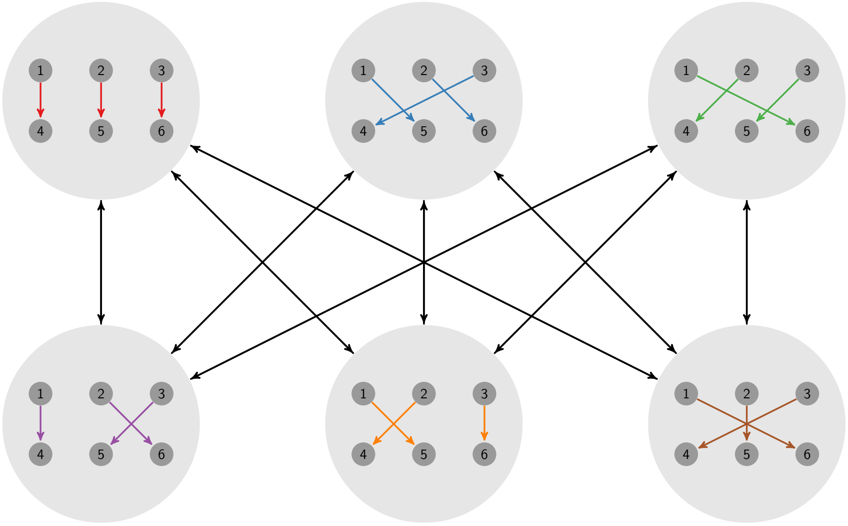


Figure 2.6.: The state graph for simple directed networks with degree sequences $k^{in} = (0, 0, 0, 1, 1, 1)$ and $k^{out} = (1, 1, 1, 0, 0, 0)$ is bipartite. The corresponding Markov chain is periodic with period two.

irreducible and aperiodic for *most* networks. In practice, all networks of interest result in an aperiodic, irreducible Markov chain.

The next section discusses the convergence properties of switching models with respect to different network classes. It also discusses a variation of the switching model for simple directed networks that is irreducible for *all* simple directed networks.

2.2. Switching models for different network classes

In this section we describe the switching model with respect to the network classes in Table 2.1. The switching model discussed in the previous section randomized networks from class $\mathcal{G}_3$. We first describe the switching model with respect to the other three classes of directed networks ($\mathcal{G}_4$, $\mathcal{G}_7$, $\mathcal{G}_8$) in Table 2.1. We then describe the switching model with respect to the four classes of undirected networks ($\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_5, \mathcal{G}_6$). Finally we discuss the convergence properties of the Markov chains corresponding to the switching model with respect to these classes.

Class	Name	Multiple	Directed	Self loops
$\mathcal{G}_1$	Simple undirected	No	No	No
$\mathcal{G}_2$	Undirected	No	No	Yes
$\mathcal{G}_3$	Simple directed	No	Yes	No
$\mathcal{G}_4$	Directed	No	Yes	Yes
$\mathcal{G}_5$	Undirected multigraphs without self loops	Yes	No	No
$\mathcal{G}_6$	Undirected multigraphs	Yes	No	Yes
$\mathcal{G}_7$	Directed multigraphs without self loops	Yes	Yes	No
$\mathcal{G}_8$	Directed multigraphs	Yes	Yes	Yes

Table 2.1.: Some basic classes of networks on which switching models can be defined. For each class of networks it needs to be determined separately whether or not the switching method converges to sampling from the uniform distribution.

Definition 2.1.3 can be generalized to describe switching models for all directed network classes in Table 2.1. In a sense, the switching model simplifies for these classes, since more of the edge pairs are allowed to be switched: that is when switched the resulting network is more likely to be of the same class since self-loops or multiple edges are now allowed.

Definition 2.2.1. Let G be a network in $\mathcal{G}_k$ where k equals $3, 4, 7$ or 8. The **switching model for G with respect to $\mathcal{G}_k$** is defined by a Markov chain starting at G. The states of this Markov chain are all the networks in $\mathcal{G}_k$ that have the same degree sequences as G. If there exists a directed switch between G_i and G_j then the transition probability p_{ij} is the probability of selecting an edge pair corresponding to this switch.

To formulate the switching model with respect to undirected networks we first define an undirected switch.

Definition 2.2.2. There exists a **switch** from the undirected network $G_i = (V_i, E_i)$ to the undirected network $G_j = (V_j, E_j)$ iff $V_j = V_i$, $E_i \neq E_j$ and there are two edges $\{x, y\}$ and $\{u, v\}$ in E_i such that either $E_j = (E_i \backslash \{\{x, y\}, \{u, v\}\}) \cup \{\{x, v\}, \{u, y\}\}$ (**switch 1**) or $E_j = (E_i \backslash \{\{x, y\}, \{u, v\}\}) \cup \{\{x, u\}, \{y, v\}\}$ (**switch 2**), see Figure 2.7.

Notice that if network G_i and G_j differ by switch 1, then $x \neq u$ and $y \neq v$, since $E_i \neq E_j$.

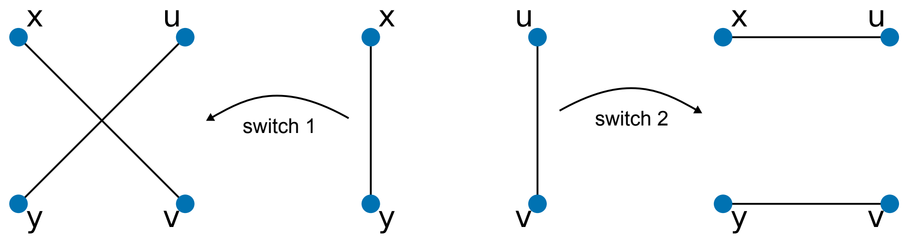


Figure 2.7.: For undirected networks, there are two possible switches for each pair of edges $\{x, y\}$ and $\{u, v\}$.

Similarly if G_i and G_j differ by switch 2 then $x \neq v$ and $y \neq u$.

In a sense, this definition of a switch is not precise. We could restrict to switch 1 only , since switch 2 is just switch 1 where one edge is labelled in reverse order. However, the reason that we *do mention* both switches is that it is important to realise that selecting an edge pair does not correspond to selecting a switch, there are *two* potential switches for each pair of edges. This is important for two reasons. Firstly, when implementing the switching algorithm for undirected networks, it is most likely that edges are stored with vertices in fixed order. In other words whenever we select the undirected edge $\{x, y\} = \{y, x\}$ we will get the edge as *the same* ordered pair of vertices. Thus, if only switch 1 is implemented, we only ever move by switch 1 and never by switch 2. This results in biased sampling, as discussed in Section 2.3. In fact, the implementation of the switching model for undirected networks in MFinder B.2.5 that was used in the famous motif paper [82] was implemented this way. *Thus, results obtained prior to its recent correction*[1]*, can not be relied on.* Secondly, in order to find the transition probabilities for the Markov chain, we need to take into account that switch 1 and switch 2 may result in different networks.

We now define switching models with respect to the undirected network classes in Table 2.1 ($\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_5, \mathcal{G}_6$).

Definition 2.2.3. Let G be a network from a specific class $\mathcal{G} \in \{\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_5, \mathcal{G}_6\}$ of undirected networks. The **switching model for G with respect to $\mathcal{G}$** is defined by a

[1] MFinder was updated on 28/05/2015 and can be downloaded from http://wws.weizmann.ac.il/mcb/UriAlon/download/network-motif-software.

Markov chain starting at G. The states of this Markov chain are all the networks that have the same degree sequence as G. If $G_i, G_j \in \mathcal{G}$ and there exists a switch between G_i and G_j then the transition probability p_{ij} is the probability of selecting an edge pair and picking switch 1 or switch 2 (with equal probability), corresponding to this switch.

Now that we have precise definitions for the switching model with respect to the classes in Table 2.1, we want to know if and under which conditions these switching models sample uniformly. As in Section 2.1, we proceed by finding the conditions under which the corresponding Markov chains are irreducible, aperiodic and satisfy the simplified detailed balance equations. Table 2.2 summarizes the results of this Section.

Class	Irreducible	Aperiodic	$p_{ij} = p_{ji}$	Uniform	Adjusted uniform version
$\mathcal{G}_1$	Yes [117, 38]	Yes	Yes	Yes	-
$\mathcal{G}_2$	No	Yes	No[4]	No	No
$\mathcal{G}_3$	No[3] [103, 17]	Yes[1]	Yes	No	Yes
$\mathcal{G}_4$	Yes [110, 103, 17]	Yes[2]	Yes	Yes	-
$\mathcal{G}_5$	Yes [37, 49]	Yes	No[4]	No	Yes
$\mathcal{G}_6$	Yes [37]	Yes	No[4]	No	Yes
$\mathcal{G}_7$	No[3]	Yes[1]	No[4]	No	Yes
$\mathcal{G}_8$	Yes	Yes[2]	No[4]	No	Yes

Table 2.2.: Properties of the Markov chains corresponding to switching methods with respect to the classes of networks defined in Table 2.1. [1]If the network contains at least one vertex with total degree two or more. [2]If the network contains at least one vertex with in-degree two or more or out-degree two or more. [3]The introduction of an additional switch will result in an irreducible Markov chain. [4]Acceptance probabilities can be introduced to ensure the detailed balance equations hold.

Some of the results in Table 2.2 can be found in the literature. For classes $\mathcal{G}_1, \mathcal{G}_4, \mathcal{G}_5$ and $\mathcal{G}_6$ it has been shown previously that the Markov chain is irreducible. We discuss these results in more detail below. We have already discussed the properties of the switching model with respect to class $\mathcal{G}_3$ in Section 2.1. Finally we will show that the Markov chains for the switching model with respect to $\mathcal{G}_2$ and $\mathcal{G}_7$ are not always irreducible and prove that the Markov chain of the switching model with respect to $\mathcal{G}_8$

is.

We have already seen in Lemma 2.1.4 that the Markov chain for the switching model with respect to $\mathcal{G}_3$ is aperiodic for practically all simple directed networks. Below we prove aperiodicity of the Markov chain corresponding to the switching model for *any* undirected network with respect to *any* of the classes $\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_5$ and $\mathcal{G}_6$. We also prove aperiodicity for the remaining directed classes of networks under very mild conditions on G.

Finally, we derive the transition probabilities and detailed balance equations for the classes in Table 2.2. In Section 2.1 we already showed that the switching model with respect to $\mathcal{G}_3$ satisfies the simplified detailed balance equations. We now show that the same holds for the switching models with respect to classes $\mathcal{G}_1$ and $\mathcal{G}_4$. However, the switching models with respect to classes $\mathcal{G}_2, \mathcal{G}_5, \mathcal{G}_6, \mathcal{G}_7$ and $\mathcal{G}_8$ do not satisfy the simplified detailed balance conditions. For these classes, we derive the stationary distributions of the corresponding switching models, and show that these are not quite uniform. We then introduce small modifications to these switching models to ensure uniform sampling.

2.2.1. Irreducibility

Irreducibility of a Markov chain is equivalent to the corresponding state graph Ψ being strongly connected (Definition 1.3.3). Since both the directed and undirected switch are symmetric moves, it is enough to show that Ψ is connected. This is often the hardest property to prove in demonstrating that the chain has a stationary distribution. In fact, as can be seen from Table 2.2, it does not always hold.

The following two statements are equivalent:

1. The state graph corresponding to the switching method for a graph $G = (V, E) \in \mathcal{G}_i$ with respect to class $\mathcal{G}_i$ is connected.

2. For any two graphs $G' = (V, E')$, and $G'' = (V, E'')$ with degree sequence(s) equal to those of G, there exists a sequence of graphs $G' = G_0, \ldots, G_k = G''$ with $G_j \in \mathcal{G}_i$ for every j and $|E_j \Delta E_{j+1}| = 4$.

The requirements that each G_j has degree sequence equal to G, is a graph of class $\mathcal{G}_i$ and that $|E_j \Delta E_{j+1}| = 4$, together imply that there exists a switch between G_j and G_{j+1} with respect to $\mathcal{G}_i$.

We first discuss class $\mathcal{G}_8$ since in a sense this is the simplest case for the directed network classes. Afterwards we will discuss known results from the literature and finally we investigate the situation for classes $\mathcal{G}_2$ and $\mathcal{G}_7$.

Lemma 2.2.4. Let $G = (V, E), G' = (V, E') \in \mathcal{G}_8$ be distinct graphs with equal in-degree and out-degree sequences. There exists a sequence of graphs $G = G_0, \ldots, G_k = G'$ such that $G_i \in \mathcal{G}_8$ and $|E_i \Delta E_{i+1}| = 4$ for all $i \in (0, \ldots k-1)$.

Proof. The edge set difference $E \Delta E'$ has even cardinality: $|E \Delta E'| = 2\kappa$ (Corollary 1.2.3). We prove the lemma by induction on κ. Let $\kappa > 2$ and let $(v_1, v_2) \in E \backslash E'$, then there exist edges (v_1, v_3) and (v_4, v_2) in $E \backslash E'$ with $v_3 \neq v_2$ and $v_4 \neq v_1$. Let G^* be the graph with vertices V and edges $E^* = (E' \backslash \{(v_1, v_3), (v_4, v_2)\}) \cup \{(v_1, v_2), (v_4, v_3)\}$ then $G^* \in \mathcal{G}_8$, $|E^* \Delta E'| = 4$ and $|E^* \Delta E| \leq 2\kappa - 2$. □

For directed network classes different from $\mathcal{G}_8$ the proof of Lemma 2.2.4 no longer works: there is no guarantee that G^* is a graph from the same class. In other words, the switch from G' to G^* might not be allowed, e.g. if it introduces multiple edges or self-loops.

The proof of Lemma 2.2.4 holds equally well for $\mathcal{G}_6$, undirected multigraphs that may include self-loops. Again, regardless of the switch, the resulting graph will be a member of $\mathcal{G}_6$. For the other classes of networks in Table 2.2 the question is more complicated to answer.

Statements analogous to Lemma 2.2.4 with respect to classes $\mathcal{G}_1, \mathcal{G}_3, \mathcal{G}_4, \mathcal{G}_5, \mathcal{G}_6$ have previously been discussed in the literature. In 1981, both Taylor [117, Theorem 2.1] and Eggleton and Holton [38, Theorem 5] showed that for any two finite simple undirected networks G and H with equal degree sequence, there is a sequence of switches from G to H (i.e. the Markov chain for $\mathcal{G}_1$ is irreducible). Hakimi [49], and Eggleton and Holton [37] had previously proven this result with respect to classes $\mathcal{G}_5$ and $\mathcal{G}_6$.

In 1963 Ryser [110, Theorem 3.1] proved that for any pair of binary matrices, A and B, with equal row sums and column sums, there is a sequence of binary matrices $A = M_1, \ldots, M_k = B$ such that consecutive pair of matrices differs in exactly one switch (interchange). This result implies that the Markov chain for the switching model with respect to $\mathcal{G}_4$ is irreducible: directed networks with fixed degree sequences have binary adjacency matrices with equal row sums and column sums and a directed switch in a network corresponds to a switch in its adjacency matrix. Rao et al. [103] later gave an alternative proof. Furthermore they showed that for the class of simple directed networks (i.e. $\mathcal{G}_3$), the Markov chain of the switching model is not always irreducible. This is also shown in [17] where the class of networks for which the Markov chain is reducible is classified. The simplest example is the directed 3-cycle, as shown in Figure 2.5.

The Markov chain corresponding to the switching model with respect to class $\mathcal{G}_7$ is not always irreducible. The same issue arises as for $\mathcal{G}_3$, it is not always possible to reverse the direction of a directed 3-cycle. Again the simplest example is the graph consisting of just a directed 3-cycle as shown in Figure 2.5.

The switching models with respect to classes $\mathcal{G}_3$ and $\mathcal{G}_7$ may be altered to include *triangle reorientations* [17, 3-cycle reorientation], [103, hexagonal move], [108, triangle swap].

Definition 2.2.5. There exists a **triangle reorientation** from network $G_i = (V_i, E_j)$ to $G_j = (V_j, E_j)$ iff $V_j = V_i$, $E_i \neq E_j$ and there are three edges (x, y), (y, z) and (z, x) in E_i such that $E_j = (E_i \backslash \{(x, y), (y, z), (z, x)\}) \cup \{(x, z), (z, y), (y, x)\}$.

If $G_i = (V, E_i)$ and $G_j = (V, E_j)$ are distinct directed networks with equal degree sequence, then they either differ by a directed switch or they differ by a triangle move or they differ by more than six edges. That is, G_i and G_j can not differ by a directed switch *and* a triangle reorientation at the same time. Hence, the definition of the triangle enriched switching model below is well-defined.

Definition 2.2.6. Let G be a network in $\mathcal{G}_k$ where k equals 3 or 7. The **triangle enriched switching model** for G with respect to $\mathcal{G}_k$ is defined by a Markov chain starting at G. The states of this Markov chain are all the networks in G_k that have the same degree sequences as G. If there exists a directed switch between G_i and G_j then the transition probability p_{ij} is the probability of selecting an edge pair corresponding to this switch. If there exists a triangle reorientation between G_i and G_j then the transition probability p_{ij} is the probability of selecting any one of the three edge pairs corresponding to the triangle reorientation.

The Markov chain of the triangle enriched switching model with respect to $\mathcal{G}_3$ is irreducible [16, Lemma 3.4]. The proof of this Lemma can be extended to apply to the easier case with respect to $\mathcal{G}_7$.

The triangle enriched switching model is not just a theoretical construction. It can be implemented as follows for networks in $\mathcal{G}_3$ or $\mathcal{G}_7$. If $G_i = (V, E_i)$ is the current state of the Markov chain, then we may obtain the next state in the chain as follows. Randomly select two edges (x, y) and (u, v) from E_i. If either $x = u$ or $y = v$ do nothing (i.e. repeat the current state). If $y = u$ and $(v, x) \in E_i$, reorient the triangle $(x, y), (y, v), (v, x)$ provided that the resulting graph is valid (i.e. it does not contain multiple edges if switching w.r.t. $\mathcal{G}_3$)). Similarly if $x = v$ and $(y, u) \in E_i$, attempt to reorient the triangle $(x, y), (y, u), (u, x)$. Otherwise, all four vertices are distinct and we attempt the regular directed switch. Notice that this works since a triangle reorientation is uniquely defined by two edges.

The Markov chain corresponding to the switching model with respect to class $\mathcal{G}_2$ is not always irreducible. Below we list a few networks for which this Markov chain is reducible. We are currently not aware of conditions on $G \in \mathcal{G}_2$ that would ensure irreducibility of this Markov chain.

Example 2.2.7. The Markov chain of the switching method for the n-cycle C_n (Definition 1.1.18) with respect to $\mathcal{G}_2$ is reducible. The degree sequence of C_n equals $k = (2, \ldots, 2)$. A different graph realization of this degree sequence is the a graph with a self loop at each vertex. For this graph no edge pair is allowed to be switched since this would introduce a multiple edge. Indeed, this Markov chain is reducible (see Figure 2.8(a)).

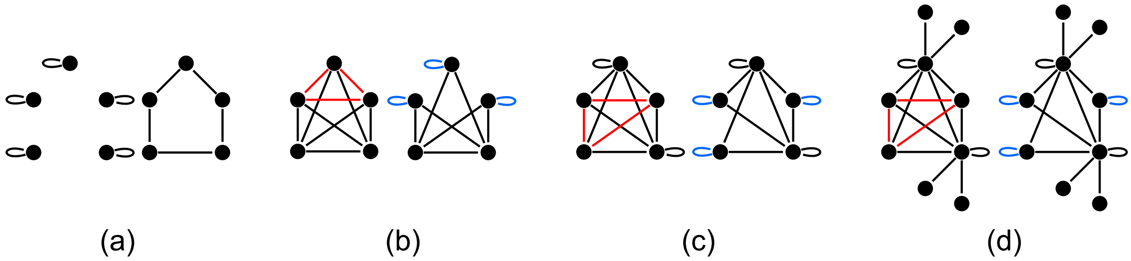


Figure 2.8.: Examples of graphs for which the switching model with respect to $\mathcal{G}_2$ has reducible Markov chain. The networks on the left in (a)-(c) do not contain any edge pairs that can be switched without creating multiple edges. However there is at least one different graph realization of the same degree sequence as depicted on the right. (a) Two graph realizations of the degree sequence $k = (2, 2, 2, 2, 2)$: a network with five vertices with self-loops and the 5-cycle C_5. (b) Two graph realizations of the degree sequence $k = (4, 4, 4, 4, 4)$: the 5-clique K_5 and the network where the red triangle has been removed and three self-loops have been inserted. (c) Two graph realizations of the degree sequence $k = (6, 4, 4, 4, 6)$. (d) Two graph realizations of the degree sequence $k = (1, 1, 8, 4, 4, 4, 8, 1, 1)$. The only switches in the network on the left that do not introduce multiple edges are between edges adjacent to vertices of degree 1. Such switches result in isomorphic networks but never in the network on the right.

Example 2.2.8. The Markov chain of the switching method for the n-clique K_n (Definition 1.1.19) with $n > 2$, with respect to $\mathcal{G}_2$ is reducible. None of the edge pairs of K_n are allowed to be switched, since this would introduce a multiple edge. However, there exist other graph realizations of the same degree sequence in $\mathcal{G}_2$. Indeed, the network where one triangle of K_n is removed and three self-loops on the corresponding vertices are added, is such a network (see Figure 2.8(b)).

Example 2.2.9. Let G be a network in $\mathcal{G}_2$ on $n > 3$ vertices with an edge between each pair of vertices. In other words K_n is a subgraph of G. Furthermore, let at least one and at most $n - 3$ of these vertices have a self-loop. The Markov chain corresponding to the switching model for G with respect to $\mathcal{G}_2$ is reducible. No edge pair of G is allowed to be switched, as this would introduce multiple edges. However, there is at least one other network in $\mathcal{G}_2$ with the same degree sequence. Indeed, since there are at least three vertices in G without self-loops, a network that differs from G by replacing the connections between three such vertices by three self-loops, is such a network (see Figure 2.8(c)).

Example 2.2.10. Let G be a network in $\mathcal{G}_2$ with $n + n'$ vertices $V \cup W$ where the vertices V form an n-clique. Let at least one and at most $n - 3$ of these vertices have a self-loop. Let each vertex $w \in W$ have degree one and be connected to some vertex in V that has a self-loop. The Markov chain for the switching model of G with respect to $\mathcal{G}_2$ is reducible. The only edge pairs in G that can be switched without introducing multiple edges are of the form $\{v_i, w_j\}$. Such switches always result in an isomorphic network. However, there exists at least one other realization of the degree sequence of G that is not isomorphic to G. A network that differs from G by replacing a triangle in the n-clique by three self-loops is such a network (see Figure 2.8(d)).

One can attempt to define a 'triangle enriched switching method' with respect to $\mathcal{G}_2$. The additional triangle move would replace a triangle with three self-loops. However, the reverse move, from three self-loops to a triangle, is not uniquely defined by two edges. Selecting a third loop to attempt the reverse move will cause an asymmetry for the detailed balance equations.

It is reasonable to expect that nearly all undirected networks obtained from real world data will have irreducible Markov chains. In each of the Examples 2.2.7, 2.2.8, 2.2.9 and 2.2.10 the reducibility of the Markov chain arises from the fact that no sequence of switches can be found that replaces a triangle by three self-loops, e.g. any such sequence of switches would introduce a multiple edge at a certain point. Below we show that this problem is resolved when a graph contains just one additional edge with one of two basic properties.

Example 2.2.11. Let $G = (V, E) \in \mathcal{G}_2$ be a graph that contains the triangle $\{v_1, v_2\}$, $\{v_2, v_3\}$, $\{v_3, v_1\}$ such that $G' = (V, E')$, the graph where this triangle has been replaced by three self-loops

$$E' = (E \backslash \{\{v_1, v_2\}, \{v_2, v_3\}, \{v_3, v_1\}\}) \cup \{\{v_1, v_1\}, \{v_2, v_2\}, \{v_3, v_3\}\},$$

is also in $\mathcal{G}_2$.

First, if G contains a vertex v_i that is connected to at least one and at most two of the vertices v_1, v_2 and v_3, then there is a sequence of switches from G to G'. To see this: Assume that v_i is connected to v_1 but not to v_3. The following sequence of switches results in G' (Figure 2.9).

$$\begin{aligned}
\{v_i, v_1\}\{v_1, v_3\} &\rightarrow \{v_1, v_1\}\{v_3, v_i\} \\
\{v_1, v_2\}\{v_2, v_3\} &\rightarrow \{v_2, v_2\}\{v_1, v_3\} \\
\{v_1, v_3\}\{v_3, v_i\} &\rightarrow \{v_3, v_3\}\{v_1, v_i\}
\end{aligned}$$

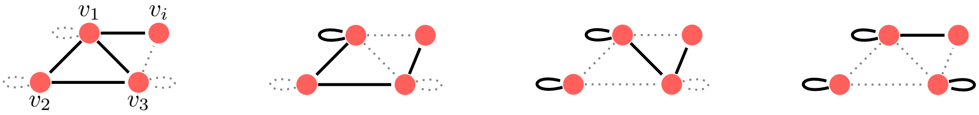


Figure 2.9.: The sequence of switches that replaces a triangle by three self-loops in the presence of an edge $\{v_1, v_i\}$ with v_i not connected to v_3. Solid lines indicate edges in the graph, dashed lines indicate that there is no edge.

Second, if G contains an edge $\{v_i, v_j\}$ such that there is an edge in the triangle that has at most one connection to $\{v_i, v_j\}$ then there exists a sequence of switches from G to G'. To see this: Assume that $\{v_i, v_j\}$ has at most one connection to $\{v_2, v_3\}$, specifically assume that neither v_i nor v_j is connected to v_3 and furthermore that v_i is not connected to v_2. The following sequence of switches transforms G to G' (Figure 2.10).

$$\begin{aligned}
\{v_2, v_3\}\{v_i, v_j\} &\rightarrow \{v_2, v_i\}\{v_3, v_j\} \\
\{v_1, v_2\}\{v_1, v_3\} &\rightarrow \{v_1, v_1\}\{v_2, v_3\} \\
\{v_2, v_3\}\{v_2, v_i\} &\rightarrow \{v_2, v_2\}\{v_3, v_i\} \\
\{v_3, v_i\}\{v_3, v_j\} &\rightarrow \{v_3, v_3\}\{v_i, v_j\}
\end{aligned}$$

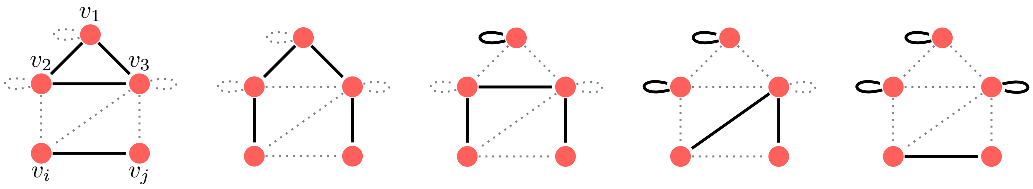


Figure 2.10.: The sequence of switches that replaces a triangle by three self-loops in the presence of an edge $\{v_i, v_j\}$ that has at most one connection to one of the edges of the triangle. Solid lines indicate edges in the graph, dashed lines indicate that there is no edge.

To summarize, the Markov chain corresponding to the switching models with respect to classes $\mathcal{G}_1, \mathcal{G}_4, \mathcal{G}_5, \mathcal{G}_6$ and $\mathcal{G}_8$ is always irreducible. For classes $\mathcal{G}_3$ and $\mathcal{G}_7$ the Markov chain might be reducible, however the triangle enriched switching models with respect to classes $\mathcal{G}_3$ and $\mathcal{G}_7$ do always have an irreducible Markov chain. Finally the Markov chain corresponding to the switching method with respect to $\mathcal{G}_2$ is not always irreducible, but it is reasonable to expect it to be irreducible for most undirected networks.

2.2.2. Aperiodicity

Let $\mathcal{G}$ be one of the classes $\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_5$ and $\mathcal{G}_6$. The switching model for G with respect to $\mathcal{G}$ is aperiodic as shown in the lemma below.

Lemma 2.2.12. Let G be any undirected network. If $G \in \mathcal{G}_i$, then the switching model for G with respect to $\mathcal{G}_i$ is aperiodic for $i \in \{1, 2, 5, 6\}$.

Proof. Let $\{x, y\}$ and $\{u, v\}$ be distinct edges in G. There are seven different configurations (Figure 2.11): one where $x = y = u = v$ (A), three with two distinct vertices (B)-(D), two on three vertices (E) and (F) and one on four distinct vertices (G). Note that (H) and (I) are equal to (G) up to relabelling of the vertices.

We now show for each of these configurations, that if G contains the configuration the state graph of the switching model is aperiodic.

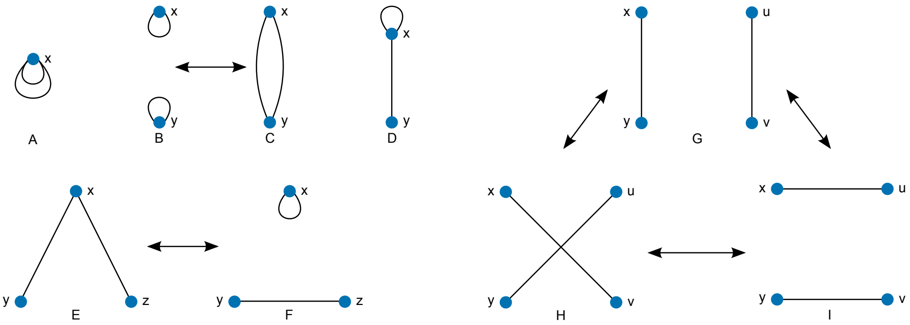


Figure 2.11.: Seven different configurations of two edges in an undirected network.

A: If G contains two edges at $\{x, x\}$ then switching the two edges does not alter the network and thus corresponds to a self-loop in the state graph.

B: If G contains configuration B then we are either switching with respect to $\mathcal{G}_2$ or with respect to $\mathcal{G}_6$. In the former case, we are not allowed to switch the two edges, thus there is a self-loop in the state graph. In the latter case, switching results in configuration C, call the new network G'. Switch 2 results in configuration B, but switch 1 does not alter G', resulting in a self-loop in the state graph.

C: See configuration B.

D: Switching the two edges does not alter the network, and thus there is a self-loop in the state graph.

E: Switch 1 for edges $\{x, y\}$ and $\{x, z\}$ does not alter the network.

F: Switching the two edges results in configuration E and the resulting state has a self-loop in the state graph.

G-I: If G contains configuration G-I, i.e. if x, y, u and v are all distinct, then we distinguish between the following two cases. (i) The induced subgraph of G on x, y, u and v contains only two edges. In this case, there is a path of length three in the state graph, as shown in Figure 2.11 - thus the state graph is aperiodic since gcd(3,2)=1. (ii) The induced subgraph of G on x, y, u and v contains more than two edges. For class $\mathcal{G}_5$ or $\mathcal{G}_6$ this does not change the argument of case (i). However for class $\mathcal{G}_1$ or $\mathcal{G}_2$, either switch 1 or switch 2 introduces a multiple edge, and we thus find a self-loop in the state graph, since the switch that introduces a multiple edge is not allowed.

□

A similar result holds for the directed network classes: $\mathcal{G}_3, \mathcal{G}_4, \mathcal{G}_7$, and $\mathcal{G}_8$.

Lemma 2.2.13. Let G be a directed network containing at least one vertex with total degree greater than or equal to two. If $G \in \mathcal{G}_i$, then the switching model with respect to $\mathcal{G}_i$ is aperiodic for $i = \{3, 7\}$.

Proof. For class $\mathcal{G}_3$ this result was proved in Lemma 2.1.4. This proof can be seen to hold for class $\mathcal{G}_7$ as well. □

For the two classes of networks that include self-loops, we have the following Lemma.

Lemma 2.2.14. Let $G \in G_i$ with $i \in \{4, 8\}$. The switching model with respect to G is aperiodic if and only if G contains at least one vertex with out-degree or in-degree greater than or equal to two.

Proof. If G contains a vertex with out-degree or in-degree greater or equal to two, then switching two incoming or two outgoing edges does not alter G and the Markov chain is aperiodic.

To prove the reverse claim, we use proof by contraposition. We show that if G does not contain a vertex v with in-degree or out-degree at least two, then the Markov chain is periodic. In this case, G is a union of chains of edges $(v_{i_1}, v_{i_2}), (v_{i_2}, v_{i_3}), \ldots, (v_{i_{k-1}}, v_{i_k})$, and vertices with self-loops, and singleton vertices. We ignore the single vertices since the switching model leaves these invariant. Thus the interesting part of G is a collection of single edges $\{(s_1, t_1), (s_2, t_2), \ldots, (s_m, t_m)\}$ where t_i may be equal to s_i. We can represent G as the ordered tuple $T = (t_1, t_2, \ldots, t_m)$. The set of directed networks with the same degree sequences corresponds to all permutations of T.

A directed switch corresponds to a transposition of two elements in $(t_1, t_2, \ldots, t_m)$. The identity is an even permutation and can thus only be obtained as the composition of an even number of transpositions. □

2.2.3. Detailed balance equations

In this section we derive the distributions π that satisfy the detailed balance equations for the switching model with respect to each of the network classes in Table 2.1.

We have already seen that the *simplified* detailed balance equations hold for the switching model with respect to $\mathcal{G}_3$. The reason that they are satisfied is two-fold, firstly the probability of selecting an edge pair corresponding to a switch was $1/M$ for any edge pair and secondly the edge pair corresponding to a switch was *unique*. In other words, if there exists a directed switch between two directed networks G_i and G_j, then $p_{ij} = 1/M = p_{ji}$. The same is true for the switching model with respect to $\mathcal{G}_4$.

The simplified balance equations also hold for the switching model with respect to $\mathcal{G}_1$. The transition probability p_{ij} between two distinct networks G_i and G_j is non-zero if and only if there exists a switch between the two networks. This switch corresponds to the selection of an edge pair together with either switch 1 or switch 2. Thus the transition probabilities p_{ij} and p_{ji} are both equal to $1/2M$ and the simplified detailed balance equations hold.

For the remaining classes, the simplified detailed balance equations no longer hold. We show that the detailed balance equations hold for distributions that are close to being uniform. Hence, by Theorem 1.3.6, these distributions are the stationary distributions of the corresponding switching models. Our approach is to first derive the transition probabilities of the switching model with respect to a given network class, then to define a candidate stationary distribution and to show that the detailed balance equations hold. Finally for each of these classes we introduce an adjusted switching model that does converge to the uniform distribution. We now discuss the general framework for adjusting these switching models.

The adjusted switching models make use of *acceptance probabilities*, which were also used in [31]. We will be introducing acceptance probabilities to force the simplified detailed balance equations to hold.

Definition 2.2.15. We may adjust a Markov chain $\{X_i\}_{i \in I}$ by introducing **acceptance probabilities** a_{ij}, $i \neq j \in I$. Let p_{ij} be the transition probabilities of the original Markov chain. We obtain a new Markov chain $\{X'_i\}$ by accepting a move from X_i to X_j with probability a_{ij}. The transition probabilities of this Markov chain are given by

$$p'_{ij} = \begin{cases} p_{ij} a_{ij} & \text{if } i \neq j \\ 1 - \sum_{k, k \neq i} p'_{ik} & \text{if } i = j. \end{cases}$$

We now derive the transition probabilities of the switching model with respect to class $\mathcal{G}_2$.

Lemma 2.2.16. Let G_i and G_j be two networks in $\mathcal{G}_2$ with equal degree sequences and let α_k equal the number of self-loops in G_k. The transition probability p_{ij} from G_i to G_j is given by

$$p_{ij} = \begin{cases} \frac{1}{2M} & \text{if } |E_i \Delta E_j| = 4 \text{ and } \alpha_i = \alpha_j \text{ or } \alpha_i = \alpha_j - 1 \\ \frac{1}{M} & \text{if } |E_i \Delta E_j| = 4 \text{ and } \alpha_i = \alpha_j + 1 \\ 1 - \sum_{k, k \neq i} p_{ik} & \text{if } j = i \\ 0 & \text{otherwise.} \end{cases}$$

As usual $M = \frac{m(m-1)}{2}$ with m the number of edges of G_i.

Proof. When there exists a switch between G_i and G_j, or in other words, when $|E_i \Delta E_j| = 4$, there are exactly two possible configuration for $E_i \Delta E_j$, see Figure 2.12(b) and (c). If $\alpha_i = \alpha_j - 1$ then the edge set difference looks like configuration (b) with the red edge pair in E_i and the blue edge pair in E_j. The probability of selecting the red edge pair equals $1/M$ as usual. Either switch 1 or switch 2, but not both, results in network G_j and thus $p_{ij} = 1/2M$. If $\alpha_i = \alpha_j$, then the edge set difference looks like configuration (c). Again either switch 1 or switch 2, but not both, results in G_j and thus $p_{ij} = 1/2M$.

Finally, if $\alpha_i = \alpha_j + 1$, the edge set difference looks like configuration (b) with the blue edges in E_i. In this case, both switch 1 and switch 2 result in G_j, and thus $p_{ij} = {}^1\!/\!_M$. □

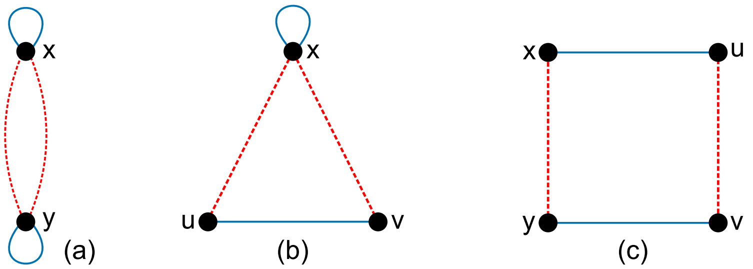


Figure 2.12.: Let $G_i = (V, E_i)$ and $G_j = (V, E_j)$ be two undirected (multi)graphs with the same degree sequence. If their symmetric edge set difference, $E_i \Delta E_j$, consists of four edges, then there are exactly three possible configurations. These configurations are shown above. Notice that the difference between the number of self-loops α_i in G_i and number of self-loops α_j in G_j is at most two.

Lemma 2.2.17. Let $G \in \mathcal{G}_2$. For any $G_i \in \mathcal{G}_2$ with the same degree sequence as G, let α_i equal the number of self-loops in G_i. Let $\Pi_i = \frac{1}{2}^{\alpha_i}$ and let $\pi_i = {}^{\Pi_i}\!/\!_{\sum_j \Pi_j}$. Then the detailed balance equations hold for π.

Proof. If there exists a switch between G_i and G_j, then the difference in number of self-loops α_i in G_i and number of self-loops α_j in G_j is at most one. If $\alpha_i = \alpha_j$ we find $\pi_i p_{ij} = \pi_j p_{ji}$ trivially. If $\alpha_i \neq \alpha_j$ we may assume $\alpha_i = \alpha_j + 1$ and

$$\begin{aligned}
\pi_i p_{ij} &= {}^1\!/\!_{\sum_k \Pi_k} \left({}^1\!/\!_2\right)^{\alpha_i} {}^1\!/\!_M \\
&= {}^1\!/\!_{\sum_k \Pi_k} \left({}^1\!/\!_2\right)^{\alpha_j+1} {}^1\!/\!_M \\
&= {}^1\!/\!_{\sum_k \Pi_k} \left({}^1\!/\!_2\right)^{\alpha_j} {}^1\!/\!_{2M} \\
&= \pi_j p_{ji}.
\end{aligned}$$

□

We define the **adjusted switching model** with respect to $\mathcal{G}_2$ as the Markov chain obtained by introducing acceptance probabilities a_{ij}, where $a_{ij} = {}^1\!/\!_2$ if $\alpha_i = \alpha_j + 1$ and $a_{ij} = 1$ otherwise. The adjusted switching model satisfies the simplified detailed balance equations and hence converges to the uniform distribution (if the Markov chain is irreducible). To implement these acceptance probabilities we need to accept each switch of the form $\{x, x\}, \{u, v\}$ to $\{x, u\}, \{x, v\}$ with probability 0.5 or else repeat the current state.

Next we look at class $\mathcal{G}_5$: undirected multigraphs without self-loops. Networks in this class may have multiple edges. This causes the transition probabilities to differ from those in Lemma 2.2.16; there may be several edge pairs that, when switched, result in the same network.

Lemma 2.2.18. Let G_i and G_j be two networks in $\mathcal{G}_5$ with equal degree sequence. The transition probability p_{ij} from G_i to G_j is given by

$$p_{ij} = \begin{cases} \frac{A^i_{xy}A^i_{uv}}{2M} & \text{if } E_j = E_i \backslash \{\{x, y\}, \{u, v\}\} \cup \{\{x, v\}, \{u, y\}\} \\ & \text{or } E_j = E_i \backslash \{\{x, y\}, \{u, v\}\} \cup \{\{x, u\}, \{y, v\}\} \\ 1 - \sum_{k, k \neq i} p_{ik} & \text{if } i = j \\ 0 & \text{otherwise,} \end{cases}$$

where $[A^i_{kl}]$ is the weighted adjacency matrix of the network G_i.

Proof. If there exists a switch between G_i and G_j, then the configuration of their edge set difference $E_i \Delta E_j$ is the configuration in Figure 2.12(c). Without loss of generality we may assume that the red edges $\{x, y\}$ and $\{u, v\}$ are in E_i. The transition probability p_{ij} depends on the multiplicity of the edges $\{x, y\}$ and $\{u, v\}$ in G_i. It is not hard to see that there are $A^i_{xy}A^i_{uv}$ distinct edge pairs $\{x, y\}$ and $\{u, v\}$. Either switch 1 or switch 2, but not both, results in G_j and thus $p_{ij} = A^i_{xy}A^i_{uv}/2M$. □

Lemma 2.2.19. Let $G \in \mathcal{G}_5$ be an undirected multigraph without self-loops. For any $G_i \in \mathcal{G}_5$ with the same degree sequence as G, let

$$\beta_i = \frac{1}{\prod_{k<l} A^i_{kl}!}$$

where $[A^i_{kl}]$ is the weighted adjacency matrix corresponding to G_i. Then $\pi_i = \beta_i / \sum_r \beta_r$ is the stationary distribution for the switching model for G with respect to $\mathcal{G}_5$.

Proof. Using the transition probabilities from Lemma 2.2.18 we now derive the detailed balance equations. Let $G_i = (V, E_i) \in \mathcal{G}_5$ and $G_j = (V, E_j) \in \mathcal{G}_5$ with the same degree sequence as G. Without loss of generality we assume that $E_j = E_i \backslash \{\{x,y\},\{u,v\}\} \cup \{\{x,v\},\{u,y\}\}$. Let $A^i_{\{rs\}}$ equal A^i_{rs}. Hence $A^i_{\{rs\}}$ is also equal to A^i_{sr}, since the network is undirected. Also let,

$$K_i = \frac{1}{2M}\frac{1}{\sum_r \beta_r}\frac{1}{\prod_{k<l,\{k,l\}\notin E_i \Delta E_j} A^i_{\{kl\}}!}$$

and notice that K_i equals K_j. Then,

$$\begin{aligned}
\pi_i p_{ij} &= \frac{\beta_i}{\sum_r \beta_r}\frac{A^i_{\{xy\}}A^i_{\{uv\}}}{2M} \\
&= K_i \frac{A^i_{\{xy\}}A^i_{\{uv\}}}{A^i_{\{xy\}}!A^i_{\{uv\}}!A^i_{\{xv\}}!A^i_{\{uy\}}!} \\
&= K_i \frac{1}{(A^i_{\{xy\}}-1)!(A^i_{\{uv\}}-1)!A^i_{\{xv\}}!A^i_{\{uy\}}!} \\
&= K_i \frac{1}{A^j_{\{xy\}}!A^j_{\{uv\}}!(A^j_{\{xv\}}-1)!(A^j_{\{uy\}}-1)!} \\
&= K_j \frac{A^j_{\{xv\}}A^j_{\{uy\}}}{A^j_{\{xy\}}!A^j_{\{uv\}}!A^j_{\{xv\}}!A^j_{\{uy\}}!} \\
&= \pi_j p_{ji}
\end{aligned}$$

□

We define the **adjusted switching model** with respect to $\mathcal{G}_5$ as the Markov chain obtained from the switching model by introducing acceptance probabilities a_{ij}. If there exists a directed switch between G_i and G_j, i.e. if $E_j = E_i \backslash \{\{x,y\},\{u,v\}\} \cup \{\{x,v\},\{u,y\}\}$, then we let $a_{ij} = {}^1/_{A^i_{xy}A^i_{uv}}$ otherwise $a_{ij} = 1$. These acceptance probabilities ensure the simplified detailed balance equations are satisfied and hence imply convergence to the uniform distribution.

Next, we look at the class of undirected multigraphs, $\mathcal{G}_6$. This class of networks has the most complicated transition probabilities.

Lemma 2.2.20. Let G_i and G_j be two networks in $\mathcal{G}_6$ with equal degree sequence. The transition probability p_{ij} from G_i to G_j is given by

$$p_{ij} = \begin{cases} \frac{A^i_{xy}A^i_{uv}}{M} & \text{if } |E_i \Delta E_j| = 4 \text{ and } \alpha_i > \ \alpha_j, \\ \frac{A^i_{xy}A^i_{uv}}{2M} & \text{if } |E_i \Delta E_j| = 4 \text{ and } \alpha_i = \alpha_j \text{ or } \alpha_i = \alpha_j - 1, \\ \frac{A^i_{xy}(A^i_{uv}-1)}{4M} & \text{if } |E_i \Delta E_j| = 4 \text{ and } \alpha_i = \alpha_j - 2, \\ 0 & \text{otherwise.} \end{cases}$$

where $\{x, y\}$ and $\{u, v\}$ are edges in $E_i \backslash E_j$.

Proof. If there exists a switch between G_i and G_j, then the edge set difference $E_i \Delta E_j$ corresponds to one of the three configurations in Figure 2.12.

If $\alpha_i > \alpha_j$, then the edge set difference looks like configuration (a) or (b) with the blue edges in E_i. In case of configuration (a), the probability of switching to G_j corresponds to the probability of selecting an edge pair $\{x, x\}, \{y, y\}$, which indeed is $A^i_{xx}A^i_{yy}/M$. We do not need to divide this by 2, since both switch 1 and switch 2 result in G_j. Similarly, for configuration (b), the probability of selecting an edge pair $\{x, x\}$ and $\{u, v\}$ equals $A^i_{xx}A^i_{uv}/M$ and we do not need to divide this by 2 since both switch 1 and switch 2 result in G_j.

For $\alpha_i = \alpha_j$, see the argument of Lemma 2.2.18.

If $\alpha_i = \alpha_j - 1$ then the edge set difference looks like configuration (b) with the red edge pair in E_i. The probability of selecting this edge pair is $A^i_{xu}A^i_{xv}/M$. To find the transition probability we have to divide this probability by 2, since only switch 1 or switch 2 but not both, results in G_j.

Finally if $\alpha_i = \alpha_j - 2$, the edge set difference looks like configuration (a) with the red edges in E_i. Here the probability of selecting an edge pair $\{x, y\}$ and $\{x, y\}$ equals $A^i_{xy}(A^i_{xy}-1)/2M$, since there are $A^i_{xy}(A^i_{xy}-1)/2$ such edge pairs. This probability needs to be divided by 2, since only switch 1 or switch 2 but not both results in G_j. Indeed we find $p_{ij} = A^i_{xy}(A^i_{xy}-1)/4M$. □

Lemma 2.2.21. Let $G \in \mathcal{G}_6$, for any $G_i \in \mathcal{G}_6$ with the same degree sequence as G, let

α_i be its number of self-loops and

$$\beta_i = \frac{1}{\prod_{k \leq l} A^i_{kl}!}.$$

Let $\Pi_i = \frac{1}{2}^{\alpha_i}\beta_i$ and let $\pi_i = {}^{\Pi_i}/\sum_j \Pi_j$. Then the detailed balance equations hold for π.

Proof. If there exists a switch between G_i and G_j then $|\alpha_i - \alpha_j|$ equals either 0,1, or 2. Suppose $\alpha_i - \alpha_j$ equals 0 or 1. Without loss of generality we assume that the vertices are labelled such that $E_j = E_i \backslash \{\{x,y\},\{u,v\}\} \cup \{\{x,v\},\{u,y\}\}$. When $\alpha_i = \alpha_j$ the derivation mimics the derivation in the proof of Lemma 2.2.19. Now let $\alpha_i = \alpha_j + 1$. By Lemma 2.2.20 we find

$$p_{ij} = \frac{A^i_{\{xy\}}A^i_{\{uv\}}}{M}$$

and

$$p_{ji} = \frac{A^j_{\{xv\}}A^i_{\{uy\}}}{2M}.$$

Let

$$L_i = \frac{1}{2M}\frac{1}{\sum_r \frac{1}{2}^{\alpha_r}\beta_r}\frac{1}{\prod_{k \leq l, \{k,l\} \notin E_i \Delta E_j} A^i_{\{kl\}}!},$$

and notice that $L_i = L_j$. Now, using the transition probabilities found in Lemma 2.2.20 we derive:

$$\begin{aligned}
\pi_i p_{ij} &= \frac{\frac{1}{2}^{\alpha_i}\beta_i}{\sum_r \frac{1}{2}^{\alpha_r}\beta_r}\frac{A^i_{\{xy\}}A^i_{\{uv\}}}{M} \\
&= L_i\,\frac{1}{2^{\alpha_i}}\frac{2A^i_{\{xy\}}A^i_{\{uv\}}}{A^i_{\{xy\}}!A^i_{\{uv\}}!A^i_{\{xv\}}!A^i_{\{uy\}}!} \\
&= L_i\,\frac{1}{2^{\alpha_j+1}}\frac{2}{(A^i_{\{xy\}}-1)!(A^i_{\{uv\}}-1)!A^i_{\{xv\}}!A^i_{\{uy\}}!} \\
&= L_i\frac{1}{2^{\alpha_j}}\frac{1}{A^j_{\{xy\}}!A^j_{\{uv\}}!(A^j_{\{xv\}}-1)!(A^j_{\{uy\}}-1)!} \\
&= L_j\frac{1}{2^{\alpha_j}}\frac{A^j_{\{xv\}}A^j_{\{uy\}}}{A^j_{\{xy\}}!A^j_{\{uv\}}!A^j_{\{xv\}}!A^j_{\{uy\}}!} \\
&= \frac{\frac{1}{2}^{\alpha_j}\beta_j}{\sum_r \frac{1}{2}^{\alpha_r}\beta_r}\frac{A^j_{\{xv\}}A^j_{\{uy\}}}{2M} \\
&= \pi_j p_{ji}
\end{aligned}$$

Finally, assume that $\alpha_i = \alpha_j + 2$. Without loss of generality we assume that $E_j = E_i \backslash \{\{x,x\},\{y,y\}\} \cup \{\{x,y\},\{x,y\}\}$. Let

$$K_i = \frac{1}{2M} \frac{1}{\sum_r \frac{1}{2}^{\alpha_r} \beta_r} \frac{1}{\prod_{k \leq l, \{k,l\} \notin E_i \Delta E_j} A^i_{\{kl\}}!},$$

and notice that $K_i = K_j$. Then,

$$\begin{aligned}
\pi_i p_{ij} &= \frac{\frac{1}{2}^{\alpha_i} \beta_i}{\sum_r \frac{1}{2}^{\alpha_r} \beta_r} \frac{A^i_{\{xx\}} A^i_{\{yy\}}}{M} \\
&= K_i \frac{1}{2^{\alpha_i}} \frac{2 A^i_{\{xx\}} A^i_{\{yy\}}}{A^i_{\{xx\}}! A^i_{\{yy\}}! A^i_{\{xy\}}!} \\
&= K_i \frac{1}{2^{\alpha_j+2}} \frac{2}{(A^i_{\{xx\}} - 1)!(A^i_{\{yy\}} - 1)! A^i_{\{xy\}}!} \\
&= K_i \frac{1}{2^{\alpha_j+1}} \frac{1}{A^j_{\{xx\}}! A^j_{\{yy\}}! (A^j_{\{xy\}} - 2)!} \\
&= K_j \frac{1}{2^{\alpha_j+1}} \frac{A^j_{\{xy\}} (A^j_{\{xy\}} - 1)}{A^j_{\{xx\}}! A^j_{\{yy\}}! A^j_{\{xy\}}!} \\
&= \frac{\frac{1}{2}^{\alpha_j} \beta_j}{\sum_r \frac{1}{2}^{\alpha_r} \beta_r} \frac{A^j_{\{xy\}} (A^j_{\{xy\}} - 1)}{4M} \\
&= \pi_j p_{ji}.
\end{aligned}$$

□

Again we may introduce acceptance probabilities a_{ij} to obtain the **adjusted switching method** with respect to $\mathcal{G}_6$. In this case we define the acceptance probabilities as follows. Let G_i and G_j differ by a directed switch, and let $E_j = E_i \backslash \{\{x,y\},\{u,v\}\} \cup \{\{x,v\},\{u,y\}\}$, then

$$a_{ij} = \begin{cases} \frac{1}{A^i_{xy} A^i_{uv}} & \text{if } \alpha_i \leq \alpha_j \\ \frac{1}{2 A^i_{xy} A^i_{uv}} & \text{if } \alpha_i = \alpha_j + 1 \\ \frac{1}{4 A^i_{xy} A^i_{uv}} & \text{if } \alpha_i = \alpha_j + 2 \end{cases}$$

In all other situations $a_{ij} = 1$. These acceptance probabilities ensure that the uniform distribution is the stationary distribution of this Markov chain.

Finally, we inspect the detailed balance equations for the remaining classes of directed networks, $\mathcal{G}_7$ and $\mathcal{G}_8$.

Lemma 2.2.22. Let G_i and G_j be two directed networks in $\mathcal{G}_7$ with equal degree sequences. The transition probability p_{ij} from G_i to G_j is given by

$$p_{ij} = \begin{cases} \frac{A^i_{xy}A^i_{uv}}{M} & \text{if } E_j = E_i \backslash \{(x,y),(u,v)\} \cup \{(x,v),(u,y)\} \\ 1 - \sum_{k,k\neq i} p_{ik} & \text{if } i = j \\ 0 & \text{otherwise.} \end{cases}$$

Proof. If G_i and G_j differ by a directed switch, we may assume that $E_j = E_i\backslash\{(x,y),(u,v)\}\cup \{(x,v),(u,y)\}$. The probability of selecting an edge pair (x,y), (u,v) in G_i equals $A^i_{xy}A^i_{uv}/M$. This is the transition probability from G_i to G_j. □

Lemma 2.2.23. Let $G \in \mathcal{G}_7$ be a directed network without self-loops. For any $G_i \in \mathcal{G}_7$ with the same degree sequence as G, and with adjacency matrix A^i, let

$$\beta_i = \frac{1}{\prod_{kl} A^i_{kl}!}.$$

The switching model with respect to $\mathcal{G}_7$ satisfies the detailed balance equations for $\pi_i = \beta_i / \sum_r \beta_r$.

Proof. The proof is analogous to the proof of Lemma 2.2.19. □

For directed networks, there are no special probabilities for self-loops unlike in undirected networks. An edge pair corresponds to a unique switch, regardless of whether the edges involved are self-loops or not. Thus, the above lemma holds for $\mathcal{G}_8$ as well.

Again we can introduce acceptance probabilities to obtain **the adjusted switching method** with respect to $\mathcal{G}_7$ and $\mathcal{G}_8$. We define the acceptance probabilities as follows. Let G_i and G_j differ by a directed switch, and let $E_j = E_i\backslash\{(x,y),(u,v)\}\cup\{(x,v),(u,y)\}$, then $a_{ij} = 1/A^i_{xy}A^i_{uv}$, otherwise $a_{ij} = 1$. These acceptance probabilities ensure that the stationary distribution is the uniform distribution.

Finally, the triangle enriched switching models do not change the discussion for the detailed balance equation. If two networks G_i and G_j differ by a triangle reorientation the transition probabilities p_{ij} and p_{ji} are zero in the (adjusted) switching method and are both equal to $^3/_M$ for the triangle enriched switching model.

To conclude, the switching model for a network G with respect to class $\mathcal{G}_1$ converges to the uniform distribution regardless of G. The switching model with respect to $\mathcal{G}_4$ and the adjusted switching model with respect to $\mathcal{G}_8$ converge to the uniform distribution for networks that contain at least one vertex with in-degree two or more, or out-degree two or more. The adjusted switching models with respect to classes $\mathcal{G}_5$ and $\mathcal{G}_6$ both converge to the uniform distribution for any G. The triangle enriched switching model with respect to $\mathcal{G}_3$ and the adjusted triangle enriched switching model with respect to $\mathcal{G}_7$ converge to the uniform distribution for all networks G that contain at least one vertex of degree two. Finally, we expect the adjusted switching model with respect to $\mathcal{G}_2$ to converge to the uniform distribution for most networks. However, we have shown that it does not do so for *all* networks; there are classes of networks for which its Markov chain is reducible.

The random network models discussed in this section have been implemented and are publicly available, see Appendix B.1.6 for more details on this software.

2.3. MFinder implementation flaw

We now describe an issue in the implementation of the switching method with respect to undirected networks in MFinder[2]. MFinder is a software package that detects motifs [82] by comparing a network to a collection of random networks, by default generated using the switching model. We first describe the implementation of the switching method with respect to $\mathcal{G}_2$ as found in MFinder, then point out the issue, and finally show how this issue leads to biased sampling.

The switching method with respect to *undirected networks* as implemented in MFinder, takes as its input an edge list and the number N of switches to attempt. It then iterates

[2] As downloaded from http://wws.weizmann.ac.il/mcb/UriAlon/download/network-motif-software on 8 March 2015

through the following steps until the number of attempts equals N.

(a) randomly select two edges from the edge list, call these $\{x, y\}$ and $\{u, v\}$,

(b) if neither of the edges $\{x, v\}$ and $\{u, y\}$ is already present in the graph and if neither is a self-loop, then replace the original edges with $\{x, v\}$ and $\{u, y\}$, otherwise do nothing,

(c) add one to the number of attempts.

The reason that this implementation produces biased samples is that edges are stored with fixed vertex order. If the two edges $\{x, y\} = \{y, x\}$ and $\{u, v\} = \{v, u\}$ are retrieved as (x, y) and (u, v) then switch 1 to $\{x, v\}$ and $\{u, y\}$ is attempted, but never switch 2 to $\{x, u\}$ and $\{y, v\}$ (Figure 2.7). In other words, restricting to switch 1 and storing the edges with ordered vertices, corresponds to treating undirected networks as directed networks, and only switching targets.

Specifically, when storing undirected edges as ordered pairs, the following steps need to be implemented:

(a) randomly select two edges from the edge list, call these $\{x, y\}$ and $\{u, v\}$,

(b0) select either switch 1 or switch 2 (Figure 2.7) with probability 0.5

(b1) if switch 1: If neither of the edges $\{x, v\}$ and $\{u, y\}$ is already present in the graph and if neither is a self-loop, then replace the original edges with $\{x, v\}$ and $\{u, y\}$, otherwise do nothing.

(b2) or if switch 2: If neither of the edges $\{x, u\}$ and $\{y, v\}$ is already present in the graph and if neither is a self-loop, then replace the original edges with $\{x, u\}$ and $\{y, v\}$, otherwise do nothing.

(c) add one to the number of attempts.

It is not immediately clear that neglecting to attempt switch 2 leads to biased sampling. We now discuss a specific example theoretically, and prove that indeed sampling will be biased. We then confirm experimentally that MFinder produces biased samples as predicted. Depending on the vertex order in the edge list, the sample is moderately to extremely biased.

Example 2.3.1. Let G_1 be a network with six vertices $v_1, \ldots, v_6$ and six edges $\{v_1, v_3\}$, $\{v_2, v_4\}$, $\{v_3, v_4\}$, $\{v_3, v_5\}$, $\{v_4, v_6\}$ and $\{v_5, v_6\}$. There are 17 simple undirected networks with the same degree sequence as G_1 as displayed in Figure 2.13.

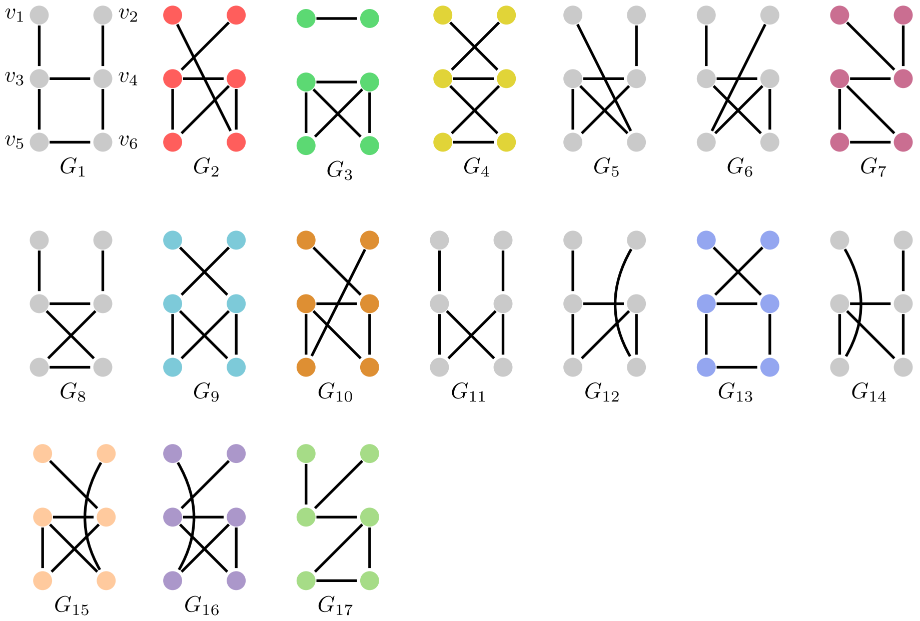


Figure 2.13.: The 17 simple undirected graphs with degree sequence $k = (1, 1, 3, 3, 2, 2)$.

We first look at a worst-case scenario, where the only switch that is being attempted, switch 1, is not allowed for any of the edge pairs (Figure 2.14(b)).

$$(v_1, v_3), (v_4, v_2), (v_4, v_3), (v_4, v_6), (v_5, v_3), (v_5, v_6)$$

One can check that for any edge pair, applying switch 1 either does not alter the network (when the edges have the same source or the same target) or introduces a self-loop or a multiple edge. Thus, any 'random' sample generated by this method will consist of only network G_1 itself.

The second case we discuss samples from a slightly larger subset of the 17 networks. Let

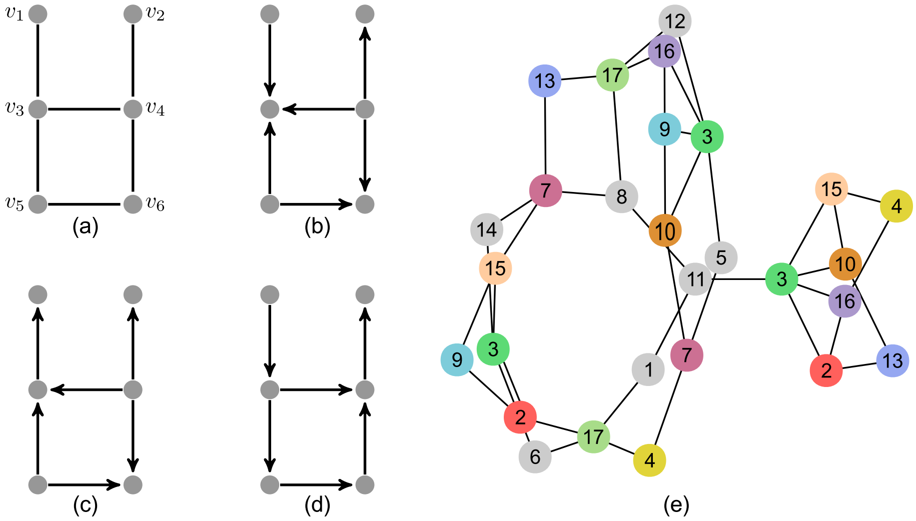


Figure 2.14.: (a) The undirected network G_1. (b)-(d) Three directed networks corresponding to storing G_1 with different vertex orderings: (b) $\{(v_1, v_3), (v_4, v_2), (v_4, v_3), (v_4, v_6), (v_5, v_3), (v_5, v_6)\}$, and (c) $\{(v_3, v_1), (v_4, v_2), (v_4, v_3), (v_4, v_6), (v_5, v_3), (v_5, v_6)\}$ and (d) $\{(v_1, v_3), (v_4, v_2), (v_3, v_4), (v_6, v_4), (v_3, v_5), (v_5, v_6)\}$. (e) The structure of the state graph corresponding to the randomisation method for undirected networks found in MFinder with edgelist as in (d). In this state graph, many of the 17 undirected networks correspond to several states. This is indicated by the colors and labels. For visual clarity we have not depicted self-loops in the state graph.

the edge list be stored as below (Figure 2.14(c)).

$$(v_3, v_1), (v_4, v_2), (v_4, v_3), (v_4, v_6), (v_5, v_3), (v_5, v_6)$$

Switching using just switch 1 is again very restricted. By checking all edge-pairs in G_1 and the networks obtained by switching edges in G_1, we find that only five of the 17 networks can be reached. This implies that the MFinder algorithm only samples the networks G_1, G_7, G_{10}, G_{13} and G_{14}.

Our third and final case shows that even when the MFinder algorithm is able to sample from the full set of seventeen networks, the sample may still be biased. This is the case when the edge list is stored with the following vertex orders (Figure 2.14(d)).

$$(v_1, v_3), (v_4, v_2), (v_3, v_4), (v_6, v_4), (v_3, v_5), (v_5, v_6)$$

R code was written to recursively find the allowed switches for each network and hence the state graph as shown in Figure 2.14(e).

The problem in this case is that not all networks are uniquely represented by an ordered edge list. For instance network G_2 in Figure 2.13 can be represented by the following two edge lists with distinct vertex orderings.

$$(v_1, v_6), (v_4, v_5), (v_5, v_3), (v_3, v_4), (v_3, v_2), (v_6, v_4) \tag{2.1}$$

$$(v_1, v_6), (v_4, v_3), (v_5, v_4), (v_3, v_5), (v_3, v_2), (v_6, v_4) \tag{2.2}$$

Similarly network G_3 in Figure 2.13 can be represented by the following three edge lists.

$$(v_1, v_2), (v_6, v_4), (v_3, v_6), (v_3, v_4), (v_4, v_5), (v_5, v_3) \tag{2.3}$$

$$(v_1, v_2), (v_6, v_4), (v_3, v_6), (v_3, v_5), (v_4, v_3), (v_5, v_4) \tag{2.4}$$

$$(v_1, v_2), (v_6, v_3), (v_3, v_4), (v_3, v_5), (v_4, v_6), (v_5, v_4) \tag{2.5}$$

The Markov chain corresponding to MFinder's undirected network randomisation algorithm corresponds to a random walk on the state graph in Figure 2.14(e). The Markov chain is irreducible and aperiodic; its state graph is connected and each state has a non-zero probability of being repeated. The probability of moving from one state to its neighbour is the probability of selecting the corresponding 'directed' switch and equals $1/M$. Here $M = m(m-1)/2$ and $m = 6$, the number of edges of G. The Markov chain converges to a stationary distribution π, and π is the uniform distribution on the states. However, some states in this state space correspond to the same undirected network. We thus find that MFinder samples undirected networks from the probability distribution:

$$(1/28, 1/14, 3/28, 1/14, 1/28, 1/28, 1/14, 1/28, 1/14, 1/14, 1/28, 1/28, 1/14, 1/28, 1/14, 1/14, 1/14).$$

To show that MFinder indeed produces samples according to the distributions we described in this example, we wrote several scripts. The resulting samples are displayed in Figure 2.15.

I have contacted the authors of [82] and this oversight has been fixed. A new version of MFinder is now available (see Appendix B.2.5).

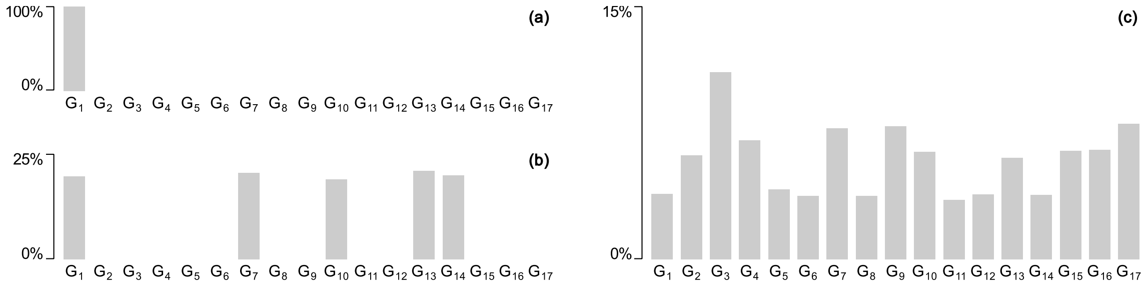


Figure 2.15.: Histograms of samples of N random undirected networks produced by MFinder. (a) Input network G_1 labelled as in Figure 2.14(b), $N = 1000$. (b) Input network G_1 labelled as in Figure 2.14(c), $N = 1000$. (c) Input network G_1 labelled as in Figure 2.14(d), $N = 10000$.

2.4. The ordered switching model

In this section we discuss the switching model for yet another class of networks, the class of directed acyclic networks. This is an important class of networks that occurs in biology, computer science, engineering, pure mathematics and statistics [67]. It is a subclass of networks in $\mathcal{G}_3$ with the property that they do not contain directed cycles (Definition 1.1.6). Examples of directed acyclic networks include citation networks, patent networks, causal structures and family trees.

After our careful definition in Section 2.1, it is straightforward to define the ordered switching model: the switching model with respect to directed acyclic networks. Recall from Proposition 1.1.8 that such networks admit a topological ordering of their vertices.

Definition 2.4.1. Let G be a directed acyclic network. The **ordered switching model** for G is defined by a Markov chain starting at G. The states of this Markov chain are all the directed acyclic networks that have the same degree sequences **and** the same topological ordering as G. If there exists a directed switch between G_i and G_j then the transition probability p_{ij} is the probability of selecting the corresponding unique edge pair. That is

$$p_{ij} = \begin{cases} \frac{1}{M} & \text{if there exists a directed switch between } G_i \text{ and } G_j \\ 1 - \frac{k(G_i)}{M} & \text{if } j = i \\ 0 & \text{otherwise} \end{cases},$$

with $M = {}^{m(m-1)}/_{2}$ as usual.

We will use Theorem 1.3.6 to show that the ordered switching model converges to the uniform distribution for most directed acyclic networks.

2.4.1. Irreducibility

We now prove that the Markov chain corresponding to the ordered switching model is irreducible. To do so, let $G = (V, E)$ and $G' = (V', E')$ be two directed acyclic networks with $V = V'$, that have equal degree sequences and equal topological ordering. For such graphs we show that there exists a sequence of directed acyclic networks $G = G_0, G_1, \ldots, G_k = G'$ such that each G_i has the same degree sequences and allows the same topological ordering as G and G', and furthermore each consecutive pair of networks G_i and G_{i+1} differs by a directed switch. The approach we take to prove this is similar to the approach in [16] for simple directed networks: we use induction on the size of the symmetric edge set difference $E \Delta E'$ of G and G' (Definition 1.2.1).

The graph $G \Delta G'$ has some nice properties due to the fact that G and G' have equal degree sequences. In particular we have seen that $E \Delta E'$ partitions into minimal closed alternating direction-alternating trails (see Lemmas 1.2.2 and 1.2.12) and that the vertices visited by such a trail are entered at most once through an incoming edge and at most once through an outgoing edge (Lemma 1.2.11). We now show that a minimal closed alternating direction-alternating trail contains five consecutive distinct vertices if it is of length greater than 4. Notice that the graph $G \Delta G'$ allows the same topological ordering as G and G'.

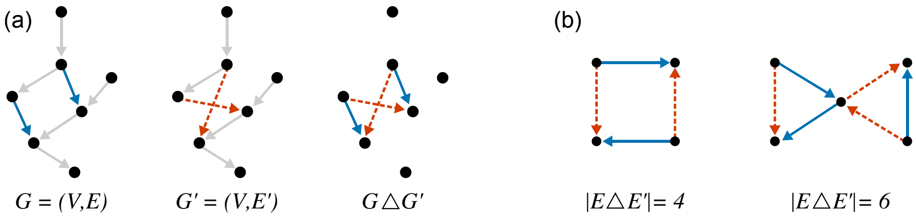


Figure 2.16.: (a) An example of two directed acyclic realizations, G and G' of $k^{in} = (1,2,2,1,0,1,0), k^{out} = (0,1,1,1,1,2,1)$, drawn such that the topological ordering corresponds to the order from bottom to top. These two networks differ in exactly one directed switch, which is equivalent to $|E\Delta E'| = 4$. Grey edges represent edges present in both E and E', red edges represent edges in $E \backslash E'$ and blue edges represent edges in $E' \backslash E$. (b) Examples of symmetric edge set differences with four and six edges respectively.

Lemma 2.4.2. Let $G = (V, E)$ and $G' = (V, E')$ be distinct directed acyclic networks with equal degree sequences, and let $V = (v_1, \ldots, v_n)$ be a topological ordering for both. Let C be a closed alternating direction-alternating trail in $G\Delta G'$. If C is minimal and $|C| > 4$ then C contains five consecutive distinct vertices.

Proof. We will prove this by contradiction. So let us assume that C is a minimal closed alternating direction-alternating trail with $|C| > 4$ and C does not contain five distinct consecutive vertices.

Let $v_{i_{max}} \in C$ be the vertex with the highest index in C. There are distinct vertices v_{i_0}, v_{i_1} such that $(v_{i_{max}}, v_{i_0}) \in E \backslash E'$ and $(v_{i_{max}}, v_{i_1}) \in E' \backslash E$. We may write $C = v_{i_0}, v_{i_{max}}, v_{i_1}, \ldots, v_{i_0}$.

First assume that the next vertex, v_{i_2}, in C is not equal to v_{i_0}. This implies that $v_{i_0}, v_{i_1}, v_{i_{max}}$ and v_{i_2} are all distinct, since $(v_{i_2}, v_{i_1}) \in E \backslash E'$ and $(v_{i_{max}}, v_{i_1}) \in E' \backslash E$. The next vertex v_{i_3}, with $(v_{i_2}, v_{i_3}) \in E' \backslash E$ can not be equal to v_{i_2} or to v_{i_1} (since $(v_{i_2}, v_{i_1}) \in E$). It is also different from $v_{i_{max}}$, since $v_{i_{max}}$ can not have incoming edges in $G\Delta G'$. Thus, if $v_{i_2} \neq v_{i_0}$ then $v_{i_3} = v_{i_0}$ since otherwise there would be five distinct consecutive vertices in C (Figure 2.17(a)). But if $v_{i_3} = v_{i_0}$ then $v_{i_0}, v_{i_{max}}, v_{i_1}, v_{i_2}, v_{i_0}$ is a proper closed alternating direction-alternating subtrail of C (Figure 2.17(b), contradicting our assumption that C is minimal.

So C has to start with $v_{i_0}, v_{i_{max}}, v_{i_1}, v_{i_0}$. The next vertex, call it v_{i_2}, with $(v_{i_0}, v_{i_2}) \in E' \backslash E$ has to be different from all previous vertices: obviously $v_{i_2} \neq v_{i_0}$, it can not be equal to v_{i_1} since $(v_{i_0}, v_{i_1}) \in E \backslash E'$ and it can not be equal to $V_{i_{max}}$ since $i_{max} > i_0$ (see also Figure 2.17(c)). Note that $i_0 > i_2$ since V is the topological ordering of G and G'.

Let v_{i_3} be the next vertex on the trail with $(v_{i_3}, v_{i_2}) \in E \backslash E'$. Hence $v_{i_3} \neq v_{i_2}$, furthermore $v_{i_3} \neq v_{i_0}$ since $(v_{i_0}, v_{i_2}) \in E' \backslash E$ and $v_{i_3} \neq v_{i_{max}}$ since $v_{i_{max}}$ can only be entered through an outgoing edge once. Thus v_{i_3} has to be equal to v_{i_1}, since otherwise $v_{i_{max}}, v_{i_1}, v_{i_0}, v_{i_2}, v_{i_3}$ are five distinct consecutive vertices.

Repeating this argument, we find that our alternating cycle is of the form

$$v_{i_0}, v_{i_{max}}, v_{i_1}, v_{i_0}, v_{i_2}, v_{i_1}, v_{i_3}, v_{i_2}, v_{i_4}, v_{i_3}, \dots, v_{i_j}, v_{i_0}.$$

This implies that $i_0 > i_1 > i_3 > i_5 \dots$ and $i_0 > i_2 > i_4 > \dots$. In particular $i_0 > i_j$, but $(v_{i_j}, v_{i_0}) \in E' \backslash E$, giving a contradiction. □

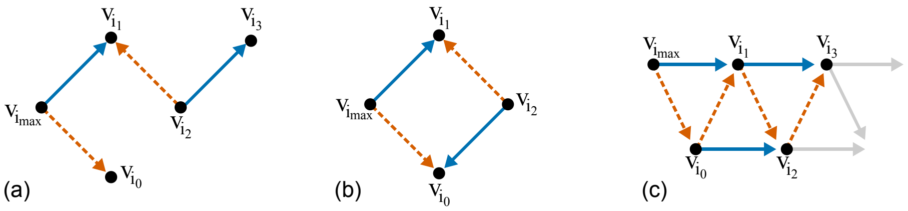


Figure 2.17.: The different cases discussed in Lemma 2.4.2. (a) The case where $v_{i_2} \neq v_{i_0}$ and $v_{i_3} \neq v_{i_0}$. (b) The case where $v_{i_2} \neq v_{i_0}$ and $v_{i_3} = v_{i_0}$. (c) The case where $v_{i_2} \neq v_{i_0}$, and we have a cycle of the form $v_{i_0}, v_{i_{max}}, v_{i_1}, v_{i_0}, v_{i_2}, v_{i_1}, v_{i_3}, v_{i_2}, v_{i_4}, v_{i_3}, \dots, v_{i_j}, v_{i_0}$.

With Lemma 2.4.2 in hand we are now ready to prove irreducibility of the Markov chain of the ordered switching model.

Lemma 2.4.3. Let $G = (V, E)$ be a directed acyclic network with a topological ordering. Let $\mathcal{G}$ be the set of all directed acyclic networks with vertices V, the same degree sequences as G and that allow the same topological ordering. Then for any network $G' \in \mathcal{G}$, there exists a sequence $G = G_0, G_1, \dots, G_k = G'$ of networks such that each G_i is an element of $\mathcal{G}$ with edge set differences, $E_i \Delta E_{i+1}$, of each consecutive pair of networks, of size 4.

Proof. We prove the lemma by strong induction on the size of the symmetric edge set difference $|E\Delta E'| = 2\kappa$. The base case, $\kappa = 2$, is trivial, since we may use the sequence $G_0 = G, G_1 = G'$.

Assume that we can find such sequences for all pairs of networks $G, G' \in \mathcal{G}$ with $|E\Delta E'| = 2\kappa$ and $\kappa \leq l$. We show this implies the statement is true for $G, G' \in \mathcal{G}$ with $|E\Delta E'| = 2l + 2$.

From Lemma 1.2.12 we know that the symmetric edge set difference $E\Delta E'$ of G and G' partitions into minimal closed alternating direction-alternating trails. We can distinguish between the following two cases.

1. There are at least two minimal closed alternating direction-alternating trails in $G\Delta G'$.

   Let C be one such minimal closed alternating direction-alternating trail. Then $|C| \leq 2l$. Define $E^* = (E \backslash (E \cap C)) \cup (E' \cap C)$. The corresponding network $G^* = (V, E^*)$ has the same in-degree and out-degree sequence as G and respects the topological ordering V. Now $|E\Delta E^*| = |C| \leq 2l$ and $|E^*\Delta E'| = |E\Delta E'| - |C| \leq 2l$. Hence, by induction, we can find sequences $G = G_0, \ldots, G_j = G^*$ and $G^* = G'_0, \ldots, G'_k = G'$ that satisfy the required properties. Concatenating the two sequences results in a sequence from G to G' with each $G_i \in \mathcal{G}$ and each two consecutive networks differing by a switch.

2. The edge set difference $E\Delta E'$ contains exactly one minimal closed alternating direction-alternating trail.

   We call this trail C, and hence $|C| = 2l + 2$. By Lemma 2.4.2, C contains five consecutive distinct vertices, label them $v_{i_1}, v_{i_2}, v_{i_3}, v_{i_4}, v_{i_5}$. Without loss of generality assume $(v_{i_1}, v_{i_2}), (v_{i_3}, v_{i_4}) \in E$ and $(v_{i_3}, v_{i_2}), (v_{i_5}, v_{i_4}) \in E'$. We may also assume $i_1 > i_4$, since if $i_1 < i_4$, then $i_5 > i_2$ which is analogous.

   We may now distinguish the following three cases:

a. $(v_{i_1}, v_{i_4}) \notin E$

Let $E^* = (E \backslash \{(v_{i_1}, v_{i_2}), (v_{i_3}, v_{i_4})\}) \cup \{(v_{i_1}, v_{i_4}), (v_{i_3}, v_{i_2})\}$. Now $|E \Delta E^*| = 4$ and $|E^* \Delta E'| \leq 2l - 2$. By induction we can find a sequence of networks from $G^* = (V, E^*)$ to G', concatenating this with G results in a suitable sequence of networks from G to G'.

b. $(v_{i_1}, v_{i_4}) \in E \backslash E'$

This can not be the case since this would imply that v_{i_4} has two incoming edges in $E \backslash E'$, contradicting the fact that C is minimal (Lemma 1.2.11).

c. $(v_{i_1}, v_{i_4}) \in E \cap E'$

Let $E^* = (E' \backslash \{(v_{i_1}, v_{i_4}), (v_{i_3}, v_{i_2})\}) \cup \{(v_{i_1}, v_{i_2}), (v_{i_3}, v_{i_4})\}$. Now $|E^* \Delta E'| = 4$ and $|E \Delta E^*| = 2l$. By induction we can find a sequence of networks from G to $G^* = (V, E^*)$. Concatenating this sequence with G' results in a suitable sequence of networks from G to G'.

This finishes the proof. □

2.4.2. Aperiodicity

Recall from Lemma 2.1.4 that the switching model for simple directed networks is aperiodic for networks that have at least one vertex of total degree 2 or more. For such networks, there is a non-zero probability of repeating any state, which trivially implies aperiodicity of the corresponding Markov chain. The ordered switching model allows only a subset of the switches that are allowed by the switching model with respect to simple directed networks. Hence, it is clearly also aperiodic for networks G that contain at least one vertex with total degree 2 or more.

2.4.3. Detailed balance equations

It is straightforward to see that the simplified detailed balance equations hold: for any two neighbouring states G_i and G_j, the transition probabilities p_{ij} and p_{ji} equal $^1/_M$.

To summarize, we have defined a switching model for the class of directed acyclic networks. We have shown that the corresponding Markov chain is irreducible, aperiodic and satisfies the simplified detailed balance equations for almost all directed acyclic networks. Hence we have proven the following theorem.

Theorem 2.4.4. *Let G be a directed acyclic network that contains a vertex v of total degree at least two. The Markov chain corresponding to the switching model for G with respect to directed acyclic networks, converges to the uniform distribution.*

2.5. Conclusion

In this chapter we have discussed a family of switching models in detail. We introduced a precise definition of the switching model with respect to nine different classes of networks. For each network class, we found conditions under which the corresponding switching model converges to a stationary distribution and derived this distribution. When necessary, we introduced acceptance probabilities, such that the resulting adjusted switching model converges to the uniform distribution.

We showed how our precise definition of switching models resolves ambiguities found in previous definitions. We found and resolved an error in a well-known software package. Finally we produced R code that implements all versions of the switching model that are discussed in this chapter.

One of the strengths of the switching model is its simplicity: it is a simple procedure of edge swaps that randomizes networks while fixing their degree sequence(s). In this chapter, we showed that when treated carefully, the switching model can be used to draw uniform samples from a variety of network classes. This makes the switching model an attractive candidate for a null-model.

However, so far we have ignored one crucial question about the switching models: how many switches do we need to attempt, in order to obtain a truly random network? Or, in other words, how many steps does the corresponding Markov chain need to take to reach its stationary distribution. This question is generally hard to answer, and is the main focus of Chapter 3.

Perhaps the choice to include classes $\mathcal{G}_5 - \mathcal{G}_8$ in our analysis seems somewhat unnatural: most real-world networks do not contain multiple edges, or are not modelled as multigraphs. The reason that we included these classes is two-fold.

Firstly, the switching model is simplest when randomizing multigraphs ($\mathcal{G}_6, \mathcal{G}_8$), since all pairs of edges are allowed to be switched. In a sense, this is the most natural setting for the switching model.

Secondly, the configuration model [84, 95] produces almost uniform [94, Section 13.2] samples from $\mathcal{G}_6$ and $\mathcal{G}_8$. It was recently suggested that combining the configuration model and a Markov chain model could be a faster way to obtain a truly random network from, for instance, classes $\mathcal{G}_1$ and $\mathcal{G}_3$ [133]. We also discuss this Expand and Contract method in more detail in Chapter 3.

3. Run-time of random network models

In this chapter we compare the run-times of several random network models. We look at several Markov chain models as well as the configuration model and the recently introduced Expand and Contract model. The mixing time of a Markov chain is needed to estimate its run-time. It is generally very complicated to estimate the mixing time. In 1999, Kannan et al. conjectured that the switching model mixes rapidly and proved that it does for regular bipartite networks [66]. Their claim has since also been proven for regular [46] and semi-regular [41] networks.[1] However, there is no proof for the general case.

All random network models discussed in this chapter randomise directed networks, i.e. networks in class $\mathcal{G}_4$. In the remainder of this chapter we will refer to the switching model with respect to $\mathcal{G}_4$ as *the switching model*.

In Section 3.1 we compare three Markov chain models to the switching model. The Markov chain models that we discuss, only differ from the switching model in their non-zero transition probabilities. That is, they allow the same set of *moves* as the switching model. We show that the switching model is currently the best performing model in terms of run-time and in fact is optimal given the structure of the state graph. In Section 3.2 we compare the switching model to the configuration model and discuss advantages and drawbacks of both models. We also discuss the Expand and Contract model [133]

[1] Regular bipartite networks are bipartite networks for which all nodes in the primary node set have equal degree and all nodes in the secondary node set have equal degree. Regular directed networks are networks in which all nodes have the same in-degree as well as the same out-degree (which are necessarily equal). Semi-regular networks are networks where all nodes have the same in-degree but their out-degree may differ (or vice versa).

and show that in practice, it does not improve on the run-time of the switching model. We finish this chapter with our conlcusions in Section 3.3.

My contributions to the area in this chapter are the following.

- Comparison of several random network models in terms of their state graph and run-time. These results are not yet published.
- Derivation of a simple formula for the change in *mobility* of a state after applying a switch. This result has not yet been published.

3.1. Alternative Markov chains

In Chapter 2 we discussed the switching model in detail. The definition given there (Definition 2.2.1) was phrased in such a way that the implementation of the model is implied. That is, by specifying the transition probabilities in terms of selecting edge pairs, we left no ambiguity for the implementation of the algorithm.

The Markov chains that we discuss in this section are defined in a more theoretical way. That is, it is not clear from their definition alone, how they should be implemented. After studying their implementations (if available) or implementing the algorithms ourselves, we found that even though theoretically these Markov chains converge faster than the switching model, in practice they run slower. We show that the switching model already runs at optimal speed.

Some of the algorithms we discuss here were originally defined for **simple** directed networks. Here we discuss versions of these algorithms where self-loops are allowed to be formed, since this clarifies our argument. However, our argument holds equally well for the run-times of these models with respect to simple directed networks.

3.1.1. Definitions

The three models that we discuss here are (1) a Markov chain model introduced by Rao et al. [103] which we will refer to as *Rao's model*, (2) a 'canonical' model introduced by Coolen and Roberts [31, 108, 107] which we will refer to as *the canonical model*, and (3) the 'switch and add' model that was introduced by Artzy-Randrup and Stone [6] which we will refer to as *the switch and add model*.

The random network models that we discuss here are Markov chains with the same states and the *same moves* as the switching model.

Definition 3.1.1. Let X_i be a state of a finite Markov chain with transition probabilities $\{p_{ij}\}$. The set of states that can be reached from X_i, that is $\{X_j \mid i \neq j,\ p_{ij} \neq 0\}$, is referred to as the set of **moves** from X_i. The number of moves from a state X_i is called the **mobility** of the state and denoted by m_i [108].

Since these models have the same sets of moves as the switching model, the structure of their state graphs is almost exactly equal to that of the switching model. If two distinct states are connected in the state graph of the switching model then they are also connected in the Markov chains corresponding to these methods. However, some of the states that have a non-zero probability of being repeated in the switching model have a zero probability of being repeated in these models. Hence the state graphs may differ in the number of self-loops. The main difference between these models and the switching model is in the value of their non-zero transition probabilities.

As some of these state graphs do not include any repeated states we will need the following lemma to ensure aperiodicity.

Lemma 3.1.2. Let $G = (V, E)$ be a directed network that contains three edges (u_1, v_1), (u_2, v_2), (u_2, v_3) with $u_1 \neq u_2$, $v_1 \notin \{v_2, v_3\}$ and $\{(u_1, v_2), (u_1, v_3), (u_2, v_1)\} \cap E = \emptyset$. The state graph Ψ of the switching model for G is aperiodic even after removing all self-loops.

Proof. The state graph Ψ is connected and finite, hence it is enough to show that it

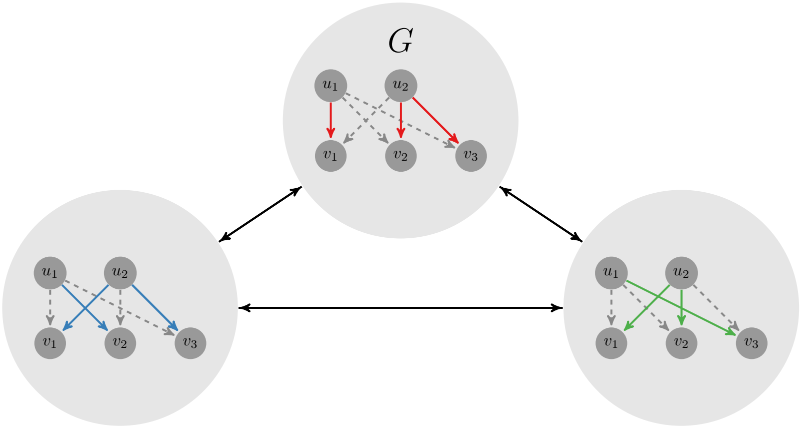


Figure 3.1.: For each state in this figure, there is a closed path of length three starting and ending at the state, as well as paths of length two. Hence each state is aperiodic (Lemma 3.1.2). Coloured edges indicate edges that are present in each network, grey dashed edges are *not* present in the networks.

contains one aperiodic state. G corresponds to an aperiodic state since there is a path of length three starting and ending at G given by the following sequence of switches, corresponding to moving clockwise from G in Figure 3.1.

$$\begin{aligned}
\{(u_1, v_1), (u_2, v_3)\} &\rightarrow \{(u_1, v_3), (u_2, v_1)\}, \\
\{(u_1, v_3), (u_2, v_2)\} &\rightarrow \{(u_1, v_2), (u_2, v_3)\}, \\
\{(u_1, v_2), (u_2, v_1)\} &\rightarrow \{(u_1, v_1), (u_2, v_2)\}.
\end{aligned}$$

□

In the rest of this section *we assume that all networks satisfy the conditions of Lemma 3.1.2*. The definitions that we give for the random network models are rephrased in a way that simplifies their comparison. We only focus on the run-time of these algorithms. However, this was not necessarily the original reason to create these algorithms. For instance the canonical model was introduced to allow for several different sampling distributions. For more information and context for each of these algorithms we refer the reader to the original papers [6, 108, 103].

The first Markov chain that we discuss, was introduced by Rao et al. in 1996 [103]. In fact Rao et al. introduce three closely related methods.

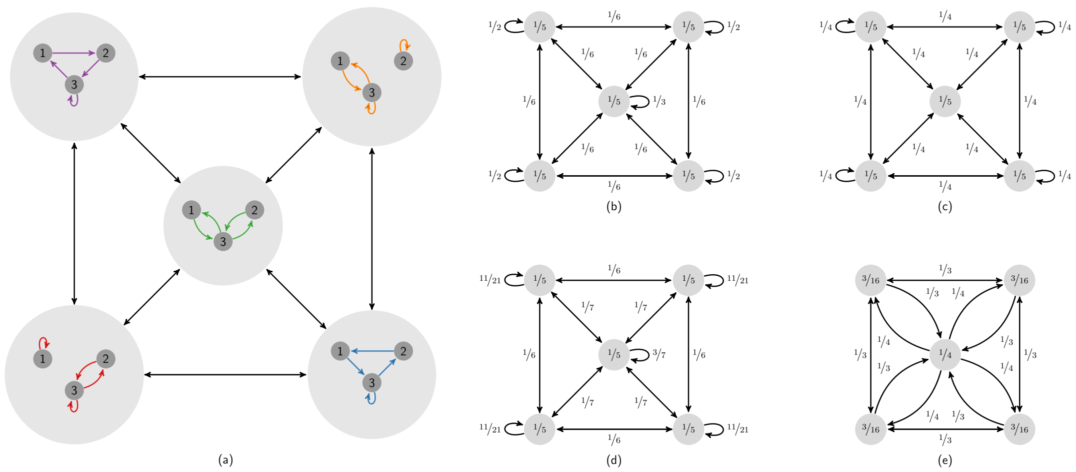


Figure 3.2.: (a) The structure of the state graph of the switching model for networks with $k^{in} = k^{out} = (1, 1, 2)$. (b)-(e) The state graphs of four different Markov chains: (b) The switching model, (c) Rao's model with $m = 4$, (d) the canonical model and (e) the switch and add model. Edge labels correspond to transition probabilities and node labels to the stationary distribution of the Markov chains.

Definition 3.1.3. Let G be a directed network. **Rao's model** for randomising G is defined by a Markov chain starting at G. The states of this Markov chain are all the directed networks with the same degree sequences as G. If there exists a directed switch between G_i and G_j then the transition probability p_{ij} is given by $\frac{1}{m}$, where m equals $\max_i(m_i)$ or an estimate of $\max_i(m_i)$, and m_i is the mobility of G_i.

Figure 3.2(c) shows the state graph and transition probabilities for a simple example, with $m = \max_i(m_i) = 4$.

In Rao's model, all non-zero transition probabilities are equal to $^1/_m$. Notice that when m equals $\max_i(m_i)$, then this is the maximum transition probability for which $\sum_{j \neq i} p_{ij} \leq 1$ for all i. The state graph of this Markov chain is aperiodic and irreducible. When $m = \max_i(m_i)$ the simplified detailed balance equations hold, and hence, like the switching model, Rao's model converges to the uniform distribution.

The difference between the three methods introduced by Rao et al. lies in the determination of (an estimate) of $m = \max_i(m_i)$. The first method introduced in [103, method

I] uses an estimate for m that we will denote by $\tilde{m}$. This leads to sampling from a distribution with probabilities proportional to $\max(m_i, \tilde{m})$. Notice that the sampling is uniform if $\tilde{m} \geq m$. The second method [103, Method II] is no longer a Markov chain, the probability of a switch is again set to be $1/\tilde{m}$, however $\tilde{m}$ is updated to m_i if $m_i > \tilde{m}$. The authors prove that this stochastic process converges to the uniform distribution. Intuitively this makes sense, since after a certain number of switches $\tilde{m}$ should be a constant, which turns the chain into a Markov chain. Finally their third method [103, Method III] is a combination of methods I and II, where a number of switches is executed to obtain a good estimate for m and then method I is used with this estimate. Figure 3.2(c)) shows the state graph of the Markov chain with $\tilde{m} = m$ for a small network. In the remainder of this chapter we assume that we know the true maximum m.

The next model that we discuss is the canonical model as introduced in [108]. For a discussion of this method with respect to undirected networks, we refer the reader to [31]. The formulation of the canonical method is slightly more complicated than that of the switching model. This added complexity does however make it is easy to adjust the model to be more sophisticated. For instance, the authors show that the model is easily adjusted to incorporate degree-degree correlations.

Definition 3.1.4. Let G be a directed network. The **canonical model** for randomising G is defined by a Markov chain starting at G. The states of this Markov chain are all the directed networks with the same degree sequences as G. If there exists a directed switch between G_i and G_j then the transition probability p_{ij} is given by $(1/m_i)(m_i/m_i+m_j)$, with m_i the mobility of G_i. The probability $1/m_i$ corresponds to the probability of selecting a move from the set of all possible moves at G_i. The probability m_i/m_i+m_j is the probability of accepting the selected move.

The acceptance probabilities in the canonical model are chosen such that the simplified detailed balance equations hold:

$$p_{ij} = \frac{1}{m_i}\frac{m_i}{m_i + m_j} = \frac{1}{m_i + m_j} = \frac{1}{m_j}\frac{m_j}{m_i + m_j} = p_{ji}.$$

Figure 3.2(d) shows the transition probabilities of this Markov chain for a small example. Hence, this Markov chain also converges to the uniform distribution.

Finally we discuss the switch and add model as introduced in [6]. This model is slightly different from the other models in that it involves an additional step after running the Markov chain.

Definition 3.1.5. Let G be a directed network. The **switch and add model** for randomising G is defined by a two-stage process involving a Markov chain starting at G and a weighting step. The states of this Markov chain are all the directed networks with the same degree sequences as G. If there exists a directed switch between G_i and G_j then the transition probability p_{ij} is given by $\frac{1}{m_i}$ with m_i the mobility of G_i. The probability $\frac{1}{m_i}$ corresponds to the probability of selecting a move from the set of all possible moves at G_i. To ensure uniform sampling, the weighting step repeats each state a number of times inversely proportional to its mobility.

The Markov chain component of the switch and add model converges to a distribution in which states with high mobility are more probable then states with low mobility (see Example 2.1.2 and Figure 3.2(e)). The introduction of a weighting step is a clever way to ensure the final distribution is unbiased.

We now have four different Markov chains at our disposal that all converge to the uniform distribution on sets of labelled directed networks with fixed degree sequences. In the next two sections we focus on the mixing time and the run-time of these algorithms, to find out which one is fastest and hence most practical.

3.1.2. Mixing times

In order to create uniform samples using Markov chain based random network models, we need to run a Markov chain until it reaches its uniform distribution. The mixing time, $\tau(\epsilon)$ of a Markov chain gives us an indication of how many steps are needed to get within a distance ϵ from this stationary distribution (see Definition 1.3.9). To find out how long a Markov chain will actually run, we need to multiply its mixing time by the time it takes to run each step. We define $t_s(N)$ as the step run-time for N steps; that is the time it takes to run N steps in a chain. The product $\tau(\epsilon)t_s(1)$ gives us an estimate of how long it will take in practice for the Markov chain to reach its stationary distribution.

It is generally hard to determine the mixing time of Markov chains. In practice, the switching model is stopped after a rather arbitrary number of switching attempts [105]. However, for small networks, we can compute the complete state graph and it is possible to calculate the *exact* mixing time.

In general it is understood that Markov chains with high probability of repeating states have a higher mixing time than ones with lower probability of repeating states [6]. The probability of repeating a state G_i is $p^s_{ii} = 1 - {}^{m_i}\!/_M$ for the switching model, $p^r_{ii} = 1 - {}^{m_i}\!/_m$ for Rao's model, $p^c_{ii} = 1 - \sum_{j,|E_i \cap E_j|=4} {}^1\!/_{m_i+m_j}$ for the canonical model and $p^{sa}_{ii} = 0$ for the switch and add model. Notice that due to the assumptions in Lemma 3.1.2 m_i has to be smaller than M since switching an edge pair $\{(u_2, x), (u_2, y)\}$ results in the same network. This implies $m < M$ too. Thus, both p^r_{ii} and p^{sa}_{ii} are always smaller than p^s_{ii}. The probability p^c_{ii} is not necessarily smaller than p^s_{ii}. Hence we generally expect that the switch and add model and Rao's model have smaller mixing time than the switching model.

We ran an experiment to obtain the state graphs of the Markov chain for 15 small directed networks. This was done by recursively enlisting all neighbouring states of each known state, until no new states are found. We then computed the transition probabilities for each of the four methods. Finally we determined the exact mixing times for each chain by taking powers of the transition matrices.

Table 3.1 shows that for small state graphs, the mixing time of the switching model is nearly always larger than that of Rao's model, the canonical model and the switch and add model. Notice that especially for the larger examples the mixing time of the switching model is much larger than that of the other models. These results are consistent with the theoretical expectations.

Most networks of interest are much larger than the examples tested above. For larger networks, the mixing time can not be computed explicitly and needs to be estimated. We do this by measuring the perturbation of the networks in the Markov chain with respect to the initial network [115].

Networks			**Mixing Times** $\tau(\epsilon)$			
k^{in}	k^{out}	# realizations	Switching	Switch and add	Canonical	Rao
201	111	3	1	20	10	20
121	121	5	19	33	21	10
122	221	5	37	33	21	10
1022	1112	12	30	26	33	15
0222	2112	15	44	20	36	20
1213	2131	13	78	23	40	25
2321	2231	24	89	20	46	20
21003	11112	22	40	17	42	21
22111	11113	150	50	33	63	36
31220	03221	24	89	20	46	20
22212	32130	136	79	27	60	36
11323	30142	21	173	20	49	29
32213	14321	148	154	34	74	49
12531	31143	5	283	33	21	10
33223	31333	828	143	34	76	37

Table 3.1.: Mixing times of the four different Markov chains for small networks. Here $\epsilon = 10^{-6}$. For each small network, the best performing model(s), i.e. with lowest mixing time, is highlighted in green, and worst performing model(s) in red.

Definition 3.1.6. Let $G_i = (V, E_i)$ and $G_j = (V, E_j)$ be two directed networks with $m = |E_i| = |E_j|$. We define the **perturbation score** of G_i and G_j as

$$s_p(G_i, G_j) = {}^{|E_i \cap E_j|}\!/_{m}.$$

In terms of the adjacency matrices A_i and A_j of G_i and G_j the perturbation score corresponds to the fraction of ones that are common to A_i and A_j. The perturbation score can also be generalized to non-square matrices.

The mixing time can now be approximated by the step at which the perturbation score stabilizes. Figure 3.3 shows the perturbation scores of the first 3000 networks in Markov chains for an Erdős-Rényi random network and a power-law random network[2]. No-

[2] Note that in the Barábasi-Albert model m equals the number of edges added at each step, not the

tice that for both the switch and add model and Rao's model, the perturbation score stabilizes faster than that of the switching model.

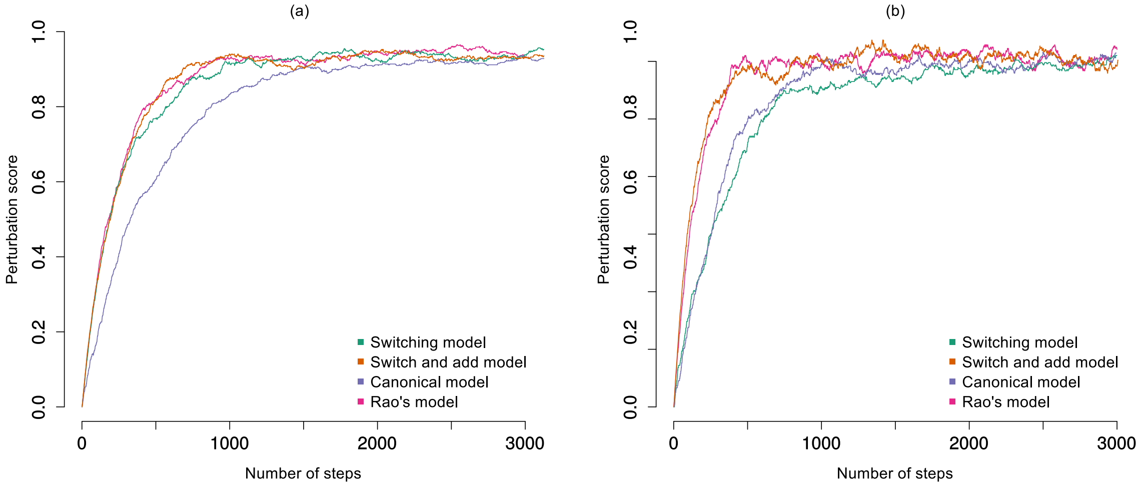


Figure 3.3.: (a) Perturbation scores for a $G(n, p)$ Erdős-Rényi random network: n=100, p=0.05 and (b) perturbation scores for a Barábasi-Albert network with n=100, m=5 [12].

Our comparison of mixing times shows that the switch and add model, and Rao's model generally mix slightly faster than the switching model. However, so far we have ignored the step run-time of all methods. In the next section we compare the step run-time and overall run-time of these four methods.

3.1.3. Run-times

In this section we show that with currently available implementations, the three Markov chains, as defined in Section 3.1.1, are unlikely to improve on the run-time of the switching model.

The outline of our argument is as follows, (1) the three alternative Markov chains rely on being able to *select an allowed switch uniformly at random* in each step. (2) There are several methods available for selecting a switch uniformly at random, but *the fastest available method* selects a switch by randomly checking pairs of edges until an allowed switch is found. (3) Using this method corresponds to a grouping of the steps in the

total number of edges in the network

Markov chain of the switching model, and hence is unlikely to improve on the run-time of the switching model.

(1) To see that all three alternative Markov chains rely on being able to select an allowed switch uniformly at random, see Definitions 3.1.3, 3.1.4 and 3.1.5. Notice that for each of these models, a transition probability is defined *given there exists a directed switch between two networks*. However, none of these methods mentions *how to select* such a switch when running the Markov chain. They all *assume* that it is possible to select an allowed switch (a move from the set of available moves) for the current state at random, and then accept it with the given transition probability.

By inspecting the implementations of these models where available, we found that the selection of a random move is done by trial and error. That is, edge pairs are selected at random until a pair is found that is allowed to be switched. We can think of two alternative methods to select a move at random.

(2) We now discuss these alternative methods and show that neither outperforms the trial and error method mentioned above.

First, after discussions with Prof. Lewi Stone and Dr. Yael Artzy-Randrup a different method for randomly selecting allowed moves emerged: compute the mobility m_i of the current state, randomly pick a number k between 1 and m_i and enlist all allowed switches until we find the k-th allowed switch. Enlisting allowed switches is done by going through all edge pairs in a certain order, while checking if the edges are allowed to be switched. To save time, this should be done from either the beginning of the list or the end of the list, depending on whether k is larger or smaller than half the mobility. We now compare the speed of this method to that of the trial and error method. To do so we derive the expected number of edge pairs that need to be checked in each case.

For the trial and error method, we argue that the number of failures (i.e. selecting an edge pair that is not allowed to be switched) before the first success (finding an edge pair that can be switched) follows a geometric distribution.

Definition 3.1.7. The **geometric distribution** is the discrete probability distribution of the number X of Bernoulli trials, needed to get one success. Let p be the probability of success for each Bernoulli trial, then the geometric distribution is given by:

$$P(X = k) = (1 - p)^{(k-1)} p.$$

The expected value of the geometric distribution equals $^1/_p$.

For the 'trial and error' method, the random selection of an edge pair corresponds to a Bernoulli trial. The probability of success p equals $^{m_i}/_M$, since there are m_i edge pairs that are allowed to be switched corresponding to successes, out of a total of M edge pairs. Hence the expected number of edge pairs that need to be checked by this trial and error method equals $^M/_{m_i}$.

We now discuss the speed of the method where edge pairs are checked until the k-th switch is found. First of all, for this method the mobility of a state m_i needs to be computed. The fastest way to find the mobility of the networks in the Markov chain is by using a formula that calculates the change in mobility after each switch and updates the mobility accordingly [108]. In [108] formulas are introduced to calculate the change in mobility for simple directed networks. In Section 3.1.4 we derive a much simpler formula for directed networks. Calculating the change in mobility is generally speaking very fast, and hence we ignore it in our comparison.

The number of edge pairs needed to be checked before finding the k-th switch follows a negative hypergeometric distribution.

Definition 3.1.8. The **negative hypergeometric distribution** is the discrete probability distribution of the number of selections required until the k-th success is obtained, when sampling without replacement from a set of L objects of which K are successes. The hypergeometric distribution is given by:

$$P(X = x) = \frac{\binom{K}{k-1}\binom{L-K}{x-k}}{\binom{L}{x-1}} \frac{K - k + 1}{L - x + 1}$$

and its expected value equals $k\ ^{L+1}/_{K+1}$.

We make the assumption that the allowed switches are spread evenly throughout the list of edge pairs. Hence, there are ${}^{M}\!/\!{}_{2}$ different edge pairs to select from, that is $L = {}^{M}\!/\!{}_{2}$. Out of these $\frac{m_i}{2}$ correspond to successes, an edge pair allowed to be switched, and $K = {}^{m_i}\!/\!{}_{2}$. Thus the expected number of edge pairs we need to check equals

$$k\,\frac{\frac{M}{2}+1}{\frac{m_i}{2}+1} = k\,\frac{M+2}{m_i+2}.$$

The expected value for k equals ${}^{m_i}\!/\!{}_{4}$ since if $k > \frac{m_i}{2}$ we start from the end of the list. Hence the expected number of pairs to be checked for the 'enlisting method' equals $\left({}^{m_i}\!/\!{}_{4}\right)\left({}^{M+2}\!/\!{}_{m_i+2}\right)$.

Now, if the mobility of a network in the Markov chain is greater than or equal to six, the expected number of edge pairs to check is greater for the the enlisting method than for trial and error method. At $m_i = 6$,

$$\left({}^{m_i}\!/\!{}_{4}\right)\left({}^{M+2}\!/\!{}_{m_i+2}\right) = \frac{6}{4}\frac{M+2}{8} = \frac{3}{16}M + \frac{3}{8} > \frac{M}{6} = \frac{M}{m_i}$$

Hence, the trial and error method is usually faster, since m_i will almost always be greater than five.

Second, we can think of at least one other potential method for the random selection of moves. Instead of enlisting all allowed switches at each step, we could store the allowed switches in memory, and update them after applying a switch. For this method there is a trade-off between speed and memory, since all allowed switches need to be stored in memory. At this stage we are not aware of a method to update the allowed switches in a network after applying a switch and hence we were not able to test this approach. This method may be a worthwhile approach, however one has to keep in mind that the number of allowed switches in a network can be huge. For instance, we found over a million allowed switches for a relatively small $G(n, p)$ Erdős-Rényi random network with 100 nodes and $p = 0.2$. We expect that updating this list will be very slow.

We now show that by using the trial and error method of selecting an allowed switch, the three alternative Markov chains are unlikely to improve on the run-time of the switching model.

(3) Selecting an allowed switch by trial and error, is exactly what the switching model does. That is, at each step in the Markov chain of the switching model, an edge pair is selected at random and switched if possible. Finding an allowed switch by trial and error *within* each step of the Markov chain just corresponds to grouping the steps, as illustrated in Figure 3.4.

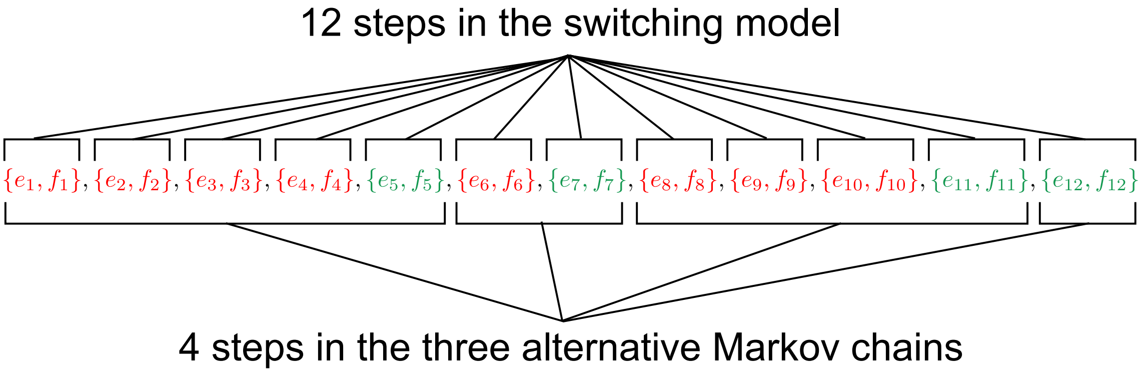

12 steps in the switching model
$\{e_1, f_1\}, \{e_2, f_2\}, \{e_3, f_3\}, \{e_4, f_4\}, \{e_5, f_5\}, \{e_6, f_6\}, \{e_7, f_7\}, \{e_8, f_8\}, \{e_9, f_9\}, \{e_{10}, f_{10}\}, \{e_{11}, f_{11}\}, \{e_{12}, f_{12}\}$
4 steps in the three alternative Markov chains

Figure 3.4.: Conceptual illustration of twelve steps in a switching model. At each step a pair, $\{e_i, f_i\}$, of edges is selected. Red edge pairs are not allowed to switch, whereas green edge pairs are. Using the trial and error method for the three alternative Markov chains corresponds to a grouping of the steps of the switching model. That is, for the same sequence of randomly selected edge pairs, a step does not finish until an allowed switch is found. Thus, the above sequence of attempted switches corresponds to just four steps in the three alternative Markov chains.

The repeated states that increase the mixing time of the switching model as compared to the three alternative Markov chains, are thus still present in these three Markov chains when using the trial and error method to select an allowed switch. The only difference is that they are now ‘hidden’ within the steps of the Markov chain. This implies that the *run-time* of these alternative models is unlikely to be faster than that of the switching model.

Both the switch and add model and the canonical model in addition need to compute the mobility of states, slowing them down as compared to the switching model. Furthermore, for both the canonical model and Rao’s model, once a move is found, it is not always accepted, which is likely to further slow down these methods as compared to the switching model.

To summarize, with currently available algorithms, the run-time of the switching model is unlikely to be improved on by the three Markov chains discussed here. In fact, all Markov chains that allow the same set of moves (i.e. allowed switches) as the switching model, are slowed down by repeated states just as much as the switching model is. Unless we find a faster way to randomly select a move, it is unlikely to see an improvement on the run-time of the switching model by a Markov chain with the same set of moves as the switching model.

In Section 4.1 we discuss another random network model that does converge faster than the switching model, the Curveball algorithm. The state graph of the Markov chain corresponding to the Curveball algorithm is *structurally different* from that of the switching model.

3.1.4. A formula for change in mobility

In this section we derive a formula for the change in mobility caused by applying one directed switch. This formula is useful in for instance the canonical model. Our formula simplifies the one found in [108].

Lemma 3.1.9. Let G_i and G_j be two directed networks with the same labelled node set $\{1, 2, \dots, n\}$ and that differ by the directed switch $\{(x, y), (u, v)\}$ to $\{(x, v), (u, y)\}$. Let m_i be the mobility of G_i and let A be the adjacency matrix of G_i. The mobility of G_j is given by

$$m_j = m_i + \sum_{r \in \bar{R}} \sum_{c \in \bar{C}} (-1)^{A_{rv}+A_{rc}+A_{uc}}$$

where $\bar{R} = \{r \in \{1, \dots, n\} \backslash \{x, u\} | A_{ry} \neq A_{rv}\}$ and $\bar{C} = \{c \in \{1, \dots, n\} \backslash \{y, v\} | A_{xc} \neq A_{uc}\}$.

Proof. The directed switch between G_i and G_j corresponds to the following change in the adjacency matrix A of G_i.

$$\begin{array}{c} \\ \\ x \\ \\ u \\ \\ \end{array} \begin{array}{c} \begin{array}{ccccc} & y & & v & \end{array} \\ \begin{pmatrix} \ddots & \vdots & \cdots & \vdots & \iddots \\ \cdots & 1 & \cdots & 0 & \cdots \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \cdots & 0 & \cdots & 1 & \cdots \\ \iddots & \vdots & \cdots & \vdots & \ddots \end{pmatrix} \end{array} \qquad \begin{array}{c} \\ \\ x \\ \\ u \\ \\ \end{array} \begin{array}{c} \begin{array}{ccccc} & y & & v & \end{array} \\ \begin{pmatrix} \ddots & \vdots & \cdots & \vdots & \iddots \\ \cdots & 0 & \cdots & 1 & \cdots \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \cdots & 1 & \cdots & 0 & \cdots \\ \iddots & \vdots & \cdots & \vdots & \ddots \end{pmatrix} \end{array}$$

Let r be a row different from x and u and c a column different from y and v. There are 32 distinct configurations as depicted in Figure 3.5.

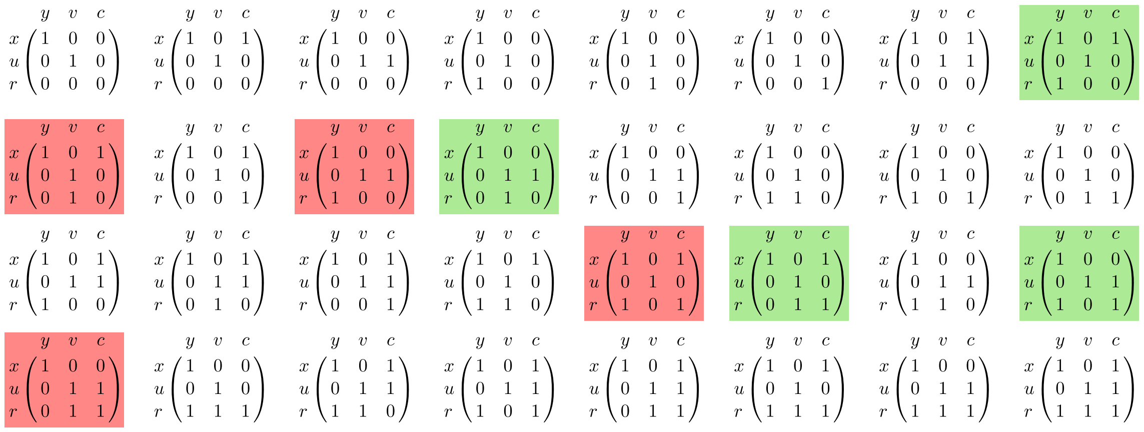


Figure 3.5.: There are 32 different 3×3 binary matrices with fixed entries in positions $(x, y), (x, v), (u, y)$, and (u, v). For each of these configurations, we can count the number of allowed switches. We are interested in the difference between this number and the allowed number of switches for the configuration where (x, y) and (u, v) are switched, i.e. where $(x, y) = (u, v) = 0$ and $(x, v) = (u, y) = 1$. For most of these configurations, the number of allowed switches remains equal. The only configurations where the number of allowed switches changes are the highlighted ones: it either increases by one (green) or decreases by one (red).

One can check directly that the highlighted configurations are the only configurations where applying the directed switch changes the *number* of allowed switches in the configuration. For instance applying the switch to the top right configuration results in the number of switches increasing from three ($\{(x, y), (u, v)\}$, $\{(x, c), (u, v)\}$, $\{(r, y), (u, v)\}$) to four ($\{(x, v), (u, y)\}$, $\{(x, v), (r, y)\}$, $\{(x, c), (u, y)\}$, $\{(x, c), (r, y)\}$). In the configurations that are not highlighted, the allowed switches change, but the number of switches remains equal.

Notice that the eight highlighted configurations are exactly the configurations where $A_{xc} \neq A_{uc}$ and $A_{ry} \neq A_{rv}$. The configurations highlighted with red have the property that applying the switch reduces the number of allowed switches in the configuration from four to three. The configurations highlighted with green on the other hand have the property that applying the switch increases the number of allowed switches from three to four. Furthermore, the configurations highlighted with red all have the property that the sum $A_{uc}+A_{rv}+A_{rc}$ is an odd number, whereas this sum is even for the configurations highlighted with green.

This finishes the proof of the Lemma. □

3.2. The configuration model

The configuration model [95, 84] is a random network model that is often used to generate random networks with a given degree sequence. It generates a network with a given degree sequence from scratch, as opposed to Markov chain models, which randomize a given network by repeatedly making small changes.

The idea of the configuration model is simple. Given the degree sequence of an undirected network, assign to each node v_i exactly k_i 'stubs', where k_i is the degree of the node as usual. Now pair up these stubs randomly to form edges. The result is a network with the desired degree sequence. The directed case is nearly identical: assign to each node 'in-stubs' and 'out-stubs' and pair these at random.

The configuration model is often used as a null-model for simple directed networks or simple undirected networks. However, it does not avoid the creation of self-loops or multiple edges, and in fact often generates networks that contain many multiple edges [56, 69]. That is, it *actually* samples from $\mathcal{G}_6$ or $\mathcal{G}_8$.

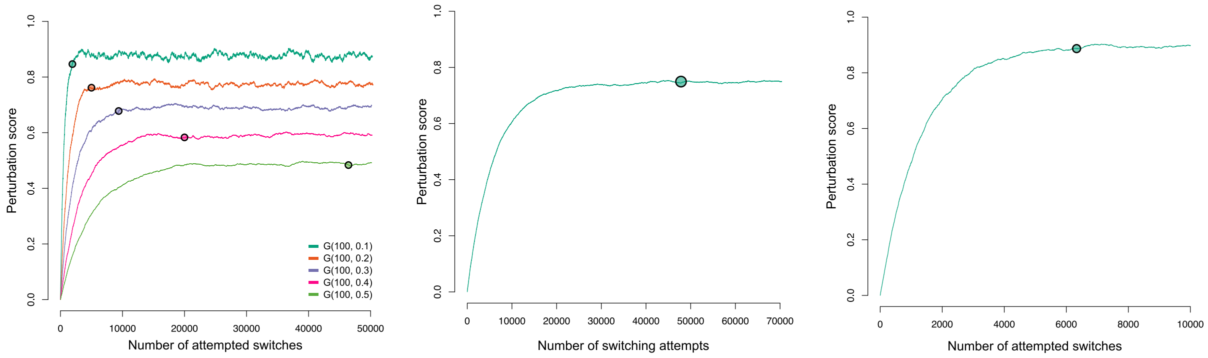
In the limit of large network size, the probability of sampling a network with multiple edges becomes negligible [94, Section 13.2]. However, in practice, when working with finite networks, one hardly ever obtains a simple network from the configuration model.

The fact that in practice the configuration model almost always generates multigraphs makes it unsuitable as a null-model for simple networks. Typically you want to obtain a large sample of random networks to compare to the original network. Using the configuration model to do so would take an impractically long time. However, the configuration model has theoretical properties that allow for the analytical derivation of certain network properties. This is of course extremely useful for hypothesis testing.

The configuration model, as opposed to the switching model and other Markov chains, has the nice property that the network that is generated is not correlated to the original network, since it builds a network from scratch. Furthermore, it runs much faster than the switching model. One run is enough to generate a random network [3].

A recent preprint suggests to combine the configuration model and a Markov chain model to obtain a random simple network (Zhao [133]). Roughly, this *Expand and Contract method* works as follows. Let $G \in \mathcal{G}_1$ and let $S \subset \mathcal{G}_1$ be the set of simple undirected networks with degree sequence equal to that of G. Similarly let $S' \subset \mathcal{G}_6$ be the set of undirected multigraphs with degree sequence equal to that of G. Define a partition $S_0 \sqcup S_1 \sqcup \cdots \sqcup S_k$ of S' where S_i is the set of networks which contain a total of i multiple-edges and self-loops, so that $S_0 = S$. The sampling process then follows the following steps. (1) Randomly sample a network H from S' using the configuration model. (2) Run a Markov chain with respect to S', for instance the switching model with respect to $\mathcal{G}_6$, starting at H, but prefer moves that go down in the partition, i.e. that reduce the number of self-loops and multiple edges. Stop the Markov chain as soon as you have found a network in $S_0 \subset \mathcal{G}_1$. Zhao proves that this method results in uniform sampling.

[3] The configuration model actually does not sample each labelled network with equal probability. However, the probability of sampling any given network can be derived analytically [94, Section 13.2]



(a) Erdős-Rényi random networks

(b) US airlines network

(c) C. Elegans neural network

Figure 3.6.: (a) The perturbation scores for five Erdős-Rényi random networks ($G(n, p)$) for different edge densities p. We use the stabilisation of the perturbation score as an estimate for the Markov chain reaching its stationary distribution. The circles indicate the number of steps needed for the Expand and Contract method to reach a simple network. (b) The same type of plot for the US airlines network. (c) The perturbation scores of the Markov chain of the switching model for randomising the C. Elegans neural network.

This method combines the strengths of the configuration model and the switching model: it uses the speed of the configuration model to generate a random multigraph that is not at all correlated to the original network, and then uses the flexibility of the switching model to obtain a simple network.

We now discuss experiments that compare the run-time of the Expand and Contract method to the run-time of the switching model. We want to know how many steps are needed in part (2) of the Expand and Contract method to obtain a simple network. In particular, we want to know how this compares to the mixing time of the switching model, i.e. the number of steps the switching model needs to take before reaching the uniform distribution. We ran the Expand and Contract method and the switching model on seven different networks; five $G(n, p)$ random networks with $n = 100$ and p varying between 0.1 and 0.5, and two real networks, a US transportation network (see Appendix A.10) and the neural network of C. Elegans (see Appendix A.2). We use the stabilisation of the perturbation score as an estimate for the number of steps needed for the switching model to reach its stationary distribution. Figure 3.6 shows the results from our experiments.

Notice that for sparse $G(n,p)$ Erdős-Rényi random networks ($p = 0.1$, $p = 0.2$), the Expand and Contract method needs somewhat fewer steps to reach a simple network than the switching model needs to reach its stationary distribution. For the Erdős-Rényi networks with $p = 0.3$ and $p = 0.4$ the number of steps for both models is of the same order. However for the dense network $p = 0.5$ it takes many more steps for the Expand and Contract method to find a simple network than it does for the switching model to reach its stationary distribution.

For the US airport network (Figure 3.6(b)) the switching model reaches its stationary distribution in fewer steps than are needed for the Expand and Contract method to obtain a simple network. For the neural network of C. Elegans, the switching model reaches its stationary distribution at approximately the same number of steps as it takes the Expand and Contract method to obtain a simple network.

These experiments indicate that the number of attempted switches needed to obtain a simple network from the initial multigraph is of a similar order as the number of steps needed for the switching model to reach its stationary distribution. However, in certain cases, the Expand and Contract method needs significantly more steps to obtain a simple network. It appears that the larger and the denser the network, the longer the Expand and Contract method takes to find a simple network. This is not surprising, since the run-time is correlated with the probability of selecting multiple edges for switches. In a large or a dense network this probability can be very small.

Our findings indicate that the Expand and Contract method generally does not improve on the run-time of the switching method. However, compared to the switching method, it has the advantage that the random networks it produces are in no way related to the network that is being randomised. The reason for this is that in its first step it uses the configuration model to build a network from scratch.

3.3. Conclusion

We discussed three alternative Markov chain models to the switching model for the randomization of networks. These models are very similar to the switching model as they allow the same set of moves for each state of the Markov chain. However, they differ in their transition probabilities. We showed that even though these models may have faster mixing times than the switching model, their run-time is unlikely to improve on that of the switching model. This apparent contradiction is explained by the fact that these alternative Markov chains ‘hide’ switching attempts inside each step of the Markov chain. We showed that with current techniques, the run-time of the switching model is likely to be the fastest among Markov chains that allow the same set of moves (i.e. allowed switches) as the switching model.

The configuration model is a very fast and useful alternative random network model and should be preferred in cases where a multigraph or a network with fewer edges is not regarded as a problem. However, if a simple network with exactly the same number of nodes and edges is sought, then the switching model is more suitable.

The Expand and Contract method, which promises to combine the strengths of both the configuration model and the switching model, does not seem to improve on the speed of the switching model in general. Specifically, it seems impractical for dense networks and for large networks. We have verified this experimentally, however, it would be very useful to have theoretical results about their relative runtimes.

The Expand and Contract method uses the configuration model as its first step, hence it similarly produces random networks that are in no way related to the original network. This is an advantage compared to the switching model, because it is unknown how many steps are necessary for the switching model to produces a truly uncorrelated network. In situations where it is practical to use the Expand and Contract method, i.e. where the time it takes to find a simple network is not prohibitive, it may be the preferred choice of null-model.

However, the Expand and Contract method has so far only appeared in a preprint, and hence care must be taken in adopting this method. In particular, we recommend thorough checks of the claim that it samples without bias.

None of the alternative random network models presented in this chapter improved significantly on the speed of the switching model with respect to simple networks. In the next chapter we will discuss a different Markov chain model that *does*.

4. Curveball algorithms

In Chapter 3 we discussed the run-time of Markov chains designed to randomise directed networks ($\mathcal{G}_4$). In this Chapter we discuss a recently introduced model that improves on the speed of the switching model: the *Curveball algorithm* [115].

We show that the Curveball algorithm corresponds to a Markov chain with a different state graph from that of the switching model. In fact, we show that the state graph of the switching model with respect to $\mathcal{G}_4$ is a subgraph of the state graph of the Curveball algorithm. We present a proof that the Curveball algorithm converges to the uniform distribution. This proof is published in [26].

Furthermore, we show that, similarly to the switching model, the Curveball algorithm provides us with a flexible framework for network randomization. We define variations of the Curveball algorithm to randomize networks in classes $\mathcal{G}_1, \mathcal{G}_3, \mathcal{G}_5, \mathcal{G}_7$ and $\mathcal{G}_8$. We will refer to the Curveball algorithm with respect to $\mathcal{G}_4$ as *the (original) Curveball algorithm*, the Curveball algorithm with respect to $\mathcal{G}_3$ as *the Simple Curveball algorithm* and the Curveball algorithm with respect to $\mathcal{G}_1$ as *the Simple Undirected Curveball algorithm.*

This chapter is organised as follows; Section 4.1 describes the Curveball algorithm and presents the proof for uniform sampling. Section 4.2 discusses three variations of the original Curveball algorithm for the randomization of several directed network classes. Section 4.3 discusses two variations of the Curveball algorithm for the randomization of undirected networks. In Section 4.4 we discuss the run-time of the Curveball algorithm. Finally, we draw conclusions in Section 4.5.

My contributions in this chapter are the following.

- Rephrasing the Curveball algorithm in terms of Markov chain language and proving that it converges to the uniform distribution. This result is published in [26].

- Introducing three new versions of the Curveball algorithm for directed networks, including the Simple Curveball algorithm, an adjusted version of the Curveball algorithm that randomizes *simple* directed networks. This work is in collaboration with Dr. Giovanni Strona, and has not yet been published.

- Introducing algorithms similar to the Curveball algorithm for the randomization of undirected networks without self-loops ($\mathcal{G}_3, \mathcal{G}_7$). This work has not yet been published.

4.1. The Curveball Algorithm

We now discuss the Curveball algorithm, a random network model for directed networks with fixed degree sequence. The Curveball algorithm was introduced in [115] and was shown to converge to its stationary distribution much faster than the switching model. Strona et al. claimed that this algorithm samples uniformly and supported their claim by numerical experiments. This section presents a proof of unbiased sampling for the Curveball algorithm.

The Curveball algorithm was developed to randomize binary matrices with fixed row and column sums. The special case of square matrices corresponds to randomizing a directed network with fixed degree sequence. In other words, the Curveball algorithm may be used to randomize networks from class $\mathcal{G}_4$. The original Curveball algorithm can not be used to randomize undirected networks, since the randomization steps do not preserve the symmetry of the adjacency matrix of an undirected network. However, we will introduce versions of the Curveball algorithm with respect to undirected networks in Section 4.3. We will describe the Curveball algorithm in terms of binary matrices, keeping in mind that we are actually interested in using it to randomize directed networks.

This Section is organised as follows, we first describe the Curveball algorithm and the corresponding Markov chain. We compare it to the switching model with respect to $\mathcal{G}_4$ and use the similarities between the two algorithms to prove that the Curveball algorithm samples uniformly. Finally we discuss two modifications of the Curveball algorithm and their impact on the convergence speed and stationary distribution.

4.1.1. Description of the Curveball algorithm

The Curveball algorithm randomizes a binary matrix A using the following steps: (a) transform A into sets of indices, A_i, for each row i, corresponding to ones in that row [1], (b) select two of these sets A_i and A_j at random, (c) compare the sets and let A_{i-j} be all indices that are in A_i but not in A_j. Similarly define A_{j-i}. (d) Create a new sets A'_i by removing A_{i-j} from A_i and adding the same number of elements randomly chosen from $A_{i-j} \cup A_{j-i}$. Combine $A_j \backslash A_{j-i}$ with the remaining elements of $A_{i-j} \cup A_{j-i}$ to form A'_j [2]. (e) Reiterate step (b)-(d) a certain number of times and (f) form a new matrix from the resulting sets. An example is given in Figure 4.1.

We will refer to one iteration of step (b)-(d) as a **trade** and the number of exchanged indices, $|A'_i \backslash A_i| = |A'_j \backslash A_j|$ as the **size** of the trade. Notice that after each iteration of steps (b)-(d) we can form a binary matrix with the same row and column sums as A.

The Curveball algorithm for a binary matrix A corresponds to a finite discrete-time Markov chain with state space all binary matrices with row and column sums equal to those of A. One iteration of step (b)-(d) corresponds to a transition to the next

[1] Depending on the dimensions of the matrix it may be faster to make sets based on the columns and depending on the number of zeros and ones in the matrix it may be faster to make sets of the zeros.

[2] This explicit description of step (d) is based on the implementation of the Curveball algorithm (see Appendix B.2.1)

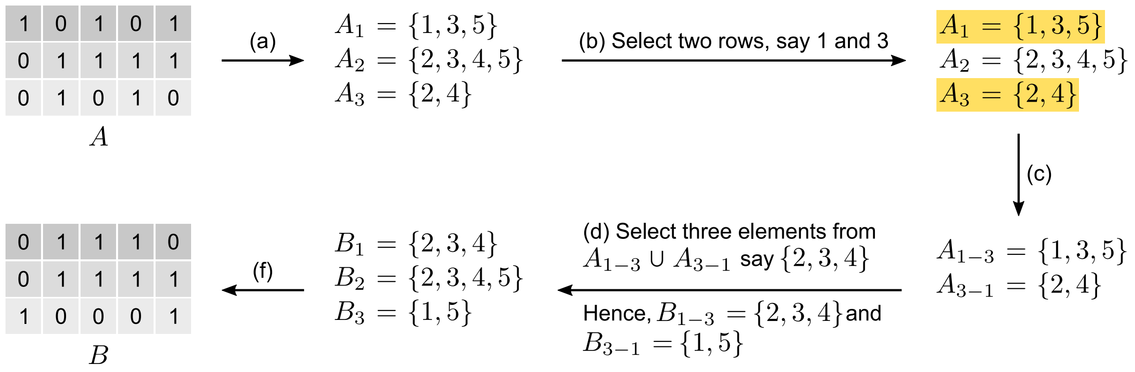


Figure 4.1.: A *trade* in the Curveball algorithm consists of (a) converting a binary matrix into sets of indices, A_i, for each row i. In step (b) two rows are selected, in this case row 1 and 3. (c) The set differences A_{1-3} and A_{3-1} are extracted. (d) The set B_1 is formed by removing A_{1-3} from A_1 and adding $|A_{1-3}|$ elements randomly chosen from $A_{1-3} \cup A_{3-1}$, in this case $\{2,3,4\}$. B_3 is formed by removing A_{3-1} from A_3 and adding the remaining elements of $A_{1-3} \cup A_{3-1}$. Step (f) converts the resulting sets of indices B_i into the matrix B.

state. Notice that it is possible for consecutive matrices to be equal, since either A_{i-j} or A_{j-i} could be empty, and even if they are not, the elements randomly selected from $A_{i-j} \cup A_{j-i}$ to create A'_i could be exactly the elements in A_{i-j}. This Markov chain is uniquely described by its transition matrix $P = [P_{AB}]$, with P_{AB} the transition probability from state A to B, for all states. For the Curveball algorithm, this probability is non-zero if A and B differ in exactly two rows or if they are identical, otherwise it is zero. The transition probabilities correspond to the probability of selecting the specific trade that transforms A into B. In Section 4.1.1 we derive an explicit formula for P_{AB}.

Comparison to the switching method

The proof of uniform sampling for the Curveball algorithm is similar to the proof of uniform sampling for the switching model with respect to class $\mathcal{G}_4$. As mentioned in [115], the Curveball algorithm and switching method are 'in a sense closely related'. Not only do they correspond to Markov chains on the same state space, every switch of the switching model corresponds to a trade in the Curveball algorithm and every trade in the Curveball algorithm corresponds to a sequence of switches, potentially of length zero, of the switching model. This implies that the state graph of the switching model is a subgraph of the state graph of the Curveball algorithm. For an illustration see Figure 4.2. In the next section we exploit this similarity to prove that the Curveball algorithm samples uniformly.

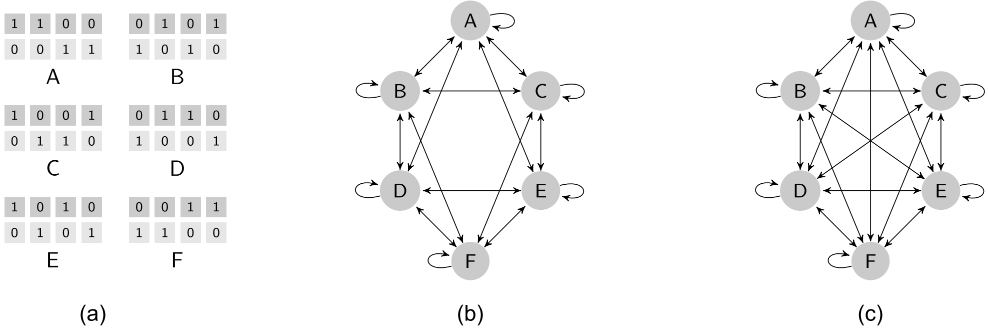


Figure 4.2.: (a) There are six different matrices with row sums $(2, 2)$ and column sums $(1, 1, 1, 1)$. (b) The structure of the state graph for the switching method. Notice that there are no switches between state A and F, between state B and E, and between state C and D since these pairs of matrices differ in two switches. (c) The structure of the state graph of the Curveball algorithm. There are trades between A and F, between state B and E, and between state C and D.

Proof of uniform sampling

We start this Section by showing that the Markov chain corresponding to the Curveball algorithm is irreducible and aperiodic. We then derive the transition probabilities and show that the simplified detailed balance equations hold.

Lemma 4.1.1. The Markov chain corresponding to the Curveball algorithm is irreducible.

Proof. The state graph of the Curveball algorithm is strongly connected since it has a strongly connected subgraph that includes all its nodes. This is the subgraph corresponding to the state graph of the switching model. □

Lemma 4.1.2. The Markov chain corresponding to the Curveball algorithm is aperiodic.

Proof. The Markov chain corresponding to the Curveball algorithm is trivially aperiodic since step (d) ensures that there is a non-zero probability to repeat each state. □

It remains to show that the simplified balance equations hold. To do so, we first derive the transition probabilities.

Lemma 4.1.3. Let A and B be two binary matrices with equal row and column sums. The transition probability P_{AB} from A to B is given by

$$P_{AB} = \begin{cases} \frac{2}{r(r-1)} \frac{s_i! s_j!}{(s_i+s_j)!} & \text{if } A \text{ and } B \text{ differ only in row } i \text{ and } j, \\ 1 - \sum_{C, C \neq A} P_{AC} & \text{if } A = B, \\ 0 & \text{otherwise.} \end{cases}$$

with r the number of rows of A and $s_i = |A_{i-j}|$ and $s_j = |A_{j-i}|$ [3].

Proof. The probability of selecting row i and j for a trade is $^2/_{r(r-1)}$. In step (d) the set A_{i-j}, of indices in row i but not in row j, is taken from A_i and put together with the set A_{j-i} taken from A_j. The resulting set is shuffled, the first s_i elements are returned to A_i, and the remaining s_j elements to A_j. The probability that shuffling results in state B, equals the inverse of the number of ways you can select s_i unordered elements from a set of $s_i + s_j$ elements. This probability is exactly $^{(s_i+s_j)!}/_{s_i! s_j!}$. □

Notice that P_{AB} only depends on r, s_i and s_j. It is straightforward to see that these are equal for the reversed trade, and thus $P_{AB} = P_{BA}$, (see also Figure 4.4(a)). Theorem 4.1.4 follows from Corollary 1.3.7.

Theorem 4.1.4. *The Markov chain corresponding to the Curveball algorithm converges to the uniform distribution.* □

[3] These formulae for the transition probabilities are not in correspondence with those presented in [115] for the small example of matrices with row and column sum equal to $(1, 2, 1)$. However, these formulae correspond to the algorithm as found in the Supplementary code of [115]. The probabilities presented in [115] correspond to the Good-Shuffle algorithm discussed in Section 4.1.2.

4.1.2. Modifying the Curveball algorithm

In this section we discuss the effect of two modifications of the Curveball algorithm. When the probabilities of repeating states are high, the mixing time of a Markov chain increases [6]. It is thus desirable to refrain from unnecessary repetitions. The following two situations cause repeated states in the Curveball algorithm: firstly, when two rows are selected that do not allow any trades and secondly when the shuffling of $A_{i-j} \cup A_{j-i}$ results in sets $A'_i = A_i$ and $A'_j = A_j$, in other words, when this shuffling leaves row i and j unchanged. We will refer to the former as *no-trade row-pairs* and to the latter as *no-trade shuffles.*

Excluding no-trade Shuffles

We first show that the Curveball algorithm may be adjusted by excluding no-trade shuffles, and that this modification does not affect random sampling for all matrices except a pathological class P of matrices. The no-trade shuffles can be removed by modifying step (d). Instead of always accepting the newly created sets A'_i and A'_j, this step should be repeated until $A'_i \neq A_i$ and $A'_j \neq A_j$, in other words until an actual trade has been made. The transition probability for distinct neighbouring states A and B then becomes

$$P_{AB} = \frac{2}{r(r-1)} \frac{s_i! s_j!}{(s_i + s_j)! - (s_i! s_j!)}$$

Again P_{AB} only depends on r, s_i and s_j and thus $P_{AB} = P_{BA}$. Furthermore, this Markov chain is irreducible by the same argument as before. Finally, we need to find out under which conditions this chain is aperiodic. The argument for the aperiodicity of the Curveball algorithm can not be used, since it relied on the inclusion of no-trade shuffles.

Lemma 4.1.5. Let P be the class of matrices with column sums $(1, \ldots, 1, 0, \ldots, 0, r, \ldots, r)$ with exactly r columns summing to 1, up to reordering of columns. Here r is the number of rows of the matrix. The Markov chain corresponding to the Curveball algorithm excluding no-trade shuffles for a matrix A is periodic if and only if $A \in P$.

Proof. It is clear that the Markov chain is aperiodic if A contains no-trade row-pairs. Furthermore, if A contains a row-pair i, j such that $\max(|A_{i-j}|, |A_{j-i}|) > 1$ then there are two states that differ from A by exactly one trade, and that also differ from each other by a trade. This is illustrated in Figure 4.3 and implies that the Markov chain is aperiodic.

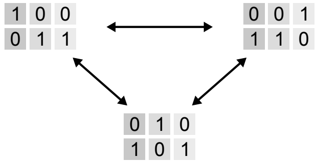


Figure 4.3.: When a matrix A contains two rows i and j with $\max(|A_{i-j}|, |A_{j-i}|) > 1$ it contains, up to row permutations, a submatrix of one of the forms above. This implies that there is a path of three trades as well as a path of two trades starting and ending at A. Thus state A is aperiodic.

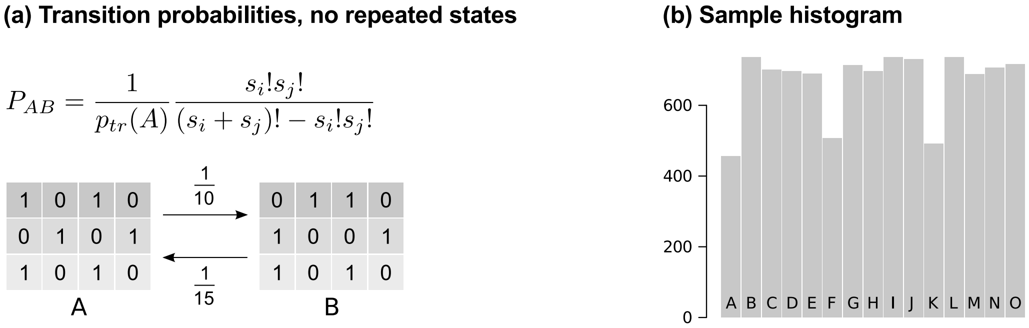
Thus the only matrices A for which the Markov chain could potentially be periodic are those for with $|A_{i-j}| = |A_{j-i}| = 1$ for all row-pairs i and j. Such matrices correspond, up to permutations of the columns, to an identity matrix and columns of all ones or all zeros. These are exactly the matrices in class P.

To see that the Markov chain is indeed periodic for matrices $A \in P$, notice that columns consisting of all ones or all zeroes are left invariant by the Curveball algorithm. Thus without loss of generality we may assume that A is an r by r identity matrix. It is easy to see that each trade corresponds to a column-swap, in other words a transposition of columns. The identity permutation is even and can thus only be formed by an even number of transpositions of columns, which means exactly that this Markov chain is 2-periodic. □

To summarize, no-trade shuffles may almost always be removed from the Curveball algorithm without affecting the stationary distribution of the Markov chain. In practice all matrices of interest are randomized without bias. From now on, we will refer to this modified Curveball algorithm, for matrices not in P, as the Good-shuffle Curveball algorithm [4].

[4] The Good-Shuffle algorithm is publicly available, see Appendix B.1.1

Excluding no-trade row-pairs


(a) Transition probabilities, no repeated states

(b) Sample histogram

$P_{AB} = \frac{1}{p_{tr}(A)} \frac{s_i!s_j!}{(s_i+s_j)! - s_i!s_j!}$
1 0 1 0
0 1 0 1
1 0 1 0
A
$\frac{1}{10}$
$\frac{1}{15}$
0 1 1 0
1 0 0 1
1 0 1 0
B
600
400
200
0
A B C D E F G H I J K L M N O

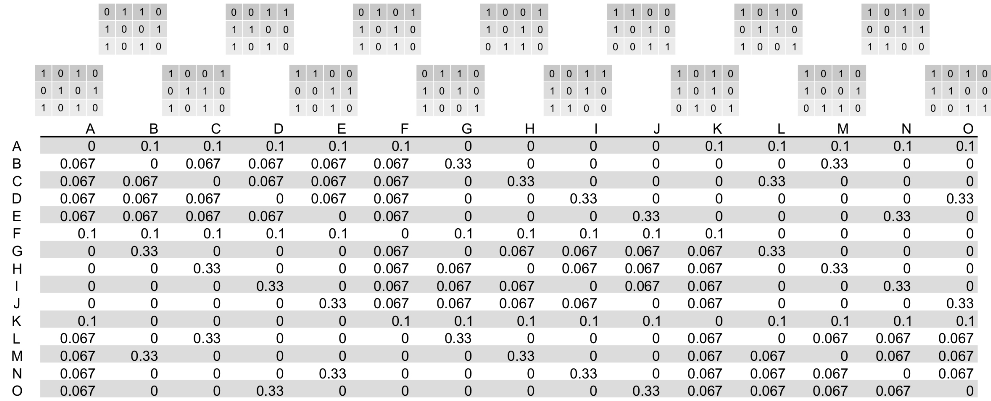
Figure 4.4.: (a) When all repeated states are excluded from the Curveball algorithm, P_{AB} is not always equal to P_{BA}. In this example, both A_{1-2} and A_{2-1} contain two elements, thus $s_1 = s_2 = 2$. There are two row pairs in A that can make trades, resulting in $P_{AB} = 1/10$. This does not equal P_{BA}, since there are three row pairs in B that can make trades and hence $P_{BA} = 1/15$. (b) The number of binary matrices with row sums $(2, 2, 2)$ and column sums $(2, 1, 2, 1)$ in a biased sample, generated by the modified Curveball algorithm, where all repeated states are excluded. The sample consists of 10,000 matrices sampled at every 1,000th trade of the Markov chain.

We now show that repeats caused by no-trade row-pairs should not be excluded from the Curveball algorithm, otherwise sampling is no longer guaranteed to be uniform. This is in contradiction with a comment made by Strona. et al that removing all repeated states does not affect the Curveball algorithm. Their argument is made plausible by presenting the transition matrix of a single example where all repeats are removed [115, Supplementary Information, Table 1]. However, it is a coincidence that sampling is uniform for this example. We give another example to show that in general sampling may be biased.

To remove repeats caused by no-trade row-pairs, step (b) is modified: instead of randomly selecting any row-pair, randomly select a row-pair that can make trades. The transition probability for distinct neighbouring states A and B then becomes

$$P_{AB} = \frac{1}{p_{tr}(A)} \frac{s_i!s_j!}{((s_i+s_j)! - s_i!s_j!)}$$

with $p_{tr}(A)$ the number of row-pairs in A that can make trades. In this Markov chain, P_{AB} is no longer guaranteed to equal P_{BA}. See Figure 4.4(b) for an example where



	1 0 1 0 0 1 0 1 1 0 1 0	0 1 1 0 1 0 0 1 1 0 1 0	1 0 0 1 0 1 1 0 1 0 1 0	0 0 1 1 1 1 0 0 1 0 1 0	1 1 0 0 0 0 1 1 1 0 1 0	0 1 0 1 1 0 1 0 1 0 1 0	0 1 1 0 1 0 1 0 1 0 0 1	1 0 0 1 1 0 1 0 0 1 1 0	0 0 1 1 1 0 1 0 1 1 0 0	1 1 0 0 1 0 1 0 0 0 1 1	1 0 1 0 1 0 1 0 0 1 0 1	1 0 1 0 0 1 1 0 1 0 0 1	1 0 1 0 1 0 0 1 0 1 1 0	1 0 1 0 0 0 1 1 1 1 0 0	1 0 1 0 1 1 0 0 0 0 1 1
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
A	0	0.1	0.1	0.1	0.1	0.1	0	0	0	0	0.1	0.1	0.1	0.1	0.1
B	0.067	0	0.067	0.067	0.067	0.067	0.33	0	0	0	0	0	0.33	0	0
C	0.067	0.067	0	0.067	0.067	0.067	0	0.33	0	0	0	0.33	0	0	0
D	0.067	0.067	0.067	0	0.067	0.067	0	0	0.33	0	0	0	0	0	0.33
E	0.067	0.067	0.067	0.067	0	0.067	0	0	0	0.33	0	0	0	0.33	0
F	0.1	0.1	0.1	0.1	0.1	0	0.1	0.1	0.1	0.1	0.1	0	0	0	0
G	0	0.33	0	0	0	0.067	0	0.067	0.067	0.067	0.067	0.33	0	0	0
H	0	0	0.33	0	0	0.067	0.067	0	0.067	0.067	0.067	0	0.33	0	0
I	0	0	0	0.33	0	0.067	0.067	0.067	0	0.067	0.067	0	0	0.33	0
J	0	0	0	0	0.33	0.067	0.067	0.067	0.067	0	0.067	0	0	0	0.33
K	0.1	0	0	0	0	0.1	0.1	0.1	0.1	0.1	0	0.1	0.1	0.1	0.1
L	0.067	0	0.33	0	0	0	0.33	0	0	0	0.067	0	0.067	0.067	0.067
M	0.067	0.33	0	0	0	0	0	0.33	0	0	0.067	0.067	0	0.067	0.067
N	0.067	0	0	0	0.33	0	0	0	0.33	0	0.067	0.067	0.067	0	0.067
O	0.067	0	0	0.33	0	0	0	0	0	0.33	0.067	0.067	0.067	0.067	0

Figure 4.5.: The transition probabilities of the Curveball algorithm without repeated states, for all matrices with row sums $(2,2,2)$ and column sums $(2,1,2,1)$.

they differ. Figure 4.5 shows all binary matrices with row sums $(2,2,2)$ and column sums $(2,1,2,1)$, and the transition probabilities for the Curveball algorithm without repeated states (i.e. without no-trade row-pairs and no-trade shuffles). The Markov chain corresponding to this example is irreducible and aperiodic, as can be checked from its transition matrix in Figure 4.5. Furthermore, one can verify that the detailed balance equations hold for $\pi = {}^1\!/_{42}(2,3,3,3,3,2,3,3,3,3,2,3,3,3,3)$. Thus, the Markov chain converges to π and not to the uniform distribution. It is less likely to generate matrix A, F or K than the other matrices, as shown experimentally in Figure 4.4(b).

4.1.3. Numerical results

We now compare the performance of the Curveball and the Good-Shuffle Curveball algorithm. In Table 4.1 the mixing time (see Definition 1.3.9) and step runtime (see Section 3.1.2) of the Curveball and Good-Shuffle algorithms are listed for nine small matrices. The Good-Shuffle Curveball algorithm generally mixes faster, which can be explained by the exclusion of some of the repeated states. However, there is a trade-off in terms of step runtime, since step (d) is more complicated for the Good-Shuffle algorithm. Indeed, Table 4.1 clearly shows that the step runtime is longer for the modified algorithm than for the original algorithm. Overall, the Good-Shuffle algorithm almost always outperforms the original Curveball algorithm.

			Curveball			**Good-Shuffle**		
Row sums	Column sums	State count	Mixing $\tau(\epsilon)$	Step $t_s(n)$	Total $t_s(\tau(\epsilon))$	Mixing $\tau(\epsilon)$	Step $t_s(n)$	Total $t_s(\tau(\epsilon))$
1,2,1	1,2,1	5	33	0.668	0.022	19	0.888	0.017
2,2,1,1	1,2,1,2	34	48	0.704	0.034	30	0.963	0.029
3,1,3	2,2,2,1	7	74	0.520	0.038	54	0.628	0.034
2,2,2,2	1,1,4,2	12	35	0.734	0.026	13	1.104	0.014
2,1,0,2,1	3,1,0,1,1	18	95	0.528	0.050	59	0.655	0.039
3,2,1,1,1	1,1,3,2,1	141	95	0.821	0.082	69	0.875	0.060
2,2,0,3,2	2,2,3,1,1	120	99	0.539	0.053	77	0.710	0.055
1,1,4,2,2	2,3,2,1,2	198	101	0.666	0.067	75	0.874	0.066
3,1,2,4,3	2,4,2,3,2	237	92	0.647	0.059	64	0.855	0.055

Table 4.1.: Mixing times and run times of the Curveball and Good-Shuffle algorithms for nine small matrices. Here $\epsilon = 10^{-6}$ and $n = 1000$. The total run-times are approximate, computed from the mixing and step times. Specifically $t_s(\tau(\epsilon))$ is computed as the product of $\tau(\epsilon)$ and $t_s(n)/n$.

Most matrices of interest are much larger than the above examples. Besides, if all matrices with given row and column sum can be enumerated, there is no need to use a Markov chain; sampling can be done directly. For larger matrices, the mixing time needs to be estimated. This can be done by measuring the perturbation of each matrix in the Markov chain with respect to the initial matrix (see Section 3.1.2). The mixing time is approximated by the step at which the perturbation score stabilizes.

Figure 4.6 shows that for a 10 × 10 matrix, the perturbation scores of the Curveball and Good-Shuffle algorithm stabilize at roughly the same time. For a 100 × 100 matrix the perturbation scores are indistinguishable. This suggests that for larger matrices, it takes roughly the same number of steps to reach the uniform distribution for both algorithms. There is a good explanation for this: the sizes s_i and s_j of the sets A_{i-j} and A_{j-i} are larger for larger matrices. For large s_i and s_j, the difference between transition probabilities

$$P_{AB}^{Good-Shuffle} - P_{AB}^{Curveball} = \frac{2}{r(r-1)} \left(\frac{s_i!s_j!}{(s_i+s_j)! - (s_i!s_j!)} - \frac{s_i!s_j!}{(s_i+s_j)!} \right),$$

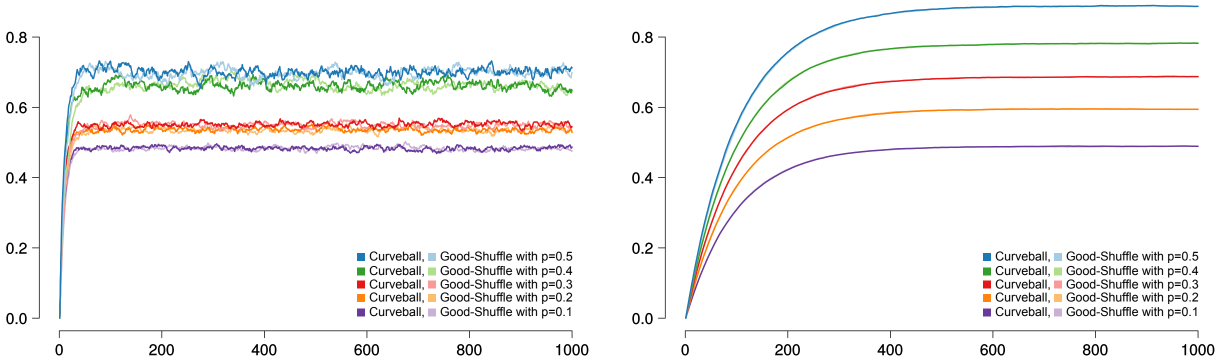


Figure 4.6.: Due to the large variance in perturbation scores for different runs of each Markov chain, average perturbation scores over 100 runs are shown. (a) Five random 10×10 binary matrices were created, by letting each matrix entry be one with probability either 0.1, 0.2, 0.3, 0.4 or 0.5. For each matrix, the perturbation scores of the Curveball and Good-Shuffle algorithm stabilize at roughly the same point, after about 50 steps. (b) The same experiment was repeated for 100×100 matrices, here the perturbation scores of the Curveball and Good-Shuffle algorithm are indistinguishable.

becomes negligible, since $s_i!s_j! \ll (s_i + s_j)!$.

The step runtime of the Good-Shuffle algorithm is still longer than that of the Curveball algorithm. Thus, the Curveball algorithm runs faster than the Good-Shuffle algorithm.

4.2. Curveball algorithms for directed networks

We first focus on simple directed networks, i.e. class $\mathcal{G}_3$ from Chapter 2. We would like to use the fast Curveball method, instead of the switching model, to randomize such networks. However, certain trades of the Curveball algorithm will create self-loops (trades that introduce diagonal elements in the adjacency matrix). By making a small adjustment to the Curveball algorithm, we can use it as a random network model for *simple* directed networks ($\mathcal{G}_3$). We call this algorithm the Simple Curveball algorithm.

The Curveball algorithm was designed to randomize binary matrices. Here we are interested in randomizing the $n \times n$ adjacency matrix A of a simple directed network $G = (V, E)$, where n is the number of nodes in the network. The matrix A has zeroes on its diagonal, since G does not contain self-loops. We now describe the Simple Curveball algorithm, which only differs from the Curveball algorithm in step (c) (see Section 4.1.1).

The Simple Curveball algorithm uses the following steps. Let A be the adjacency matrix of a simple directed network G. (a) Transform A into sets of indices, A_i for each row i, corresponding to the ones in that row. Note that A_i does not contain i. (b) Select two of these sets A_i and A_j at random. (c) Compare these sets and let A_{i-j} be all indices that are in A_i but that are not in A_j and that are not equal to j. Similarly define A_{j-i}. (d) Create a new set B_i by removing A_{i-j} from A_i and adding the same number of elements randomly chosen from $A_{i-j} \cup A_{j-i}$. Combine $A_j \backslash A_{j-i}$ with the remaining elements of $A_{i-j} \cup A_{j-i}$ to form B_j. (e) Reiterate step (b)-(d) N times, for a certain fixed number N, and (f) form a new matrix from the resulting sets.

It is important to notice the following two properties of the Simple Curveball algorithm. Firstly, the trades that are absent in the Simple Curveball algorithm as compared to the Curveball algorithm are exactly those that introduce self-loops. That is, the indices that would introduce self-loops are explicitly absent from a trade: A_{i-j} does not contain j. Secondly, any switch in the switching model with respect to $\mathcal{G}_3$ corresponds to a trade of size one in the Simple Curveball algorithm: if (x, y) and (u, v) are allowed to be switched, then $y \in A_{x-u}$ and $v \in A_{u-x}$. Hence, if row x and row u are selected for a trade, there is a non-zero probability that this trade coincides with the switch to (x, v) and (u, y). We now derive the transition probabilities of the Simple Curveball algorithm, in order to find its stationary distribution.

Lemma 4.2.1. Let A and B be two $n \times n$ binary matrices with zero diagonal and equal row and column sums. The transition probability P_{AB} from A to B, in the Simple

Curveball algorithm, is given by

$$P_{AB} = \begin{cases} \frac{2}{n(n-1)} \frac{s_i!s_j!}{(s_i+s_j)!} & \text{if } A \text{ and } B \text{ differ only in row } i \text{ and } j, \\ 1 - \sum_{C,C\neq A} P_{AC} & \text{if } A = B, \\ 0 & \text{otherwise.} \end{cases}$$

Here $s_i = |A_{i-j}|$ and $s_j = |A_{j-i}|$.

Proof. The probability of transitioning from a state A to another state B that differs in a trade between row i and j can be found as follows. The probability of selecting row i and j equals ${}^2/_{n(n-1)}$. The probability that shuffling $A_{i-j} \cup A_{j-i}$ results in state B equals the inverse of the number of ways you can select s_i unordered elements from a set of $s_i + s_j$ elements. This probability equals ${}^{s_i!s_j!}/_{(s_i+s_j)!}$.

□

Theorem 4.2.2. *Let $G \in \mathcal{G}_3$. If the Markov chain corresponding to the switching model for G with respect to $\mathcal{G}_3$ is irreducible, then the Markov chain of the Simple Curveball algorithm converges to the uniform distribution on all simple directed networks with the same degree sequences as G.*

Proof. The state graph of the switching model with respect to $\mathcal{G}_3$ is a subgraph of the state graph of the Simple Curveball algorithm. Hence, irreducibility for this version of the switching model implies irreducibility for the Markov chain of the Simple Curveball algorithm. The Markov chain of the Simple Curveball algorithm is always aperiodic, since there is a non-zero probability of repeating each state, due to no-trade shuffles (see Section 4.1.2). Finally the simplified detailed balance equations hold due to an argument similar to that for the Curveball algorithm. □

The Markov chain of the Simple Curveball algorithm is not always irreducible. For instance, it is reducible for the directed 3-cycle (see Figure 2.5). For completeness, we discuss how this issue can be resolved. However, for most networks, the Markov chain of the Simple Curveball algorithm will be irreducible.

We may introduce a triangle reorientation move (see Definition 2.2.5). The Adjusted Simple Curveball algorithm attempts a trade with probability p_t (for a chosen fixed p_t) and a triangle reorientation move with probability $1 - p_t$. A triangle reorientation move proceeds as follows. (a) Randomly select a set A_i. (b) Randomly select an element j from A_i. (b) Randomly select an element k from A_j. (c) If $k \neq i$, $i \in A_k$, $i \notin A_j$, $j \notin A_k$ and $k \notin A_i$ then replace j by k in A_i, replace k by i in A_j and replace i by j in A_k.

Lemma 4.2.3. The transition probabilities P_{AB} of the Adjusted Simple Curveball algorithm are given by

$$P_{AB} = \begin{cases} p_t \frac{2}{n(n-1)} \frac{s_i! s_j!}{(s_i+s_j)!} & \text{if } A \text{ and } B \text{ differ in a trade between} \\ & \text{row } i \text{ and } j, \\ (1-p_t)\frac{1}{n}\left(\frac{1}{|A_i||A_j|} + \frac{1}{|A_j||A_k|} + \frac{1}{|A_j||A_i|}\right) & \text{if } A \text{ and } B \text{ differ in a triangle re-} \\ & \text{orientation involving rows } i, j, k, \\ 1 - \sum_{C \neq A} P_{AC} & \text{if } A = B, \\ 0 & \text{otherwise.} \end{cases}$$

Proof. The probability of transitioning from a state A to another state B that differ in a triangle reorientation from (i,j), (j,k), (k,i) to (i,k), (k,j), (j,i) can be found as follows. First notice that there are three different ways of selecting this triangle move, corresponding to selecting set A_i, A_j or A_k in step (a). In each case, the probability of attempting a triangle reorientation is equal to $1 - p_t$ and the probability of selecting the set equal $1/n$. Now if we selected set A_i, then the probability of selecting $j \in A_i$ equals $1/|A_i|$ and the probability of selecting $k \in A_j$ equals $1/|A_j|$. Similarly, for set A_j, the probability of selecting $k \in A_j$ and $i \in A_k$ equals $1/|A_j||A_k|$ and for set A_k, the probability of selecting $i \in A_k$ and $j \in A_i$ equals $1/|A_k||A_i|$. Hence the total probability of switching from A to B equals $(1-p_t)\frac{1}{n}\left(\frac{1}{|A_i||A_j|} + \frac{1}{|A_j||A_k|} + \frac{1}{|A_j||A_i|}\right)$. □

In particular the simplified detailed balance equations hold for the Adjusted Simple Curveball algorithm.

We ran experiments to verify that the Simple Curveball algorithm converges to its sta-

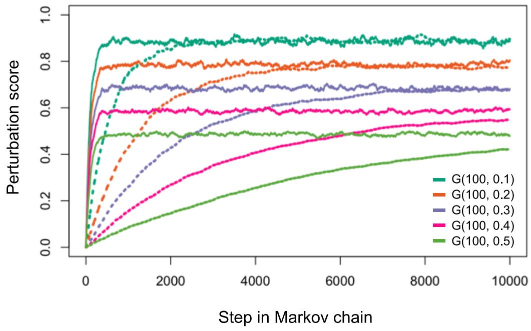
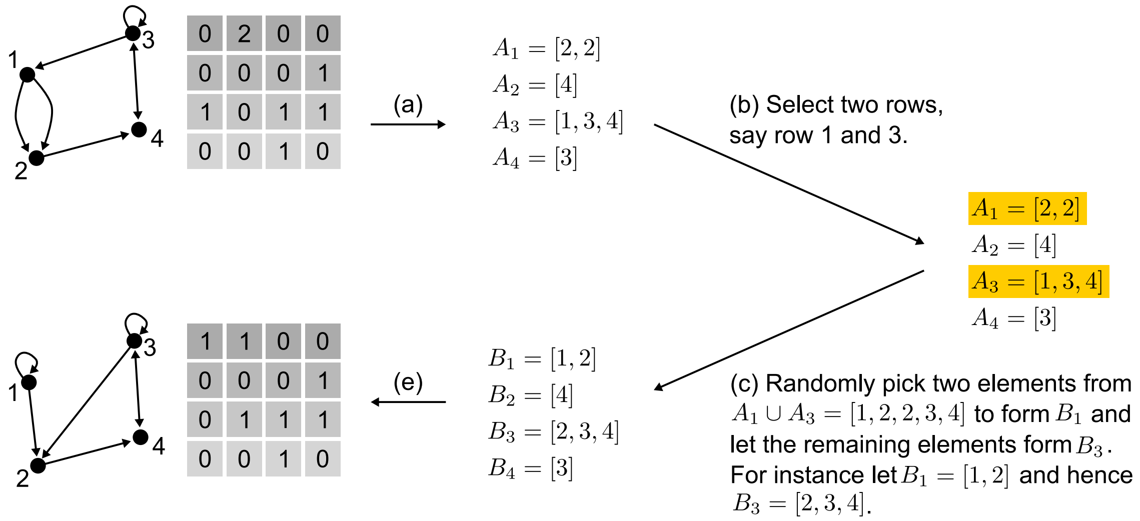


Figure 4.7.: The perturbation score at each step in the Simple Curveball algorithm (solid) and the switching model with respect to $\mathcal{G}_3$ (dashed) as compared to the initial network. We randomise five Erdős-Rényi random networks $G(n, p)$ with different edge densities p. The perturbation score stabilizes much earlier for the Simple Curveball algorithm than for the switching algorithm, indicating faster convergence to the stationary distribution.

tionary distribution faster than the switching model with respect to $\mathcal{G}_3$ does. We use the perturbation score as an estimate of the mixing time (see Section 3.1.2). Figure 4.7 shows that indeed it converges faster, especially for networks with high edge density. The Curveball algorithm furthermore has shorter step run-times than the switching model, making it faster still (see Section 4.4). We recommend using the Simple Curveball algorithm instead of the switching model for the randomization of simple directed networks, especially when randomizing large or dense networks.

Next we look at the Curveball algorithm for directed multigraphs, classes $\mathcal{G}_7$ and $\mathcal{G}_8$. The adjacency matrix of a multigraph is no longer binary, it may contain other positive integers besides 0 and 1. The main difference from the Curveball algorithm is that we no longer have *sets* of indices, A_i, instead we obtain *multi-sets*, where indices may occur more than once. We will use square brackets, [ ], to denote a multiset. We write $|[\,]|$ for the size of a multiset *taking into account the multiplicity of elements*. Similarly to the switching model with respect to $\mathcal{G}_8$, the Curveball algorithm simplifies for the class of multigraphs.

The Curveball algorithm with respect to $\mathcal{G}_8$ randomizes a matrix A using the following steps: (a) transform A into multisets of indices, A_i, for each row i, corresponding to non-zeroes in that row, (b) select two of these multisets A_i and A_j at random, (c) create a new multiset B_i by randomly selecting $|A_i|$ indices from $A_i \cup A_j$, and let B_j be the multiset of remaining indices. (d) Reiterate steps (b) and (c) N times, for a certain fixed number N, and (e) form a new matrix from the resulting multisets. Figure 4.8 illustrates this algorithm.



(b) Select two rows, say row 1 and 3.

(c) Randomly pick two elements from $A_1 \cup A_3 = [1,2,2,3,4]$ to form B_1 and let the remaining elements form B_3. For instance let $B_1 = [1,2]$ and hence $B_3 = [2,3,4]$.

Figure 4.8.: A *trade* in the Curveball algorithm with respect to $\mathcal{G}_8$ consists of (a) converting the adjacency matrix A of a directed multigraph into multisets of indices, A_i, for each row i. In step (b) two rows are selected, in this case row 1 and 3. (c) Let B_i be a multiset of $|A_i|$ randomly selected elements from the multiset $A_i \cup A_j$. Let B_j be the multiset of remaining elements from $A_i \cup A_j$. Steps (b) and (c) are repeated N times, before using step (e) to convert the resulting multisets of indices B_i to the matrix B.

We may similarly define the Curveball algorithm with respect to directed multigraphs without self-loops ($\mathcal{G}_7$). The steps to randomize a matrix A with zeros on its diagonal are: (a) transform A into multisets of indices, A_i, for each row i, corresponding to non-zeroes in that row. Notice that $i \notin A_i$. (b) Select two of these multisets A_i and A_j at random. (c) Let A_{i-j} be the multiset A_i with all occurrences of j removed. Similarly define A_{j-i}. (d) Create a new multiset B_i by removing A_{i-j} from A_i and adding $|A_{i-j}|$ randomly selected elements from $A_{i-j} \cup A_{j-i}$. Let B_j be the multiset formed by removing A_{j-i} from A_j and adding the remaining indices of $A_{i-j} \cup A_{j-i}$. (e) Reiterate steps (b)-(d) N times, for a certain fixed number N, and (f) form a new matrix from the resulting multisets.

We will not derive the stationary distributions for the Curveball algorithms with respect to classes $\mathcal{G}_7$ and $\mathcal{G}_8$. However, we give an outline of how this could be done. To check irreducibility of the Markov chain, we can use the irreducibility of the Markov chain of the switching model. Every switch in the switching model with respect to classes $\mathcal{G}_7$ and $\mathcal{G}_8$ can again be shown to correspond to a trade of size one in the corresponding Curveball algorithm. Aperiodicity of the Markov chain follows from each state having a non-zero probability of being repeated (a no-trade shuffle). Finally the transition probabilities will need to be derived to find the stationary distribution. Notice that the simplified detailed balance equations no longer hold. For instance, in Figure 4.8 the trade from A to B is more likely than the trade from B to A since there are multiple ways of selecting the trade (due to 2 occurring with multiplicity 2 in A_1).

4.3. Curveball algorithms for undirected networks

With a little more effort, the Curveball algorithm can also be used to randomize simple undirected networks ($\mathcal{G}_1$). We define the Simple Undirected Curveball algorithm as follows.

Let A be the (symmetric) adjacency matrix of a simple undirected network G. (a) Transform A into sets of indices, A_i for each row i, corresponding to the ones in that row. (b) Select two of these sets A_i and A_j at random. (c) Compare these sets and let A_{i-j} be all indices that are in A_i but that are not in A_j and that are not equal to j. Similarly define A_{j-i}. (d) Create a new set B_i by removing A_{i-j} from A_i and adding the same number of elements randomly chosen from $A_{i-j} \cup A_{j-i}$. Combine $A_j \backslash A_{j-i}$ with the remaining elements of $A_{i-j} \cup A_{j-i}$ to form B_j. (d′) For each index $k \in B_i \backslash A_i$, replace j by i in B_k, similarly for each $l \in B_j \backslash A_j$, replace i by j in B_l. (e) Reiterate step (b)-(d) N times, for a certain fixed number N, and (f) form a new matrix from the resulting sets.

Notice that step (d′) is well-defined: $k \in B_i \backslash A_i$ implies $k \notin A_i$ and $k \in A_j$. Furthermore, since A is symmetric, this also implies that $i \notin A_k$ and $j \in A_k$, thus we can replace j by i in A_k to obtain B_k. Similarly $l \in B_j \backslash A_j$ implies that i is an element of A_l and that

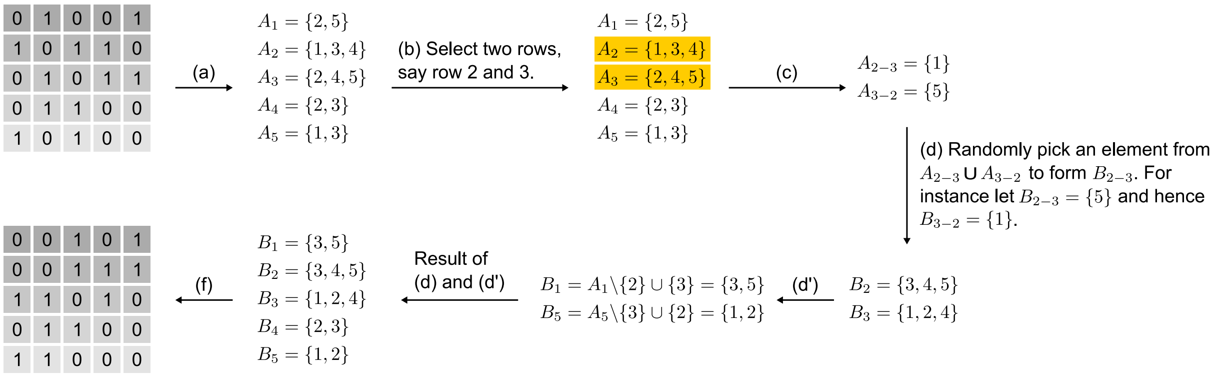


Figure 4.9.: A *trade* in the Simple Undirected Curveball algorithm consists of (a) converting a *symmetric* adjacency matrix of an undirected network into a sets of indices, A_i, for each row i. In step (b) two rows are selected, in this case row 2 and 3. In step (c) the sets $A_{2-3} = \{1,3,4\}\backslash(\{2,4,5\}\cup\{3\}) = \{1\}$ and $A_{3-2} = \{2,4,5\}\backslash(\{1,3,4\}\cup\{2\}) = \{5\}$ are extracted. (d) The set B_2 is formed by removing A_{2-3} from A_2 and adding $|A_{2-3}|$ elements randomly selected from $A_{2-3} \cup A_{3-2}$. Similarly B_3 is formed by removing A_{3-2} from A_3 and adding the remaining elements of $A_{2-3}\cup A_{3-2}$. In this case we obtain $B_2 = \{3,4,5\}$ and $B_3 = \{1,2,4\}$. (d$'$) To ensure the matrix remains symmetric, we need to update the rows corresponding to the indices involved in the trade. In this case indices 1 and 5 were traded. We update row 1 by replacing index 2 by 3 and we update row 5 by replacing the index 3 by 2. We now obtain sets of indices corresponding to another symmetric matrix. In step (e) of the algorithm the steps (b),(c),(d) and(d$'$) are repeated. Step (f) builds a symmetric matrix from the resulting sets of indices.

j is not. And thus, replacing i by j in A_l is well-defined. Step (d$'$) ensures that B is a symmetric matrix.

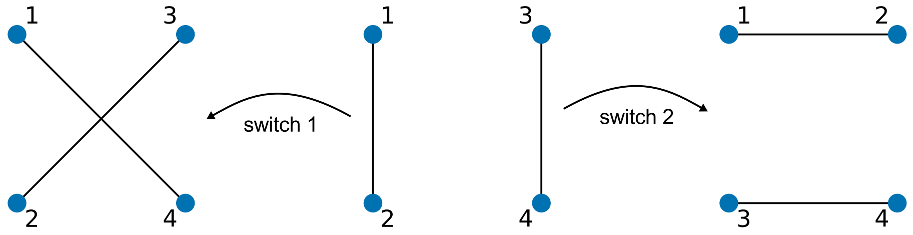
Also notice that, similarly to the Simple Curveball algorithm, the Simple Undirected Curveball algorithm does not introduce any self-loops. This is achieved by explicitly removing the indices that would introduce self-loops from trades. That is, if present, j is removed from A_i and i from A_j. Figure 4.9 illustrates the Simple Undirected Curveball algorithm.

We now show that the state graph of the switching model with respect to $\mathcal{G}_1$ is a subgraph of the state graph of the Simple Undirected Curveball algorithm.

Lemma 4.3.1. Let $G, G' \in \mathcal{G}_1$ such that G and G' differ by a switch. There is a trade of size one in the Simple Undirected Curveball algorithm from G to G'.

Proof. Without loss of generality we may assume that G and G' differ by a switch from $\{x, y\}$ and $\{u, v\}$ to $\{x, v\}$ and $\{u, y\}$. Let A be the adjacency matrix of G, then $y \in A_{x-u}$ since the edge $\{x, y\}$ is an edge of G, the edge $\{u, y\}$ is not and y can not be equal to u since $\{u, y\} \in E'$ and G' is a simple network. Similarly we find that $v \in A_{u-x}$ and hence the trade that swaps y and v between rows x and u results in the network G'. □

For an illustration, see Figure 4.10.



$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \qquad \underset{A}{\begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}} \qquad \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

Figure 4.10.: Both switch 1 and switch 2 can be realized by the Simple Undirected Curveball algorithm. Switch 1 is realised by either selecting row 1 and 3, or row 2 and 4 in matrix A (if the shuffling results in the unique trade of size 1). Similarly switch 2 is realised by either selecting row 1 and 4, or row 2 and 3.

In fact, Figure 4.10 shows that for each switch in the switching model, there are two different trades of size 1 in the Simple Undirected Curveball algorithm.

Lemma 4.3.2. Let A and B be two symmetric $n \times n$ binary matrices with zero diagonal and equal row and column sums. The transition probability P_{AB} from A to B, in the

Simple Undirected Curveball algorithm, is given by

$$P_{AB} = \begin{cases} \frac{2}{n(n-1)}\left(\frac{s_i!s_j!}{(s_i+s_j)!} + \frac{s_k!s_l!}{(s_k+s_l)!}\right) & \text{if } A \text{ and } B \text{ differ by a trade of size one, between rows} \\ & i \text{ and } j, \text{ involving columns } k \text{ and } l, \\ \frac{2}{n(n-1)}\frac{s_i!s_j!}{(s_i+s_j)!} & \text{if } A \text{ and } B \text{ differ in a trade of size more than one} \\ & \text{between row } i \text{ and row } j, \\ 1 - \sum_{C, C \neq A} P_{AC} & \text{if } A = B, \\ 0 & \text{otherwise.} \end{cases}$$

with $s_i = |A_{i-j}|$, $s_j = |A_{j-i}|$, $s_k = |A_{k-l}|$ and $s_l = |A_{l-k}|$.

Proof. When matrices A and B differ by a trade of size 1 between rows i and j and involving columns k and l, then they also differ by a trade of size 1 between rows k and l involving columns i and j. Hence, we need to add the probabilities of selecting either one of these trades. When A and B differ in trade of size larger than one, there is just one unique row-pair that corresponds to this trade, hence we find the usual transition probability. □

Theorem 4.3.3. *Let $G \in \mathcal{G}_1$. If the Markov chain corresponding to the switching model for G with respect to $\mathcal{G}_1$ is irreducible, then the Markov chain of the Simple Undirected Curveball algorithm converges to the uniform distribution on all simple directed networks with the same degree sequences as G.*

Proof. The state graph of the switching model with respect to $\mathcal{G}_1$ is a subgraph of the state graph of the Simple Undirected Curveball algorithm. Hence, irreducibility for this version of the switching model implies irreducibility for the Markov chain of the Simple Undirected Curveball algorithm. The Simple Undirected Curveball algorithm is always aperiodic, since there is a non-zero probability of repeating each state, due to no-trade shuffles (see Section 4.1.2). Finally the simplified detailed balance equations hold due to an argument similar to that for the Curveball algorithm. □

We ran experiments to verify that the Simple Undirected Curveball algorithm converges to its stationary distribution faster than the switching model does. Figure 4.11 indicates

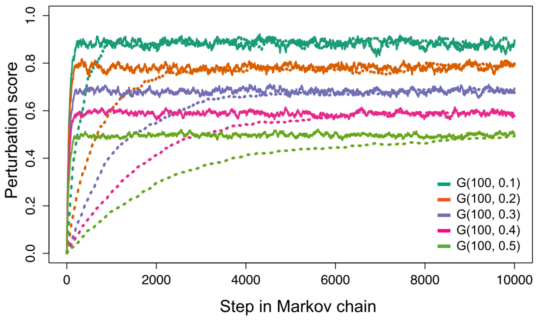


Figure 4.11.: The perturbation score at each step in the Simple Undirected Curveball algorithm (solid) and the switching model with respect to $\mathcal{G}_1$ (dashed) as compared to an initial network. We randomise five Erdős-Rényi random networks $G(n, p)$ with different p. The Simple Undirected Curveball algorithm converges faster than the switching model with respect to $\mathcal{G}_1$.

that it does indeed, in particular for networks with high edge density. We recommend using the Simple Undirected Curveball algorithm instead of the switching model for the randomization of simple undirected networks, especially when randomizing large or dense networks.

Finally, we introduce the Curveball algorithm with respect to $\mathcal{G}_5$, multigraphs without self-loops. Similarly as for directed multigraphs, sets of indices are replaced by multisets. This algorithm randomizes a symmetric matrix with positive integers and zeroes on its diagonal using the following steps. (a) Transform a matrix into multisets of indices A_i for each row. Notice that $i \notin A_i$. (b) Select two multisets A_i and A_j at random. (c) Form the multisets A_{i-j} and A_{j-i} by removing all occurrences of j from A_i and all occurrences of i from A_j respectively. (d) Form a new multiset B_i by removing A_{i-j} from A_i and adding $|A_{i-j}|$ randomly selected elements from $A_{i-j} \cup A_{j-i}$ and form B_j by removing A_{j-i} from A_j and adding the remaining elements from $A_{i-j} \cup A_{j-i}$. (e) Repeat steps (b)-(d) a certain number of times and (f) form a new symmetric matrix from the resulting multisets.

We do not derive the stationary distribution for the Curveball algorithm with respect to $\mathcal{G}_5$. As for the Curveball algorithm with respect to directed multi-graphs, the simplified detailed balance equations will no longer hold.

4.4. The run-time of the Curveball algorithm

So far, we have only discussed the speed of the Curveball algorithms in terms of their mixing time. The original Curveball algorithm mixes faster than the switching model [115, 106]. Similarly, the Simple Curveball algorithm (see Section 4.2) and the Simple Undirected Curveball algorithm (see Section 4.3) mix faster than the switching model with respect to classes $\mathcal{G}_3$ and $\mathcal{G}_1$ respectively. However, as observed in Section 3.1.2, in order to compare the *run-time* of Markov chains, we also need to take into account the *step run-time* of the algorithms. In this section we show that the step run-time of the Curveball algorithms is significantly shorter than that of the switching models. These two properties of Curveball algorithms, the short step-run time and the fast mixing time, result in algorithms which run dramatically faster than switching models.

Here, we compare the step run-time of the original Curveball and Simple Curveball algorithm to the step run-time of the switching model with respect to $\mathcal{G}_4$ and $\mathcal{G}_3$ respectively. To do so, we randomise three different real-world directed networks: the neural network of C. Elegans (see Appendix A.2), the Enron email network (see Appendix A.5) and a network of air traffic between airports in the US (see Appendix A.10). If present, we remove multiple edges and self-loops before running the Markov chains, so that we start with a simple network. For each Markov chain and each network, we measured the run-time of taking 1000 steps, that is, for attempting either 1000 trades or 1000 switches. Table 4.2 shows the results of this experiment.

The step run-time of the Curveball algorithm is significantly shorter than that of the switching model for the three directed networks. We are not certain what causes this difference, but expect it has to do with the fact that the Curveball algorithm works with adjacency sets, whereas the switching model uses the adjacency matrix. Adjacency sets are known to be an efficient storage structure for networks [94, Section 9.4].

Even though experimentally it is clear that the mixing time of the Curveball algorithms is much faster than that of the switching model, there is no theoretical proof of this.

Networks			Step-runtimes $t_s(1000)$ in seconds			
Name	n	m	Curveball $\mathcal{G}_3$	Curveball $\mathcal{G}_4$	Switching $\mathcal{G}_3$	Switching $\mathcal{G}_4$
C. Elegans	297	2345	0.569	1.045	9.488	9.612
Enron	22477	53285	0.540	0.528	116.576	117.427
US airports	755	8228	1.453	1.038	281.818	284.814

Table 4.2.: The run-time in seconds for executing 1000 steps with the Curveball algorithm (see Appendix B.1.1 and switching model (see Appendix B.1.6) with respect to classes $\mathcal{G}_3$ and $\mathcal{G}_4$ for three different simple directed networks. The Curveball algorithm is always by far faster than the switching model.

There are few theoretical results available for rapid mixing of the switching model. Recall that a Markov chain is called rapid mixing if its mixing time is bounded by a polynomial expression (Definition 1.3.10). For the special case of regular and semi-regular networks [41, 65, 46], the polynomial upper bound for the mixing time was found using a multi-commodity flow argument [60, 111]. These proofs rely on defining a special class of paths between all states in the state graph. Paths are chosen in such a way that the load on each edge (the number of paths it takes part in) is relatively small. The mixing time can then be bounded from above in terms of a product of these edge loads and the inverse of their transition probabilities.

Unfortunately the multi-commodity flow method can not be used to prove rapid mixing for the Curveball algorithm. The same class of paths could be used, but the argument breaks down when estimating the transition probabilities. The reason for this is that the transition probabilities in the Curveball algorithm can be exponentially small with respect to the number of vertices n in a network, leading to an exponential factor in the upper bound.

We do not believe that the small transition probabilities are an *actual* obstruction to fast mixing of the Curveball algorithms, since each state also has a corresponding exponential number of neighbouring states. The fastest mixing Markov chain on N states has the complete graph as its state graph, with all transition probabilities equal to $1/N$. With an exponentially large state space, these probabilities are also exponentially small. Intuitively, the Curveball algorithm is much closer to this optimal situation than the switching method.

It appears that an altogether different method is needed to find a theoretical upper bound for the mixing time of the Curveball algorithm. This is a difficult, but important open problem. The Curveball algorithm seems to be a step in the right direction for the fast generation of random directed networks. In Appendix C we illustrate several state graphs of the Curveball algorithm for regular graphs of small size. We hope that the symmetry of these state graphs may be exploited to estimate mixing times. This is left as future work.

4.5. Conclusion

We started this chapter by discussing the original Curveball algorithm and proving that it converges to the uniform distribution. We recommend the use of the Curveball algorithm for the randomization of directed networks ($\mathcal{G}_4$), as it is currently the fastest method to produce unbiased samples. Furthermore, it is best to leave the Curveball algorithm as it is, with inclusion of all repeated states.

We also introduced several different versions of the Curveball algorithm, thereby showing that the idea of the Curveball algorithm provides us with a flexible framework for network randomization. The most important of these algorithms are the Simple and Simple Undirected Curveball algorithm, since both classes of networks are common to find in applications. We show that the Simple (Undirected) Curveball algorithm converges to the uniform distribution and that it does so faster than the switching model. We recommend the use of these models over that of respective versions of the switching model. For completeness we also introduced the Curveball algorithm for several multigraph classes, that is for classes $\mathcal{G}_5, \mathcal{G}_7$ and $\mathcal{G}_8$.

We have not yet extended the Curveball algorithm to undirected networks with self-loops. We do believe it is possible to define such an extension, however preliminary study shows it needs to be a little more subtle than the extensions discussed in this chapter. This is left as future work.

It may be worthwhile to combine the Expand and Contract method (see Section 3.2) with the Curveball method, to ensure that the generated random networks are in no way related to the original network. To combine these methods we need versions of the Curveball algorithm with respect with respect to $\mathcal{G}_6$ for undirected networks and with respect to $\mathcal{G}_8$ for directed networks. We introduced the Curveball method with respect to $\mathcal{G}_8$. However the Curveball algorithm with respect to $\mathcal{G}_6$ is left as future work.

Finally we pointed out why current techniques can not be used for formal proof of rapid mixing of the Curveball algorithm. Developing new techniques and proving rapid mixing is an interesting open problem. We are excited about the prospect of using the symmetry displayed by (parts of) the state graph of the Curveball algorithm (see Appendix C) to attempt to prove rapid mixing.

5. Local network properties

In this chapter we discuss network statistics based on local network properties. By local we mean small subgraphs of the network. Specifically we look at network motifs and the neighbourhoods of nodes. This chapter contains mostly experimental results.

Network motifs [82] break complex networks into small building blocks and are a popular tool for the analysis of local network properties [128, 32, 127, 4, 3]. Here we investigate the use of this technique for a specific class of networks: directed acyclic networks. This is an important class of networks that appears in many applications. The best known example is that of citation networks. Other examples include patent networks, networks of dependencies in software as well as dependencies of lemmas, axioms and theorems in pure mathematics, and even biological networks such as predator-prey networks [67]. Section 5.1 discusses our work on motifs in directed acyclic graphs.

Intuitively, it is clear that the neighbourhood of a node can provide us with valuable information about the node itself. For instance, we can learn a lot about someone from knowing her friends and we can roughly derive the topic of a paper from its references. Many interesting applications of neighbourhood analysis can be found in the literature. For instance, structural features can be used to classify different roles in networks [52]. In [10] the structure of the neighbourhood of Facebook users, i.e. the network of their Facebook friends, is analysed to find their partner as well as to predict whether or not the relationship will be lasting. And in [32], the quality of a Wikipedia article is successfully predicted by analysing the motifs in the network of edits made to the article (the edit neighbourhood).

Here we discuss a different application of neighbourhood analysis: the identification of nodes for discovery. An example of a real scenario where discovery is important and the neighbourhood may help, is when a person of interest to authorities uses an unidentified mobile phone in order to remain untraceable, but still makes calls to other phones that are identified. Section 5.2 discusses our experimental work on the topic of neighbourhood distinctiveness and matching neighbourhoods.

My contributions to the literature in this chapter are the following:

- Derivation of a formula which relates the number of occurrences of the four distinct 3-node patterns in directed acyclic networks. This result is published in [24].
- Motif finding experiments in citation networks. These results are published in [24] and [25].
- Experimental work on the distinctiveness of neighbourhoods. This work was done in collaboration with the Information Security and Network Science Research Group at RMIT University. The results are published in [59] and [51].

5.1. Motifs in directed acyclic networks

Motifs were first introduced in [82] and are defined as “recurring **significant** patterns of interconnections in a network”. Motifs were shown to define *classes* of networks, and it was argued that these classes relate to the functioning of a network. Motifs have since been a popular method of analysing local network structure [128, 32, 127, 4, 3] and the idea has been extended in several ways [80, 131]. However, prior to the introduction of motifs, similar local network analysis techniques had been used in social network analysis for a long time [53].

Before continuing our investigation of motifs, we introduce a mathematical definition of motifs. We first formalise the notion of a ‘pattern of interconnections’.

Definition 5.1.1. Let $G \in \mathcal{G}$ be a network in class $\mathcal{G}$, and let $\{m_i^k\}$ be the set of isomorphism classes of connected networks in $\mathcal{G}$ on exactly k nodes. We refer to these isomorphism classes as **k-node patterns (with respect to $\mathcal{G}$)**.

Definition 5.1.2. Let $G = (V, E)$ be a network in class $\mathcal{G}$, and let m_i^k be a k-node pattern with respect to $\mathcal{G}$. The **number of occurrences, c_i^k, of m_i^k in** G equals the number of k-node sets in V for which the induced subgraph in G is in the isomorphism class m_i^k.

We will use the notation m_i instead of m_i^k and c_i instead of c_i^k, when either the number of nodes is irrelevant, or when the number of nodes is clear from the context.

In the literature, varying criteria are used to determine if a subgraph is a motif [82, 80, 127]. We combine criteria to introduce the following stricter definition of a motif.

Definition 5.1.3. Let $\mathcal{G}$ be a class of networks and let $G \in \mathcal{G}$. Choose a random network model M^{rnd} for $\mathcal{G}$. Let m_i be a pattern with respect to $\mathcal{G}$ and let c_i be the number of occurrences of m_i in G and $\bar{c}_i^{rnd}$ the average number of occurrences in a sample of random networks generated by M^{rnd}. The pattern m_i is a **motif** of the network G if

1. the probability that it appears in a random network generated by M^{rnd} an equal or greater number of times than in G is less than 0.01 [82], and

2. its Z score is higher than 2 [127], and

3. the number of times it appears in G is significantly higher than in the random networks, that is $c_i - \bar{c}_i^{rnd} > 0.1\bar{c}_i^{rnd}$ [82].

In [82] a subgraph needs to occur at least 4 times to be considered as a network motif. In the empirical networks that we analysed, subgraphs occur much more often than this, so we do not use this criteria.

The criteria that are used to classify anti-motifs are not discussed to a great extent in the literature. In [80] an anti-motif is defined as “a significantly under-represented subgraph". We introduce a definition of an anti-motif here, analogous to the criteria above for a motif.

Definition 5.1.4. Let $\mathcal{G}$ be a class of networks and let $G \in \mathcal{G}$. Choose a random network model M^{rnd} for $\mathcal{G}$. Let m_i be a pattern with respect to $\mathcal{G}$ and let c_i be the number of occurrences of m_i in G and $\bar{c}_i^{rnd}$ the average number of occurrences in a sample of random networks generated by M^{rnd}. The pattern m_i is an **anti-motif** of G if

1. the probability of it appearing in a random network generated by M^{rnd} an equal or smaller number of times than in G is less than 0.01, and

2. its Z score is lower than -2, and

3. the number of times it appears in G is significantly lower than in the random networks, that is $\bar{c_i}^{rnd} - c_i > 0.1\bar{c_i}^{rnd}$.

It is clear from Definition 5.1.3 that the choice of a null-model influences the patterns that will or will not be considered as significant, and be keyed as a network motif. The standard choice of null-model is the *switching* model [82, 81, 78, 121] that we discussed in Chapter 2.

There has been some criticism of [82]. For instance, in [5] it was argued that the patterns that were detected as motifs in [82] could be explained by a preference for local connections instead of network functioning. However, this criticism does not indicate a problem with the motif analysis *technique*, but with the interpretation of results obtained from it.

The dependence on a choice of null-model has been another debated issue [5, 131]. Depending on the random network model, certain network properties will be present in the random networks but other network properties will not be present. This may lead to certain patterns being judged significant, that in fact could be explained by a different null-model. This issue can be avoided by carefully selecting a null model and by being aware of its limitations.

The rest of this section is organised as follows. Subsection 5.1.1 discusses properties specific to motifs in directed acyclic networks. Subsection 5.1.2 discusses the dependence of motifs on null-models and introduces the different null-models used in our experiments. Finally, Subsection 5.1.3 presents experimental results on motifs in citation networks.

5.1.1. Three-node motifs in directed acyclic networks

The motifs discussed in [82] are motifs with respect to simple networks ($\mathcal{G}_1$ and $\mathcal{G}_3$). We are interested in motifs in directed acyclic networks, a subset of the networks in $\mathcal{G}_3$. The main difference between motif detection in directed acyclic networks and motifs in simple directed networks, is that there are fewer k-node patterns with respect to directed acyclic networks. Due to computational constraints, motif analysis usually focusses on 3-node and 4-node patterns. Figure 5.1 depicts all distinct 3-node and 4-node patterns that can appear as subgraphs of directed acyclic networks.

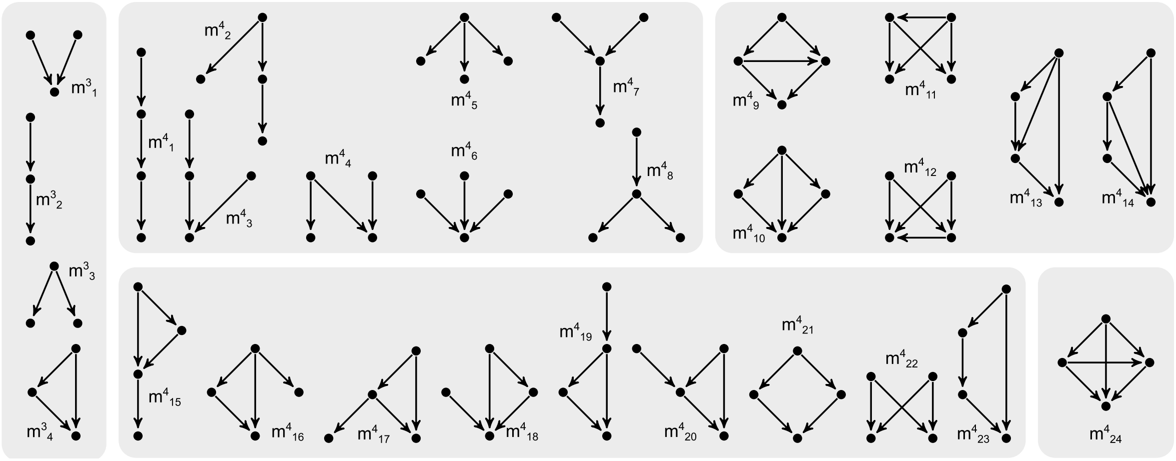


Figure 5.1.: All three node and four node patterns that may be found as subgraphs of directed acyclic networks. There are four three node patterns, labelled m^3_1, m^3_2, m^3_3 and m^3_4. There are 24 four node patterns, $m^4_1, ..., m^4_{24}$.

We restrict our attention to the four three node patterns $m^3_1, m^3_2, m^3_3, m^3_4$. From now on we will use the notation m_1, m_2, m_3 and m_4 for these four patterns. Pattern m_4 is called a *feedforward loop*. Notice that patterns are counted as *induced* subgraphs (see Definition

5.1.2). In particular, the occurrence of the feedforward pattern does not contribute to the count of patterns m_1, m_2 and m_3: these are subgraphs of m_4 but not *induced* subgraphs. We now derive a formula relating the number of occurrences of three node patterns to the degree sequences of a directed acyclic graph.

Lemma 5.1.5. Let c_1, c_2, c_3 and c_4 be the number of times pattern m_1, m_2, m_3 and m_4 occur in a simple directed acyclic graph G. Let k^{in} and k^{out} denote the in-degree and out-degree sequence of G respectively. The following formulas hold

$$\begin{aligned} c_1 &= \sum_{i=1}^{n} \frac{k_i^{in}(k_i^{in}-1)}{2} - c_4 \\ c_2 &= \sum_{i=1}^{n} k_i^{in} k_i^{out} - c_4 \\ c_3 &= \sum_{i=1}^{n} \frac{k_i^{out}(k_i^{out}-1)}{2} - c_4. \end{aligned} \tag{5.1}$$

Proof. We first show that these formulas hold when G does not contain any feedforward loops, that is when $c_4 = 0$. In this case, the formulas are the result of a simple count. For a node v with in-degree $k^{in}(v)$ we find $k^{in}(v)(k^{in}(v)-1)/2$ pairs of incoming edges (see Figure 5.2), resulting in the formula for c_1. Similarly we obtain the formula for c_3. For a node v with both incoming and outgoing edges we obtain $k^{in}(v)k^{out}(v)$ combinations of an incoming and outgoing edge, resulting in the formula for c_2 (see Figure 5.2).

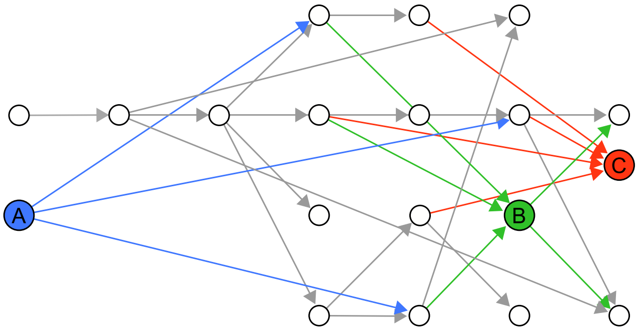


Figure 5.2.: Node A contributes $k_A^{out}(k_A^{out}-1)/2 = 3$ to c_3. Node C contributes $k_C^{in}(k_C^{in}-1)/2 = 6$ to c_1. Node B contributes $k_B^{out}(k_B^{out}-1)/2 = 1$ to c_3, $k_B^{in}(k_B^{in}-1)/2 = 3$ to c_2 and $k_B^{in}k_B^{out} = 6$ to c_3.

Now let G be a graph that *does* contain feedforward loops. We need to correct our count for each feedforward loop; since a feedforward loop contains the three other motifs as a sub-motif, we need to subtract c_4 from each of the other counts. □

Corollary 5.1.6. Let G and G' be simple directed acyclic networks with equal degree sequences. There is an integer $z \in \mathbb{Z}$ such that the following relations hold

$$\begin{aligned} c_1 = &\quad c_1' + z \\ c_2 = &\quad c_2' + z \\ c_3 = &\quad c_3' + z \\ c_4 = &\quad c_4' - z. \end{aligned} \tag{5.2}$$

where c_i' is the number of occurrences of pattern m_i in G'.

Proof. Let $z = c_4' - c_4$, then by Lemma 5.1.5

$$c_1 - c_1' = (\sum_{i=1}^{n} \frac{k_i^{in}(k_i^{in} - 1)}{2} - c_4) - (\sum_{i=1}^{n} \frac{k_i^{in}(k_i^{in} - 1)}{2} - c_4') = c_4' - c_4 = z,$$

similarly we may derive $c_2 - c_2' = z$ and $c_3 - c_3' = z$. □

We defined the ordered switching model (Definition 2.4.1) to randomise directed acyclic networks. That is, the ordered switching model generates random directed *acyclic* networks with *fixed* degree sequences. Combining this property of the ordered switching model with Corollary 5.1.6 we obtain the following result.

Corollary 5.1.7. Let G be a simple directed acyclic network and let $\{G^{rnd}\}$ be a collection of networks randomised by the ordered switching model for G. Let c_i be the number of occurrences of pattern m_i in G. Let $\bar{c}_i^{rnd}$ be the average number of occurrences of pattern m_i in $\{G^{rnd}\}$ and let σ_i be the standard deviation. Then the following relations hold

(i) $c_1 - \bar{c}_1^{rnd} = c_2 - \bar{c}_2^{rnd} = c_3 - \bar{c}_3^{rnd} = \bar{c}_4^{rnd} - c_4$, and

(ii) $\sigma_1 = \sigma_2 = \sigma_3 = \sigma_4$.

□

5.1.2. Null-models for motifs in directed acyclic networks

It is clear from Definition 5.1.3 that the choice of a null-model has an impact on which patterns are labelled as motifs. As we are interested in motifs in directed acyclic networks, it is natural to use a random network model for directed acyclic networks. We investigate the impact of using three different null-models: the switching method with respect to simple directed networks (see Definition 2.1.3), the ordered switching method (see Definition 2.4.1) and the null-model introduced in [67]. For simplicity, we will refer to the former as the switching model and the latter as the Karrer-Newman model in the rest of this section.

The switching model treats citation networks as directed networks. In other words, it does not respect the topological ordering of the nodes. Edges (x, y) and (u, v) are swapped even if $u < y$ or $x < v$. It is possible that a different topological ordering of the resulting graph exists, but it is also possible that directed cycles are introduced.

The Karrer-Newman model does maintain the topological order of directed acyclic networks. This model is a variation on the configuration model (see Section 3.2). It constructs a directed acyclic network with the same in-degree and out-degree sequence as the original network. However, it may construct a network that contains multiple edges. In fact, in our experiments this method never produced a network without multiple edges and often the number of multiple edges was high.

Both the switching model and the Karrer-Newman model maintain the degree sequences of a directed acyclic graph and randomize the edges. However these algorithms create networks that are inherently different from the original network. The Karrer-Newman algorithm allows multiple edges between nodes, whereas the switching algorithm allows edges that do not respect the topological ordering of the network and can create directed cycles.

This motivated us to develop the ordered switching model (see Definition 2.4.1) The ordered switching model overcomes these issues, randomizing a directed acyclic network while fixing its degree sequences and topological ordering.

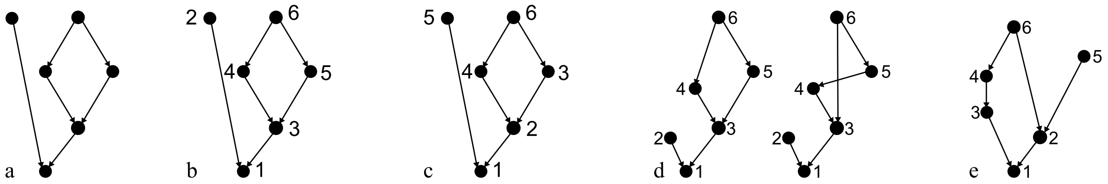


Figure 5.3.: (a) A directed acyclic graph without topological ordering. (b) A topological ordering of the graph with $k^{in} = (2, 0, 2, 1, 1, 0)$ and $k^{out} = (0, 1, 1, 1, 1, 2)$. (c) A different topological ordering with $k^{in} = (2, 2, 1, 1, 0, 0)$ and $k^{out} = (0, 1, 1, 1, 1, 2)$. (d) There are exactly two directed acyclic graph realisations of the degree sequences in (b). (e) There are 14 directed acyclic graph realisations of the degree sequences in (c), one of which is depicted here.

Both the Karrer-Newman algorithm and the ordered switching method require a topological ordering of the nodes as input. Often the problem at hand will specify this ordering, e.g. the publication dates in case of citation network. Even if the topological ordering is unknown, there are efficient algorithms to find an ordering [64, 1]. It is important to realize that the chosen ordering influences the possible outcomes of both the Karrer-Newman algorithm and the switching method. This is illustrated in Figure 5.3.

Model	**acyclic**	**simple**	**uniform**	**speed**
Karrer-Newman	yes	no	quasi [94, Section 13.2]	fast*
Switching	no	yes	yes (See Table 2.2)	slow*
Ordered switching	yes	yes	yes (Theorem 2.4.4)	slow*

Table 5.1.: Comparison of the three random network models discussed in this Chapter. * See the discussion in Section 3.2.

Table 5.1 summarizes the properties of these three network randomisation methods.

5.1.3. Motif experiments in citation networks

We ran motif finding experiments on two directed acyclic citation networks: hep-th (see Appendix A.7) and hep-ph (see Appendix A.8). Vertices correspond to papers and edges point from citing paper to cited paper. For each paper, we have a corresponding publication date. Edges only point 'backwards in time', i.e. the citing paper is always published more recently than the cited paper.

Unfortunately, the publication dates associated with papers only provide a starting point for a topological ordering: the majority of papers share their publication date with at least one other paper. Any permutation of papers with the same publication date results in a valid topological ordering. For instance, if the initial ordering is $((v_1, t_1)$, (v_2, t_1), (v_3, t_1), (v_4, t_2), (v_5, t_2), (v_6, t_3), $(v_7, t_3))$ then $V = (v_2,\ v_3,\ v_1,\ v_4,\ v_5,\ v_7,\ v_6)$ is another valid topological ordering. In this case there are 3! orderings for the first three nodes and 2! orderings for both pairs (v_4, v_5) and (v_6, v_7). In total there are 24 different topological orderings for this small set of nodes. Both the citation networks that we analysed contain many repeating publication dates resulting in a huge number of topological orderings. There are over 10^{21659} distinct topological orderings for the theoretical physics network and over 10^{21622} for the phenomenological physics network.

The chosen topological ordering influences the possible outcomes of the Karrer-Newman and ordered switching algorithm (see Figure 5.3). Therefore, to avoid sampling from one graph ensemble alone, a topological ordering was chosen uniformly at random before each run of both algorithms. This was done by selecting a permutation for each sequence of repeating dates. The random network ensembles consist of 1000 networks each, so only a small number of the possible topological orderings are sampled. At this stage we do not know exactly how this affects the results. However, we expect that the impact of these different topological orderings is small, since the changes are restricted to reshuffling the order of papers published on the same date. In the theoretical network the biggest set of papers with the same publication date consists of 43 papers and in the phenomenological network of 44 papers.

We used the following software in our experiments: MFinder1.2 (see Appendix B.2.5) to generate random networks with the switching model, our own implementation of the Karrer-Newman algorithm (see Appendix B.1.5), a modified version of MFinder1.2 (see Appendix B.1.5) to generate random networks with the ordered switching model, and the R library igraph (see Appendix B.2.3) to count three node patterns.

Table 5.2 shows the number of occurrences of directed acyclic three node patterns in the theoretical citation network as well as the average pattern counts in the three random graph ensembles. The average number of occurrences of all patterns is higher for the ordered switching method than for the other two methods. This was to be expected, since in the networks randomised by the switching model, certain triplets of nodes form connected patterns that are not acyclic, hence reducing the number of triplets that do form acyclic patterns. For the Karrer-Newman method, the difference in pattern counts can be attributed to multiple edges being treated as single edges by the motif counting algorithm, thus in effect reducing the number of edges. On average there were 3747 multiple edges in these random networks.

Notice that, for the ordered switching method, the difference between c_i and $\overline{c}_i^{rnd}$ equals 1,364,629 and the standard deviation σ_i equals 943, for all four patterns as dictated by Corollary 5.1.7.

When we introduced our definitions of motifs and anti-motifs (Definitions 5.1.3 and 5.1.4) we mentioned that, as opposed to [82], we do not require patterns to occur at least 4 times to be considered a motif. The reason for us to exclude this criteria is that it is always satisfied for the motifs detected in our experiments. It hence does not make a difference to our results. However, in the case of anti-motifs, requiring a minimal number of occurrences would make a difference in our data. The non-acyclic patterns do not occur at all in our data, but do occur in the randomised networks generated by the switching model.

Even though there are differences in pattern counts for the three different random network models, the directed acyclic patterns that are motifs and anti-motifs are the same. The feedforward loop (m_4) is a motif and patterns m_2 and m_3 are anti-motifs. However, for the switching model, which randomizes with respect to $\mathcal{G}_3$, there are also occurrences of other (cyclic) patterns. Figure 5.4 shows the nine additional cyclic patterns,

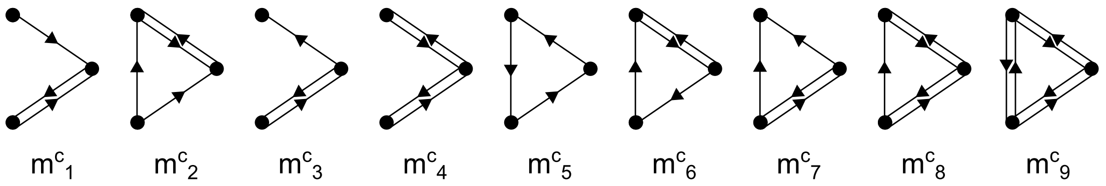


Figure 5.4.: The nine cyclic 3-node patterns that may occur in simple directed networks ($\mathcal{G}_3$), in addition to the four directed acyclic 3-node patterns illustrated in Figure 5.1.

$m^c_1, \ldots, m^c_9$ that can appear in simple directed networks. Patterns $m^c_1, m^c_2, m^c_3, m^c_4, m^c_5, m^c_6$ and m^c_7 are all classified as anti-motifs. For each of these patterns, the probability p of a pattern appearing an equal or smaller number of times in the random networks than in the real network equals 0. Pattern m^c_5 is the most significant anti-motif with a Z score of -43.

	c_1	c_2	c_3	c_4
HEP-th (real network)	22567810	5943062	4098942	1469679
Karrer-Newman	21266991	7018270	5229368	94858
	±47922	±9022	±6192	±827
Switching	23868779	7204695	5457398	80223
	± 9040	±11078	±3129	±791
Ordered Switching	23932439	7307691	5463571	105050
	± 943	±943	±943	± 943

Table 5.2.: The number of times pattern m_1, m_2, m_3 and m_4 were found in the theoretical physics network and the average number (± s.d.) of times they were found in the three random network ensembles. Each random network ensemble consisted of 1000 graphs.

The findings for the phenomenological physics citation network are similar to the findings for the theoretical citation network. Pattern counts are higher for the networks generated by the ordered switching model than the networks generated by the other two random network models. In this case the average number of multiple edges in the Karrer-Newman random graph ensemble was 1241. Pattern m_2 and m_3 are anti-motifs, and the feedforward loop (m_4) is a motif. When using the switching model as a null-model, patterns $m^c_1, m^c_2, m^c_3, m^c_5, m^c_6$ and m^c_7 are identified as anti-motifs ($p = 0$). The most significant of these anti-motifs is pattern m^c_5 with a Z score of -27.

	c_1	c_2	c_3	c_4
HEP-ph (real network)	9035618	3853645	3884377	976048
Karrer-Newman	9804897 ±7067	4753445 ±2103	4757988 ±3146	26156 ±215
Switching	9977282 ±1624	4784925 ±2410	4830385 ±1020	22511 ±187
Ordered Switching	9984869 ±212	4802896 ±212	4833628 ±212	26797 ±212

Table 5.3.: The number of times pattern m_1, m_2, m_3 and m_4 were found in the phenomenological physics network and the average number (± s.d.) of times they were found in three random graph ensembles, all consisting of 1000 graphs.

The abundance of feedforward loops in citation networks may seem surprising at first, but has previously been observed [30] and explained in [130]. The authors' explanation of this abundance is two-fold; firstly, cited papers are often on a similar topic as the citing paper and secondly, scientists often navigate the literature by following citations and references.

5.2. Neighbourhood distinctiveness

In this section we investigate the potential for using neighbourhood attributes to match unidentified entities across networks. The motivation is to identify individuals across the dark social networks that underlie recorded networks [88]. There are many sets of large databases that contain overlapping and complementary information. For instance, a social network example is the Twitter, Facebook and LinkedIn databases and a bibliographic example is the CiteSeer, DBLP, Google Scholar and Scopus databases.

Sometimes the same entity appears in multiple databases but with a different description, either due to errors or to the data having inherent differences, such as user names within different social media databases. Matching across databases at the instance level (also termed reference reconciliation), that is, matching different individual descriptions referring to the same real-world entity, is important for both discovery and database management [43, 28, 71, 98].

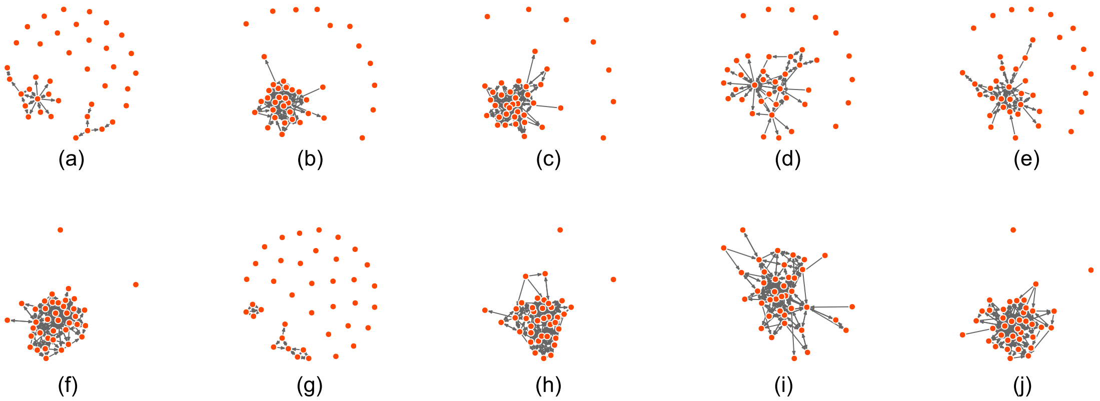


Figure 5.5.: Ten out-neighbourhoods in the Enron network, each with exactly 39 nodes.

Our focus here is discovery: sometimes entities represent humans or organisations operating incognito or under several aliases, for either legal or illegal reasons. From this perspective it is natural to investigate the *context* of an entity: those entities in a database that are directly linked or related to it [28], and further, to investigate their inter linkages (eg. [71, Figure 1]), i.e. in network terms, the neighbourhood of a node (see Definition 1.1.14).

In Subsection 5.2.1 we introduce the concept of an *influence neighbourhood* in a communication network and a citation network. In Subsection 5.2.2 we discuss our experiments on the distinctiveness of these influence neighbourhoods.

5.2.1. Influence neighbourhoods

In a citation network the in-neighbourhood (Definition 1.1.14) of a paper corresponds to the subnetwork of papers that were directly influenced by the paper. In a communication network the out-neighbourhood (Definition 1.1.14) of a sender corresponds to the subnetwork of receivers that were directly influenced by the communication. We will refer to both the in-neighbourhood of a paper and the out-neighbourhood of an email contact as their *influence neighbourhood*.

We first check our approach with an Enron email database, since it is a publicly available directed communication network in which a subset of people in the underlying social network were acting illegally. The CiteSeer network was selected to test if we can used the influence neighbourhood to match entities across databases. In [51], my collaborators describe experiments that match papers in the CiteSeer database to papers in a different citation database.

We show experimentally that these influence neighbourhoods are highly distinctive in both the Enron email database (see Appendix A.5) and the CiteSeer citation network (see Appendix A.3). Figure 5.5 shows ten influence neighbourhoods in the Enron email network, all with exactly 39 nodes. Figure 5.6 shows ten influence neighbourhoods in the CiteSeer citation network, all with exactly 130 nodes. Visually the structures of these neighbourhoods are quite distinctive.

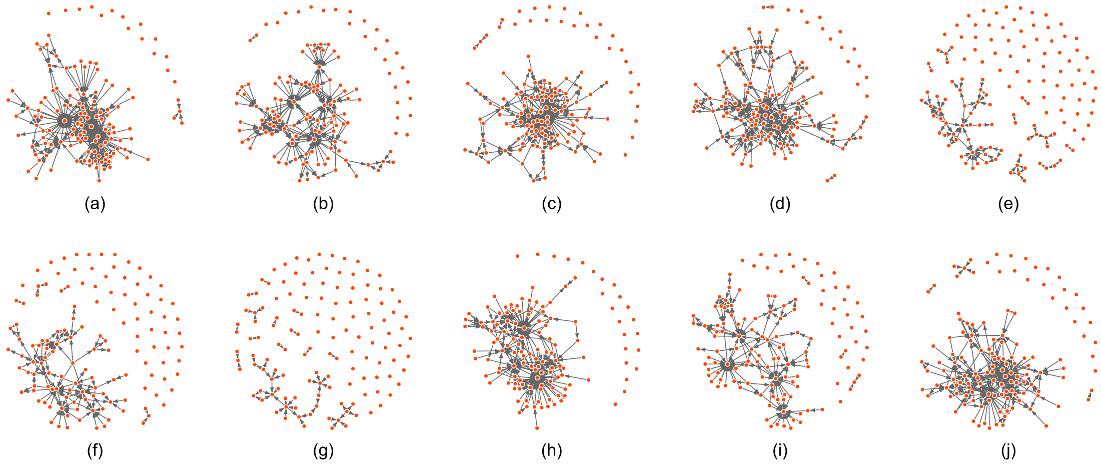


Figure 5.6.: Ten in-neighbourhoods in the CiteSeer network, each with exactly 130 nodes.

In order to measure the distinctiveness of neighbourhoods it is necessary to have a measure of either their similarity or their difference. We use the Jaccard distance (Definition 5.2.1) to measure the dissimilarity of the influence neighbourhoods. This metric is a simple measure of difference in labelled nodes. In essence this is the ground truth. If the nodes are uncertainly labelled then metrics which are structural or fuzzy would be necessary.

Definition 5.2.1. Let A and B be finite sets. The Jaccard distance between A and B is defined as:

$$d_J(A, B) = 1 - |A \cap B|/|A \cup B|.$$

5.2.2. Experiments in communication and citation networks

We ran two experiments to test if the influence neighbourhoods in the CiteSeer and Enron networks are distinctive. We partitioned the papers in the CiteSeer database by citation number (in-degree) into ranges that increase exponentially, see Table 5.4. The smaller Enron database was similarly partitioned by out-degree, for comparison.

For the first experiment, 100 nodes were selected from each of the partitions listed in Table 5.4 (with replacement if necessary). Each node was paired with 1,000 nodes, randomly selected from the whole database excluding the node itself, and the Jaccard distance between them was calculated.

For both the Enron network and the CiteSeer network, the cumulative relative frequency drops from 1 very rapidly. For the Enron network, out of the 1,000,000 random pairings only 15 pairs are found with Jaccard distance 0, all in Partition 2, only 858 pairs with Jaccard distance below 0.7 and only 8968 pairs below 0.9. For the CiteSeer network, out of the 1,100,000 random pairings 0 pairs are found with Jaccard distance 0, only 6 below 0.7 and only 46 pairs below 0.9.

Partition	***k* Range**	**Enron**	**CiteSeer**
1	1	8711	51949
2	2-3	3270	50823
3	4-7	1320	40313
4	8-15	528	26669
5	16-31	214	14510
6	32-63	98	6700
7	64-127	56	2543
8	128-255	41	793
9	256-511	9	207
10	512-1023	1	43
11	$\geq$ 1024	0	8

Table 5.4.: Partitioning of the databases by number of recipients (Enron) or citations (CiteSeer).

For our second experiment we look at worst-case matching, where neighbourhoods are guaranteed to overlap. Again, we chose 100 nodes randomly from each partition, and then matched each of the nodes to all of the nodes in the database that had at least one common neighbour with the selected node. This means only node pairs which are most likely to have a low Jaccard distance are tested.

For the Enron network, out of the 531,714 nearby pairings only 759 pairs are found with Jaccard distance of 0, and all of these are in Partitions 1 and 2. There are only 3781 pairs with Jaccard distance below 0.7 and 30,957 pairs with Jaccard distance below 0.9. For the CiteSeer network, out of the 537,932 nearby pairings 0 pairs are found with Jaccard distance of 0, only 54 pairs with Jaccard distance below 0.7 and 1163 pairs with Jaccard distance below 0.9.

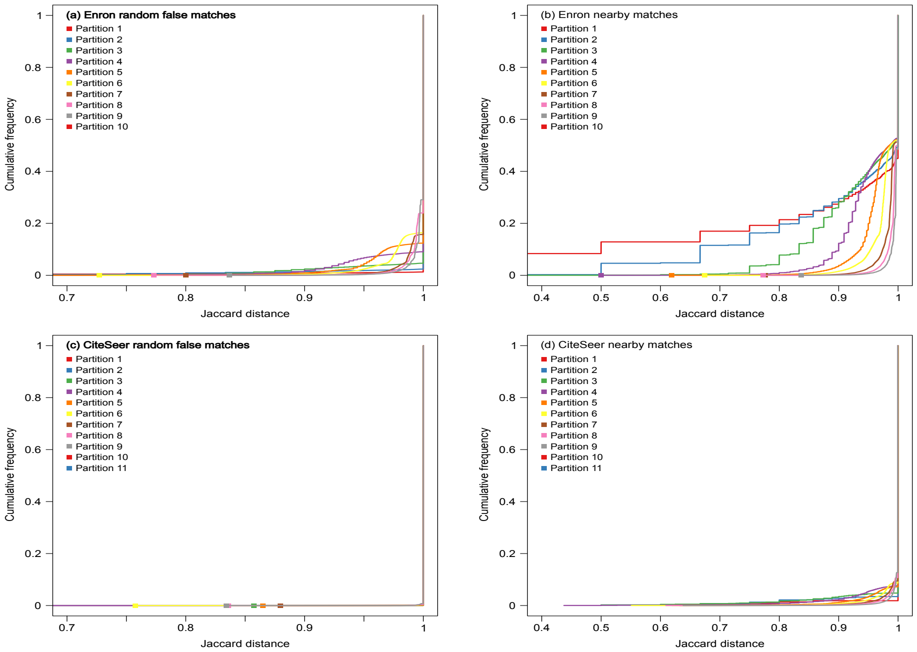


Figure 5.7.: Distinctiveness of neighbourhoods within Enron and CiteSeer networks

The cumulative distributions are shown in Figure 5.7. We conclude the neighbourhoods are highly distinctive in each of these databases, and distinctive in similar ways. So, citation networks are reasonable proxies for the type of networks we are interested in. These CiteSeer results appear in [59], in the rest of this paper my collaborators showed that the Jaccard distance between influence neighbourhoods is a promising approach for matching nodes across databases. This approach relies on being able to (partially) match node labels across the networks.

The Enron and CiteSeer comparison appears in [51]. Based on this, my collaborators present results of a similar but more elaborate experiment, that confirms that influence neighbourhoods of nodes can be used to match nodes across databases, provided there is a substantial number of nodes for which labels can be matched.

5.3. Conclusion

We discussed our work on motif finding in directed acyclic networks. We showed that regardless of what null-model was used, the patterns identified as motifs in a set of citation networks remained constant. The average pattern counts in random network ensembles generated by the three different models differed, but not so much as to classify different patterns as directed acyclic motifs or anti-motifs. However, the switching model did find redundant directed *cyclic* anti-motifs. The feedforward loop was found to be a motif in the two citation networks. The abundance of feedforward loops in citation networks was observed previously in [30] and [130].

We recommend using the ordered switching model to randomize directed acyclic networks, since it is the only random network model that generates directed acyclic networks with fixed degree sequence uniformly at random. A possible problem with the ordered switching algorithm is that it can be slow (see also Chapter 3). It would be interesting to develop a Curveball style algorithm (see Chapter 4) for directed acyclic networks.

We have demonstrated that the influence neighbourhood is highly likely to differentiate one paper from another within the CiteSeer database. Similarly the influence neighbourhood of an email address in the Enron email network is highly likely to differentiate one person from another. The influence neighbourhood shows initial promise for good matching performance across databases. Further experimental results are discussed in [59, 51].

6. Topological data analysis for networks

In this chapter we discuss recent developments in the topological data analysis of networks. Topological data analysis has emerged over the past 15 years and has found many applications in the analysis of point cloud data, image and shape analysis [22, 23, 109, 87, 29]. For an introduction to topological data analysis and in particular persistent homology, see Section 1.5.

More recently, several papers have been published in which ideas from topological data analysis are used in the analysis of complex networks [27, 33, 55, 99, 101, 74, 15]. This brings a new approach to the study of complex networks. Most existing network measures originate in statistical mechanics and focus on statistics of local network properties. For instance, the node degree, the clustering coefficient and motifs fall in this category. Such statistics have proven very useful. However, they do not capture the complete structure of a network. Topological data analysis provides an additional network metric.

The persistent homology of a network differs from other network metrics in three crucial ways. Firstly, homology reveals information on the mesoscopic structure of a network that local statistics do not provide. Secondly, the idea of representing relations as a network can be enriched by using simplicial complexes. Finally, a parametrised family of networks can be analysed as a single object. This reveals features that would be overlooked by analysing networks at a single parameter value.

The aim of this chapter is two-fold. Firstly we provide an overview of the three aspects of persistent homology mentioned above, with respect to the current literature. Secondly we describe our experimental results for collaboration networks.

The rest of this chapter is organised as follows. Section 6.1 discusses the current literature on topological data analysis for networks and how it relates to our experimental work. Section 6.2 discusses how simplicial complexes can be used in the study of relational data. Section 6.3 discusses the interpretation of the Betti numbers for networks and simplicial complexes associated to networks. Section 6.4 discusses different ways to filter networks and the simplicial complexes associated to networks. In Section 6.5 our results from a case study on collaboration networks are discussed. Finally Section 6.6 discusses our conclusions and points out potential future directions.

My contributions to the literature as discussed in this chapter are outlined below.

- Presenting an overview of the current literature on, and techniques available in topological data analysis of networks.

- One of the first case studies where persistent homology is used to analyse a weighted network. Some of the results discussed here have been published in [27].

In the remainder of this chapter we will use the term *cycle* to refer to a generator of a 1-dimensional homology class, not to the graph theoretical concept as in Definition 1.1.4.

6.1. Literature review

The earliest application of persistent homology to a network related problem that we know of, studies the coverage of a domain for sensor networks [44, 33]. Ideas from persistent homology were used to find out if a given domain is covered by sensors of which the location is unknown. Each sensor has local information about nearby sensors and the boundary of the domain. The authors use homology and ideas from persistence to answer this coverage question.

Our interest in persistent homology is to use it as an additional statistic in the analysis of complex networks. This is more similar to the work by Horak et al. [55], where persistent homology is used to distinguish networks generated by different random network models. However, the filtration used in that work does not fully capture the idea of persistence. In fact, it does not show interesting features in addition to the clique structure of a network, which can be studied without referring to homological concepts [100]. We discuss this in more detail in Section 6.4.

The work by Lee et al. [74, 73, 75] successfully uses persistent homology and single linkage clustering to classify functional brain networks into three different categories: ADHD, autism spectrum disorder and control. In their work the zeroth Betti number and the corresponding single linkage dendrogram are used to analyse weighted networks. The filtration they used is based on edge weights and does capture the idea of persistence.

We combined the ideas from Lee et al. and Horak et al. by using both the weighted network filtration and the clique complex construction. That is, we computed the persistent homology groups of the filtration of clique complexes of threshold networks.

After some of the results of this chapter were published [27], we became aware of a paper [100] that also uses persistent homology for network analysis. In that paper, three different constructions to obtain a filtration from a network are discussed. The most effective one was the same as the filtration we had used to analyse collaboration networks. The other two methods were: firstly associating a metric with the nodes of the network (i.e. shortest path distance) and treating the nodes as points of a pointcloud with the given metric, and secondly the k-skeleton filtration that was introduced in [55]. It was found that when using the metric construction, persistent homology points out an optimal threshold at which to measure homology. However, tracking the homology over a range of distances did not provide additional information. The authors also found that the k-skeleton filtration gave little information beside the clique structure of the network.

The persistent homology of the most effective filtration, the clique complexes of the threshold filtration, was named the **weighted clique rank homology** [100]. In a later paper [101] by the same authors, weighted clique rank homology is further investigated. They found two distinct classes of networks, one with cycle distributions that are markedly different from randomized versions of the network and one with cycle distributions very close to random versions.

Other recent applications of persistent homology in network analysis are analysing different states of brain networks [99] and predicting survival rates in cancer based on protein interaction networks [15].

6.2. Simplicial complexes for relational data

The use of simplicial complexes (see Definition 1.5.1) to study relational data dates back to Atkins [7] in the early 1970s. Simplicial complexes can be seen as a generalization of networks. A 1-dimensional simplicial complex is equivalent to a simple undirected network, with each 1-simplex corresponding to an edge. In addition, higher dimensional simplicial complexes can also encode relations between three or more entities. A nice consequence is that all statistics for simplicial complexes are also statistics for networks.

Recently the use of simplicial complexes for relational data analysis has seen a renewed interest [123, 129, 124, 85, 77, 47, 76]. In [77] the statistical properties of simplicial complexes associated to complex networks were investigated. Several constructions were used to associate simplicial complexes to networks (see also [61]) but the resulting statistics did not differ significantly between different constructions. We therefore follow the authors in focussing on the simplest construction: the clique complex.

Definition 6.2.1. Let $G = (V, E)$ be a simple undirected network. The **clique complex** $C(G)$ of G is the simplicial complex with vertices V, and simplices corresponding to the cliques of G. That is, for each k-clique in G there is a $k - 1$ simplex on the corresponding k vertices in $C(G)$.

Notice that the clique complex is indeed a simplicial complex. Any subset of vertices of a simplex is by definition itself a simplex, since any subset of vertices of a k-clique is an l-clique (with $l \leq k$).

The kind of information most obviously contained in the clique complex of a network is edge density. Densely connected areas are represented as higher dimensional structures, in a sense merging such areas into single entities. We will discuss the relation to other network properties and homological measures in more detail in Section 6.3.

Perhaps a more natural setting in which to encode relational data as a simplicial complex is found in the study of bipartite networks [85, 124]. Bipartite networks represent a relation between two distinct sets of objects. There are many examples of such relations, such as actors playing in movies, scientists writing papers, consumers buying products and people attending events.

Definition 6.2.2. A **bipartite network** is a network $G = (V, E)$ with nodes that can be partitioned in a set of primary and secondary nodes $V = P \sqcup S$ in such a way that all edges are of the form $\{p, s\}$ with $p \in P$ and $s \in S$. We will sometimes use the notation $G = (P, S, E)$.

It is common to analyse bipartite networks as simple non-bipartite networks by projecting the network onto the most interesting type of its nodes [96, 134, 135]. This is called the one-mode projection of a bipartite network.

Definition 6.2.3. Let $G = (P, S, E)$ be a bipartite network. The **(unweighted) one-mode projection** of G onto P is the network $G_P = (P, E_P)$ with nodes P and $\{p_1, p_2\} \in E_P$ if and only if p_1 and p_2 share a neighbour in G. We can similarly define G_S, the (unweighted) one-mode projection of G onto S[1].

The one-mode projection of a bipartite network loses some information of the network.

[1] In Section 6.5 we analyse collaboration networks. These networks are constructed as the one-mode projection of bipartite networks with weights assigned to their edges. The weights are chosen in such a way to reflect the strength of each relation [90]. Note that this is not the only commonly used assignment of weights (another is the number of common neighbours).

Take for instance collaboration networks: the one-mode projection of a bipartite network of scientists writing papers. In a collaboration network, scientists are connected by an edge if they have co-authored at least one paper. In such a network, triangles are ambiguous. A triangle, connecting three scientists, could represent three collaborations, each written by a distinct pair of scientists. However, it could equally well represent a single paper co-authored by the three scientists. This information can quite naturally be incorporated in a simplicial complex. By simply inserting higher dimensional faces corresponding to collaborations between three or more authors [85] we can distinguish between these situations. We use the terminology found in [123] to define this construction.

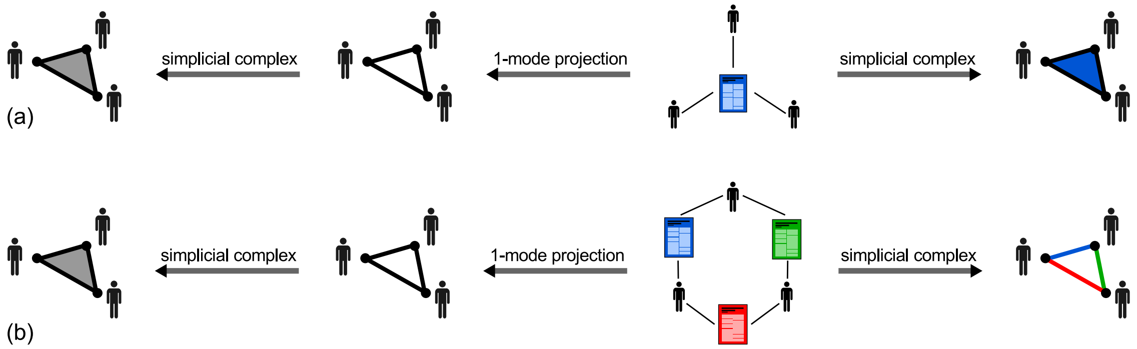


Figure 6.1.: The one-mode projection and its associated clique complex can not distinguish between the following two situations. (a) Three scientists have written one paper together. (b) Three scientists who have coauthored three papers pair-wise. The labelled complex does distinguish between these two situations.

Definition 6.2.4. Let $G = (P, S, E)$ be a bipartite network. The **labelled complex** $\Delta_P(G)$ associated to G with respect to P has vertices P and simplices $\sigma_s = N(s)$, $s \in S$ (Definition 1.1.14) together with all faces of these simplices. That is, each simplex corresponds to the nodes related to a node $s \in S$ (its neighbours). Similarly we may define $\Delta_S(G)$.

Notice that the one-mode projection of G onto P is the 1-skeleton of $\Delta_P(G)$. As such, the labelled complex truly generalizes the one-mode projection of a bipartite network. Figure 6.1 shows the difference between the clique complex associated to the one-mode projection of a bipartite network and the labelled complex associated to a bipartite network.

An interesting fact about these labelled complexes is that the homology groups of $\Delta_P(G)$ are equal to those of $\Delta_S(G)$. In [124] this is exploited to simplify homology computations and to find the homological backbone of such simplicial complexes.

For completeness we mention that it is possible to construct a simplicial complex for non-bipartite networks in a similar fashion.

Definition 6.2.5. Let $G = (V, E)$ be an undirected network. The **neighbourhood complex** of G is the simplicial complex with vertices V and simplices $\sigma_v = N(v)$, $v \in V$, together with all faces of these simplices. Similarly for $G = (V, E)$ a directed network, the **out-neighbourhood complex** and **in-neighbourhood complex** are defined as the complexes with vertices V and simplices $\sigma_v = N^{out}(v)$ or $\sigma_v = N^{in}(v)$, $v \in V$, together with all faces of these simplices, respectively.

6.3. Network homology

We limit our attention to low-dimensional homology groups. These groups are easiest to interpret, fastest to compute and contain a wealth of information. We will make use of the following well-known facts (see Section 1.5.1). Firstly, the homology groups of a d-dimensional simplicial complex are trivial in dimension $d+1$ and higher. Secondly, the i-th homology group of a simplicial complex only depends on the $i+1$-skeleton of the simplicial complex, that is $H_i(X) = H_i(X_{i+1})$ with X_i the i-skeleton of X. We discuss the interpretation of the zeroth, first and second homology groups of the skeleta of a clique complex. We use the famous network known as Zachary's karate club (Appendix A.11) as an illustrative example.

Zachary's karate club is a network representing the friendships between 34 members of a karate club as illustrated in Figure 6.2. To analyse the low-dimensional homology groups of its clique complex, K, we built the 3-skeleton, K_3 of K. This simplicial complex consists of 34 0-simplices (vertices), 78 1-simplices (edges), 45 2-simplices (triangles) and 11 3-simplices (tetrahedra). Table 6.1 gives an overview of the Betti numbers of the skeleta of this clique complex.

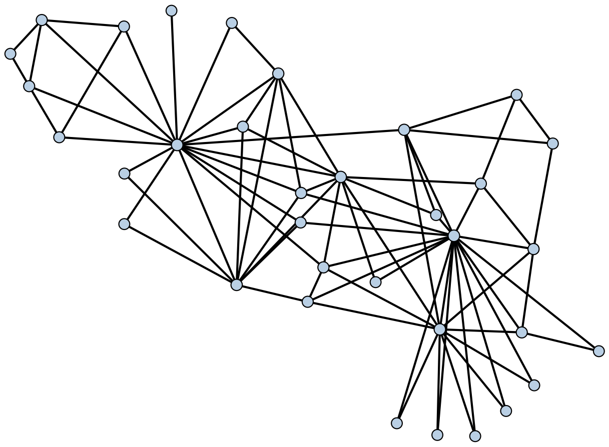


Figure 6.2.: Zachary's karate club.

Skeleton	β_0	β_1	β_2
K_0	34	0	0
K_1	1	45	0
K_2	1	9	9
K_3	1	9	0

Table 6.1.: Betti numbers of the clique complex K of Zachary's karate club. Notice that $\beta_i(X_j) = 0$ when $i > j$ since the homology groups of a j-dimensional complex are trivial in dimension $j+1$ and higher. Furthermore $\beta_i(K_j) = \beta_i(K_{i+1})$ for all $j > i$ since the i-th homology group of a simplicial complex only depends on its $i+1$-skeleton. Hence, the highlighted cells are the only cells that contain interesting information.

The 0-skeleton of a clique complex of a network corresponds to the nodes of the network. Its zeroth Betti number equals the number of connected components, in other words the number of nodes n. For Zachary's karate club $\beta_0(K_0)$ equals 34.

The 1-skeleton of the clique complex of a network is the network itself. The Betti numbers can only be non-trivial in dimension zero and one. By measuring the homology in these dimensions we obtain information on the number of connected components of a network (zeroth Betti number) as well as the number of independent cycles in the network (first Betti number). The first Betti number is strongly correlated with the number of triangles in a network and hence measures a type of clustering. In our example, Zachary's karate club consists of a single connected component, and hence $\beta_0(K_1) = 1$. The first Betti number of the network equals 45. In this case, the first Betti number equals the number of triangles in the network, however this is not a general rule. In general the first Betti number can be higher or lower than the number of triangles in a network. For instance the 4-cycle, C_4 has first Betti number equal to one, but does not contain any triangles. The 4-clique K_4 on the other hand contains four triangles but has first Betti number three; it only contains three independent cycles, any triangle can be obtained by 'adding' the other three.

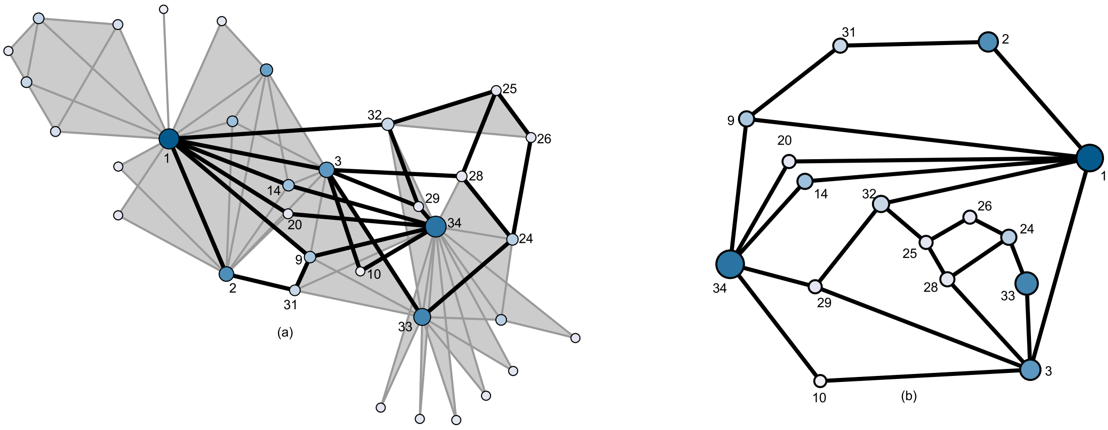


Figure 6.3.: (a) The 2-skeleton of the clique complex of Zachary's karate club. This simplicial complex has first Betti number equal to 9; the corresponding cycles are highlighted. (b) The homological backbone of the network. The backbone includes only edges that contribute to the first homology group of the clique complex.

If we instead analyse the homological properties of the 2-skeleton of the clique complex, we obtain information not readily measured by current network statistics. The 0-dimensional homology groups do not change, as they only depend on the unchanged 1-skeleton of the simplicial complex. However, the 1-dimensional homology group changes dramatically, since every triangle will be 'filled in', i.e. a surface, in the clique complex. This causes many cycles to disappear and the first Betti number to be much lower. The first Betti number now measures the independent cycles in the network that are more than three edges long, and not filled by cross-links. These cycles in a sense form structural holes and can be seen as a backbone of the network [99, 124]. The additional structure offered by the clique complex reveals an interesting new measure of the topology of the network. Figure 6.3 shows the nine cycles that generate the first homology group of the clique complex of Zachary's karate club.

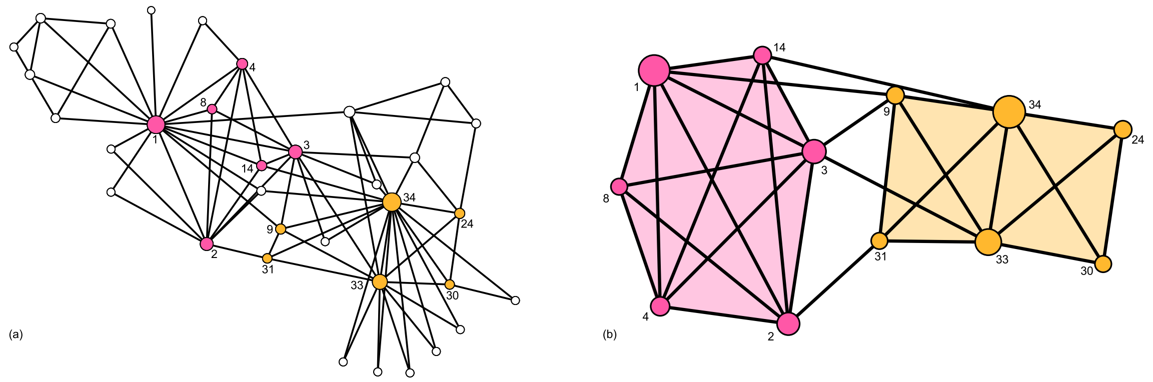


Figure 6.4.: The 2-skeleton of the clique complex (see Figure 6.3) has non-trivial 2-dimensional homology, its second Betti number equals 9. (a) The nodes involved in 2-dimensional homology classes. (b) In this case, each 2-dimensional homology class corresponds to a tetrahedron, or a 4-clique. There are two classes generated by the group of yellow nodes, and seven by the pink nodes. None of these homology classes will exist in the 3-skeleton of the clique complex, since all cliques will be filled in by solid tetrahedra.

The second Betti number of the 2-skeleton corresponds to the number of voids and is dominated by empty tetrahedra. As such the second Betti number roughly corresponds to the number of 4-cliques in the network, and gives us a measure of strong local clusters in the network. Figure 6.4 shows the nodes and triangles that correspond to these voids for the 2-skeleton of the clique complex of Zachary's karate club network.

Finally the second Betti number of the 3-skeleton of a clique complex corresponds to voids that are larger than tetrahedra. All tetrahedra are now solid and thus no longer correspond to 2-dimensional homology classes. In fact, the simplest structure corresponding to a void in (the 3-skeleton of) a clique complex is the surface formed by an octahedron. This can be seen by realizing that all 'great circles' of such a void need to be of at least length four. For the clique complex of Zachary's karate club network no such higher dimensional voids are present.

6.4. Constructing filtrations from networks

The type of network filtration that is suitable to use in persistent homology computations depends on the application at hand. Several different filtrations have been discussed in the literature. Horak et al. [55] use the following filtration based on the skeleta of the clique complex of a network.

Definition 6.4.1. Let K be a d-dimensional simplicial complex. The **skeletal filtration** of K is given by

$$\emptyset \subset K_0 \subset \cdots \subset K_d = K$$

where K_i is the i-skeleton of K.

As mentioned briefly in the introduction to this chapter, this filtration does not truly capture the idea of persistence. In particular, it follows from our remarks in the previous section that any homology class in this filtration either lives forever or dies one step after being born, since any i-dimensional homology class that dies, dies as a boundary of a $i+1$-dimensional simplex (see for Example Table 6.1).

In the work of Lee et al. [74, 73] the philosophy behind persistent homology was captured much better. The neural networks analysed in this work are weighted networks. Usually these networks are converted to binary networks by selecting a suitable threshold and only considering edges that are stronger than this threshold. Lee et al. introduce a weighted filtration to analyse brain networks at multiple scales. They focus on the zeroth Betti number, monitoring how the connected components merge throughout the filtration.

Definition 6.4.2. Let $G = (V, E, w)$ be a weighted network with $w : E \to \mathbb{R}_{>0}$. The **threshold network** $G(w^*) = (V, E(w^*))$ contains all edges $e \in E$ with weight greater than or equal to w^*, that is with $w(e) \geq w^*$. Write the elements of $w(E)$ in descending order: $w_1, w_2, \ldots, w_k$. The **weighted network filtration** of G is defined by

$$\emptyset \subset G(w_1) \subset G(w_2) \subset \cdots \subset G(w_k) = G.$$

Changing the threshold weight for a threshold network is like changing the resolution at which a network is analysed [97].

In our analysis of weighted collaboration networks we use a similar construction. However we enrich the filtration by associating the clique complex to each of the networks in the weighted network filtration. A network filtration can always be extended to a filtration of the clique complex of the network. In fact, this is the same construction as that of the complexes in the Vietoris-Rips filtration of a point cloud from its 1-dimensional subcomplex (Definition 1.5.14). We define the clique complex filtration as follows.

Definition 6.4.3. Let $\{G_i\}_{i=0}^k$ be a filtration of a network G. The **clique complex filtration** of $C(G)$ is defined as

$$\emptyset \subseteq C(G_1) \subseteq C(G_2) \subseteq \cdots \subseteq C(G_k).$$

Petri et al. [101] independently came up with the idea of combining the weighted network filtration and the clique filtration to analyse weighted networks. We adopt their terminology and will refer to this filtration as the **weighted clique rank filtration**.

Although the following filtrations are not used in this thesis, we introduce them for the sake of completeness and future applications.

Similarly to filtering weighted networks by their weight, we introduce a time based filtration for networks that are growing over time.

Definition 6.4.4. Let $G = (V, E, t)$ be a growing temporal network with time function $t : V \cup E \to \mathbb{R}_{>0}$ such that for all edges $(x, y) \in E$ we have $t(x), t(y) \leq t((x, y))$. The **accumulated network** $G(t^*)$ [54] at time t^* is the network with nodes $V(t^*) = \{v \in V \mid t(v) \leq t^*\}$ and edges $E(t^*) = \{e \in E \mid t(e) \leq t^*\}$. Write the elements of $t(E)$ in ascending order: $t_1, t_2, \ldots, t_k$. We define the **temporal network filtration** of G as

$$\emptyset \subseteq G(t_1) \subseteq G(t_2) \subseteq \cdots \subseteq G(t_k) = G.$$

As with the weighted network filtration, this construction can be extended to a filtration of the clique complex of G. We will refer to this filtration as the **temporal clique rank filtration**.

As a final example, we define the weighted labelled filtration that can be obtained from a bipartite network.

Definition 6.4.5. Let $G = (P, S, E)$ be a bipartite network. The **weighted labelled complex** associated to G with respect to P has vertices P and simplices the unique[2] simplices of the labelled complex $\Delta_P(G)$. We associate a weight $w(\sigma)$ to each simplex by summing over the number of times it appears as a face of a simplex in $\Delta_P(G)$. Let $w_1, w_2, \dots, w_k$ be the finite list of these weights in ascending order. We obtain a filtration by thresholding the weighted labelled complex at different weights, and call it the **weighted labelled filtration**.

6.5. Persistent homology of collaboration networks

Most of the results of the thesis discussed in this section are published in [27]. We analyse collaboration networks as a proxy for social networks [89, 91]. A collaboration network has nodes corresponding to scientists and edges corresponding to co-authored papers. In fact, a collaboration network is the one-mode projection of a bipartite network of scientists and their papers. The edges in a collaboration network can be weighted to reflect the regularity and intensity of the collaboration between authors. We follow [90] in assigning weights to edges: a paper written by n authors contributes $1/_{n-1}$ to the weight of each edge between author pairs. For instance a paper with only two authors contributes weight 1 to the edge between its authors. A paper with six authors by contrast, only contributes weight $1/_5$ to each of the 15 edges between pairs of its authors. Strong connections in collaboration networks thus correspond to pairs of authors who collaborate often and in small groups. It is reasonable to expect that a collaboration network reflects the structure of the underlying social network, since regular collaboration probably indicates social ties.

[2] It is possible for two vertices in the secondary node set S to have the exact same neighbours in P, the corresponding simplex is only included once.

We analyse four collaboration networks as described in Appendices A.1, A.4, A.6 and A.9. These networks were derived from different collections of academic papers and are available as weighted collaboration networks. Their underlying bipartite networks are not readily available.

Granovetter's 'Strength of weak ties' theory [45] is well-known in the social sciences. Granovetter argues that two individuals who both have a strong tie with the same third person, are very likely to also know each other. For instance, your partner and your mother are likely to know each other, this tie might be weak, but nevertheless there is a tie. Taking this assumption as an axiom lead Granovetter to an interesting discovery. An edge is called a *bridge* if the nodes that it connects have no common neighbours, locally this edge is the only way for information to pass from one side of the bridge to the other. If one of the nodes incident with the bridge has at least one strong tie to a different node v, then the bridge *has to be a weak connection*: if it were strong, then by assumption, v is also connected to the other node incident with the bridge. But this contradicts the fact that the edge is a bridge. Granovetter's theory leads to a view of social networks as consisting of highly clustered regions of strong connections and weak bridges connecting these clustered areas (see Figure 6.5).

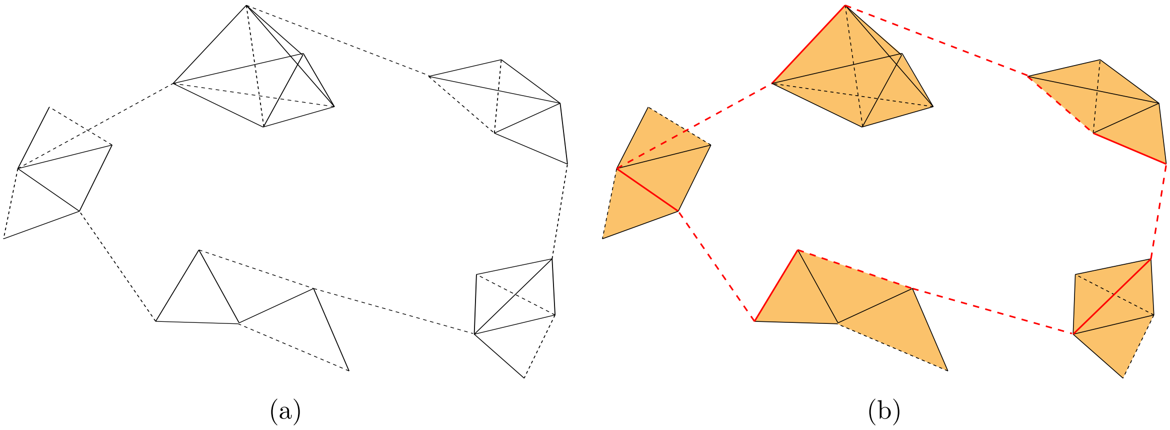


Figure 6.5.: (a) An illustration of a social network with weak (dashed) and strong (solid) ties, distributed in agreement with Grannoveter's theory. (b) The 1-simplex (red ties) of the clique complex associated with the same network.

In our analysis of collaboration networks, we used the weighted clique rank filtration (see

Definitions 6.4.2 and 6.4.3) of the weighted networks. If the weights in a network are indeed distributed according to Granovetter's theory, then we expect low weight edges as part of 1-cycles, since we expect bridges to be part of the 1-cycles of length more than 3. Figure 6.5(b) illustrates a typical generator of a 1-cycle (the red highlighted edges). That is, we expect most cycles to appear in the final stages of the filtration, when w^* is close to 0.

The next two sections discuss the results of our topological data analysis of four collaboration networks. The first network that we analysed is relatively small and we discuss qualitative results. For the larger networks this was not done. Instead, we compared the persistence barcodes of the clique rank complex for each of these networks to persistence barcodes of $G(n, m)$ Erdős-Rényi random networks (Definition 1.1.20), with equal number of nodes n and edges m. We show that certain topological features are present in the real networks but not in the random networks, implying that this structure is due to some organisational feature and not a random process.

Besides our own custom scripts, we used the following software in our experiments: JavaPlex (Appendix B.2.4) for the persistent homology computations, Gephi (Appendix B.2.2) for some of the network visualisations and the Persistence Landscape Toolbox (Appendix B.2.6) for the landscape distance computations.

6.5.1. Collaboration network of network scientists

The first network that we analyse is a collaboration network of network scientists (see Appendix A.9). We restrict our analysis to the largest connected component of this collaboration network. This component consists of 379 nodes and 914 edges with weights ranging from 0.125 to 4.75.

We use $K(w^*)$ to denote the clique complex of the threshold network $G(w^*)$. To be clear, we are inspecting the homology groups of the whole clique complex, not those of its sub-skeleta.

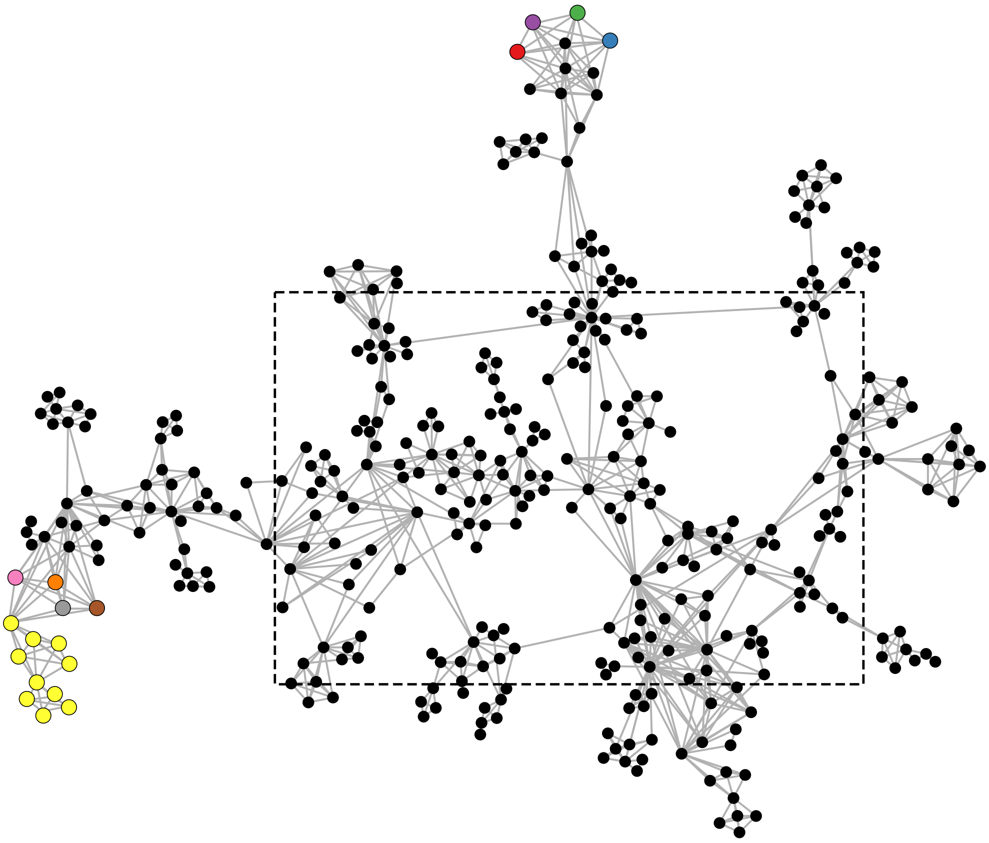
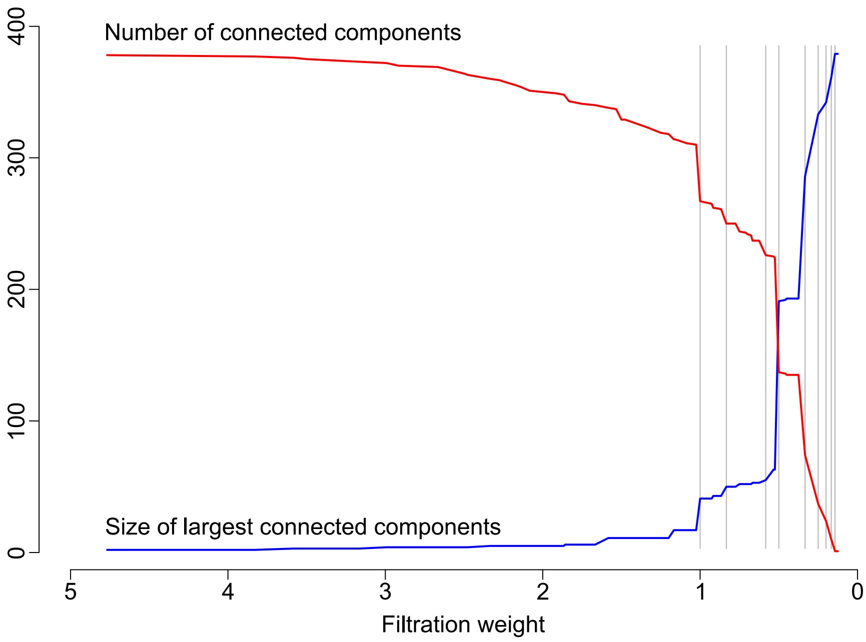


Figure 6.6.: The largest connected component of the network science collaboration network. In the final stages of the weighted clique filtration, β_0 drops from 10 to 1. The connected components corresponding to these ten homology classes are visualised above; nearly all nodes are in a single component (the black nodes), there is a connected component of size ten (the yellow nodes) and there are a further eight connected components consisting of a single node each. The box in the figure indicates the part of the network illustrated in Figure 6.8.

We first focus on the 0-dimensional homology groups of the weighted clique rank filtration. For $w^* > 4.75$, the clique complex $K(w^*)$ is just a collection of vertices. Its zeroth Betti number equals 379, the number of nodes of the network. The number of connected components reduces throughout the filtration as more and more edges are included. Finally, the zeroth Betti number of $K(0.125)$ equals one, since $G(0.125) = G$ is a connected network. Figure 6.7 shows how β_0 decreases throughout the filtration.



(a)

w*	Δm
1	47
0.83	20
0.583	25
0.5	173
0.33	194
0.25	148
0.2	54
0.167	58
0.143	47

(b)

Figure 6.7.: (a) The number of connected components (β_0) of the weighted clique rank filtration of the network science network is plotted in red. The size of the largest connected component is plotted in blue. Big changes in β_0 and the size of the largest connected component coincide with large increases in the number of edges in the filtration. The nine filtration values where the number of edges increases by more than ten are indicated by grey lines. (b) The filtration values at which the number of edges increases by more than ten, and the corresponding number, Δm, of additional edges.

The clique complex $K(w^*)$ stays disconnected as long as $w^* > {}^1\!/_7 \approx 0.143$. Edges with this weight are part of 8-cliques. Figure 6.6 highlights the vertices that join the largest connected component at this step in the filtration, different colours correspond to different connected components prior to adding the 47 edges with weight equal to 0.143[3]. The four vertices at the top form an 8-clique with some of the black vertices that were already part of the largest connected component. Similarly, on the bottom left, one vertex from the yellow component together with four single vertices and three black vertices, already part of the largest connected component, form an 8-clique. Before

[3] The two 8-cliques have 56 edges in total. Nine edges have higher weight than 0.143 due to other collaborations, and hence are already present in the filtration.

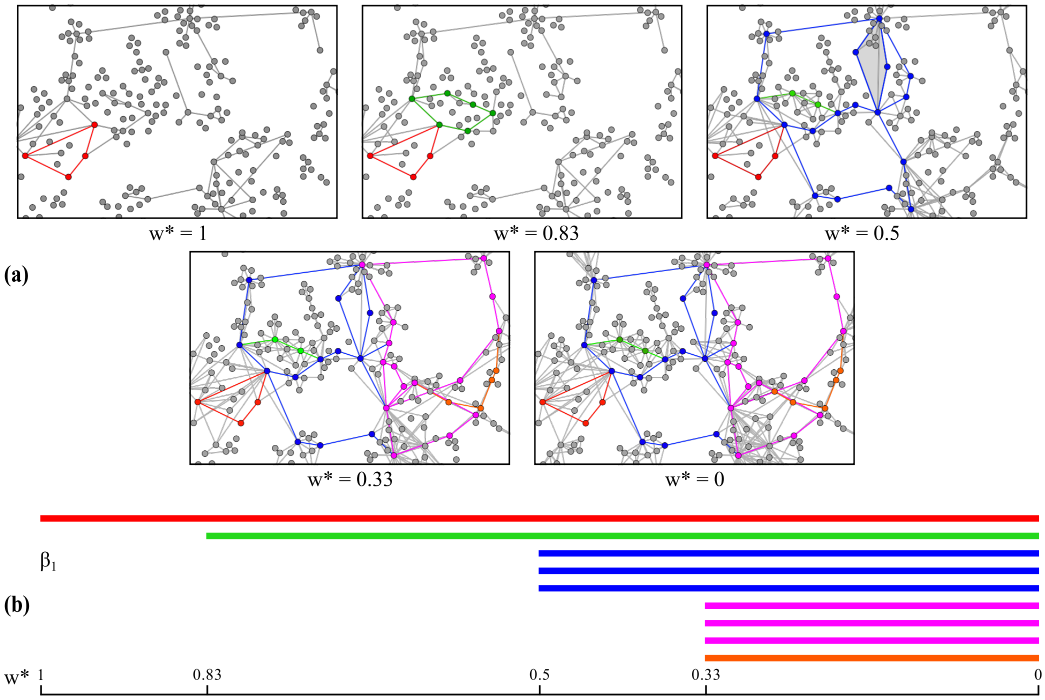
adding these edges there are ten connected components, out of which eight consist of just a single vertex. The authors corresponding to these vertices appear visually to be in the periphery of the network.

Figure 6.7(a) shows the size of the largest connected component throughout the filtration. It increases quite slowly.

Next we inspect the persistent homology in dimension 1. In Figure 6.8 we illustrate the final stages of the weighted clique rank filtration. This is the only part of the filtration where the first Betti number is non-zero. We show the correspondence between the cycles in the complex and the persistence barcode. There seems to be a tendency for larger cycles to be born at lower threshold weights, as illustrated in Figure 6.8(c). However, the average weight of the edges that form cycles is 1.13, much higher than the average edge weight for the whole network, which is 0.53.

Notice how all of the cycles that were born persisted to the end of the filtration. It would have been possible for a cycle to die. For instance if the four scientists (A. Vazquez, A. Vespignani, A. Barrat, M. Weigt) appearing in the red cycle found at $w^* = 1$ were to collaborate on a paper, there would be diagonal edges appearing at $w^* = 0.33$ which would kill the cycle. The persistent homology of the clique complex of this network is trivial in dimensions higher than 1.

We expected to see all cycles appear at the end of the filtration, when w^* is close to zero. Instead, we found exactly the opposite to be true. The edges that generate cycles in collaboration networks are generally edges of high weight and the cliques in the network consist of edges with low weights. This initially puzzling result can be explained as a side effect of the construction of the network as the one-mode projection of a bipartite network. Edges of low weight are *necessarily* part of cliques, and the lower the weight the larger the clique, since low weights correspond to large collaborations. Furthermore, bridges have relatively high weights: the fact that the nodes incident to a bridge have no shared neighbours implies a collaboration of just the two authors, and hence a minimal edge weight of 1. This result points out that collaboration networks are significantly different from social networks in terms of their weight distribution.

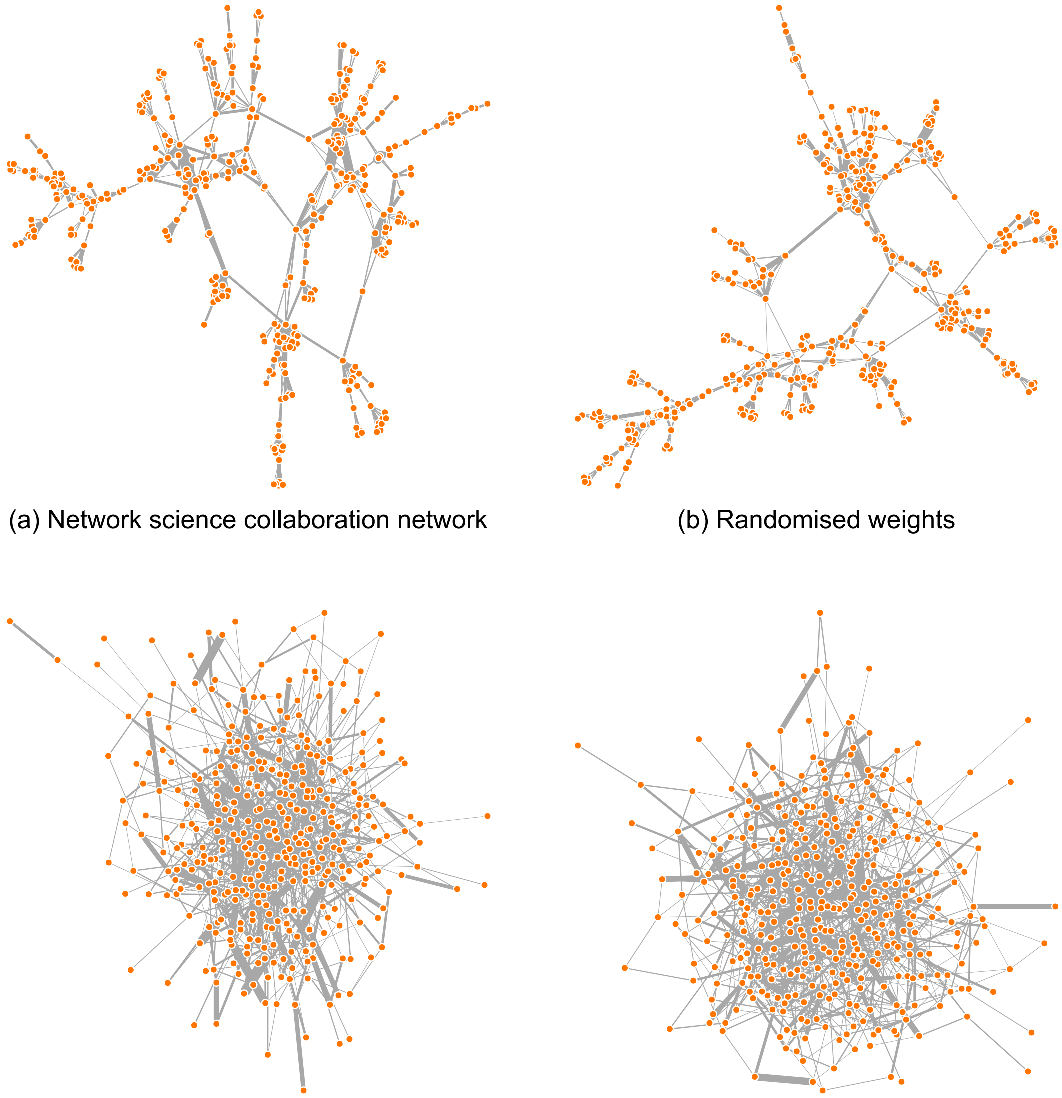


(c)

w*	1	0.83	0.5	0.33
length of cycles	4	6	5,9,10	4,5,8,9

Figure 6.8.: (a) The central part of the network science collaboration network where all cycles occur. The first and smallest cycles appears at threshold weight $w^* = 1$. Further along the filtration several other cycles appear. The two shaded triangles for $w^* = 0.5$ indicate that there is no cycle there. Three blue cycles appear at $w^* = 0.5$. (b) The persistence 1-dimensional barcode of the weighted clique rank filtration. (c) The length of the cycles that are born at each filtration value.

We now focus on a more quantitative analysis of the persistent homology of the network science network. We compare the zero and one dimensional persistent homology of the network to three sets of 1000 randomised versions of the network. The first set is most similar to the original network, it is structurally equal, but edge weights are randomly shuffled. The second set of random networks is obtained using the switching model (Definition 2.2.3); the node degrees are fixed but the edges are randomised. Finally the last set of networks are $G(n, m)$ Erdős-Rényi random networks (Definition 1.1.20). Figure 6.9 illustrates the difference between the three different random network classes.



(a) Network science collaboration network

(b) Randomised weights

(c) Switching model

(d) Erdös-Rényi model

Figure 6.9.: Illustration of the different random network classes. Edge thickness corresponds to edge weight. (a) The original network G. (b) A random network with the same structure as G but where the edge weights are randomised. (c) A random network produced with the switching model, and hence with equal degree sequence as G, but otherwise random edge placement. (d) A random network produced with the $G(n, m)$ Erdős-Rényi model, where $n = 397$ and $m = 914$. The number of nodes and edges is the same as for G, but edges are placed randomly.

Using the L^1 distance between persistence landscapes (see Section 1.5.4) we found that the average pairwise distance between the 0-dimensional landscape of the original network and the random networks was 85.86 (sd 5.47), 99.52 (sd 4.03), and 134.54 (sd 12.33) for the weight reshuffled, switching model randomised and Erdős-Rényi random networks respectively. As to be expected, on average the distance from the original network to the random networks with just the weights reshuffled is smaller than the distance from the original network to either the random networks produced by the switching model or Erdős-Rényi model.

The average pairwise distance between the 0-dimensional landscapes of the 1000 networks within each of the random network classes was 9.53 (sd 3.79), 6.99 (sd 2.90) and 17.09 (sd 10.31) for the weight reshuffled, switching model randomised and Erdős-Rényi random networks respectively. Notice that the average distance between networks within classes is much smaller than the average distance from the original network to the randomised networks. Hence, we can definitely use the 0-dimensional persistent homology to distinguish the real network from the three classes of random networks.

Even though at the start and end of the filtration these networks are (nearly) identical in terms of their number of connected components, persistent homology still detects structural differences by tracking connected components throughout the weighted filtrations.

We investigated if the first Betti numbers give us further power to distinguish between the collaboration network and the three classes of random networks. We found that in general, for all sets of random networks the number of 1-dimensional homology classes is much higher throughout the filtration. We found averages of 65.8 (s.d. 6.9), 486.5 (s.d. 6.7) and 520.6 (s.d. 4.6) classes for the weight reshuffled, switching model randomised and Erdős-Rényi random networks respectively. Recall, there are just 9 1-dimensional homology classes in the real network.

We can explain the observed higher number of 1-dimensional homology classes in the random networks as follows. Firstly, we discuss the structurally equal, but weight reshuffled networks. Just like the original network, at the end of the filtration there are 9 1-dimensional homology classes corresponding to the 9 cycles illustrated in Figure 6.8. However, there are several other 1-dimensional homology classes that are born and die

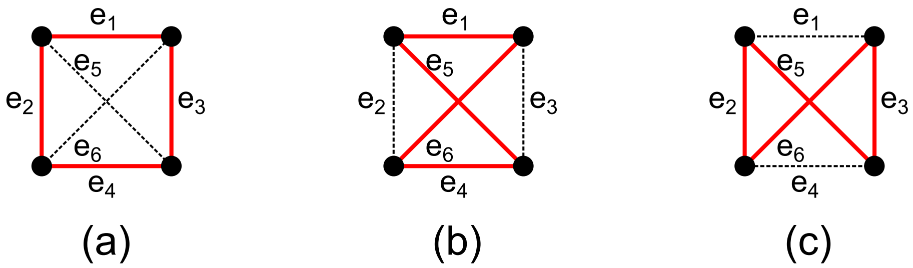


Figure 6.10.: Illustration of the three potential cycles in a 4-clique. (a) In order for the edges e_1, e_2, e_3, e_4 to correspond to a 1-dimensional homology class in the weighted clique rank filtration, the minimum weights of these edges, $\min(w(e_1), w(e_2), w(e_3), w(e_4))$, needs to be higher than the maximum weights of the remaining edges $\max(w(e_5), w(e_6))$. (b) Similarly $\min(w(e_1), w(e_4), w(e_5), w(e_6)) > \max(w(e_2), w(e_3))$ needs to hold for the edges e_1, e_4, e_5, e_6 to correspond to a 1-dimensional homology class in the weighted clique rank filtration. (c) Finally $\min(w(e_2), w(e_3), w(e_5), w(e_6)) > \max(w(e_1), w(e_4))$ needs to hold for the edges e_2, e_3, e_5, e_6 to correspond to a 1-dimensional homology class in the weighted clique rank filtration.

throughout the filtration. These cycles appear as part of densely connected areas of the network, such as cliques[4]. As an example, Figure 6.10 shows the three potential 4-cycles in a 4-clique and the conditions on the edge weights in the clique, for these to appear as 1-dimensional homology classes in the weighted clique rank filtration. Notice that these conditions are more likely to be satisfied when there is more variety of weights in the clique, that is when edges have different weights. We measure the variety of weights within cliques as follows: let σ_w^C be the fraction of unique weights in clique $C = (V_C, E_C)$, i.e. $\sigma_w^C = {}^{|w(E_C)|}/_{|E_C|}$. Let $\sigma_w^{clique}(G)$ be the average of these fractions for all complete cliques on four or more nodes in a network G. For the original network σ_w^{clique} equals 0.41 whereas for the 1000 weight reshuffled networks σ_w^{clique} is significantly larger, on average it equals 0.69 (s.d. 0.014). This greater variety of weights in the cliques probably causes the greater number of 1-dimensional homology classes throughout the filtration of the weight randomised networks.

Secondly, we discuss the two structurally different random network classes, as generated

[4] Recall that by construction there are many cliques in the one-mode projection of a bipartite network.

by the switching model and the Erdős-Rényi model. Visually (see Figure 6.9(c)(d)), these networks have a very different structure from the original network. It appears that they have many non-trivial cycles even when all edges are present. Indeed we find high first Betti numbers of the clique complexes of the random networks at $w^* = 0$. That is on average, β_1 equals 482.0 (s.d. 7.6) and 520.3 (s.d. 4.6) for the networks generated by the switching model and Erdős-Rényi random network model respectively[5]. This large difference in terms of 1-dimensional homology classes at the end of the filtration explains the large difference between the persistent homology of the weighted filtrations.

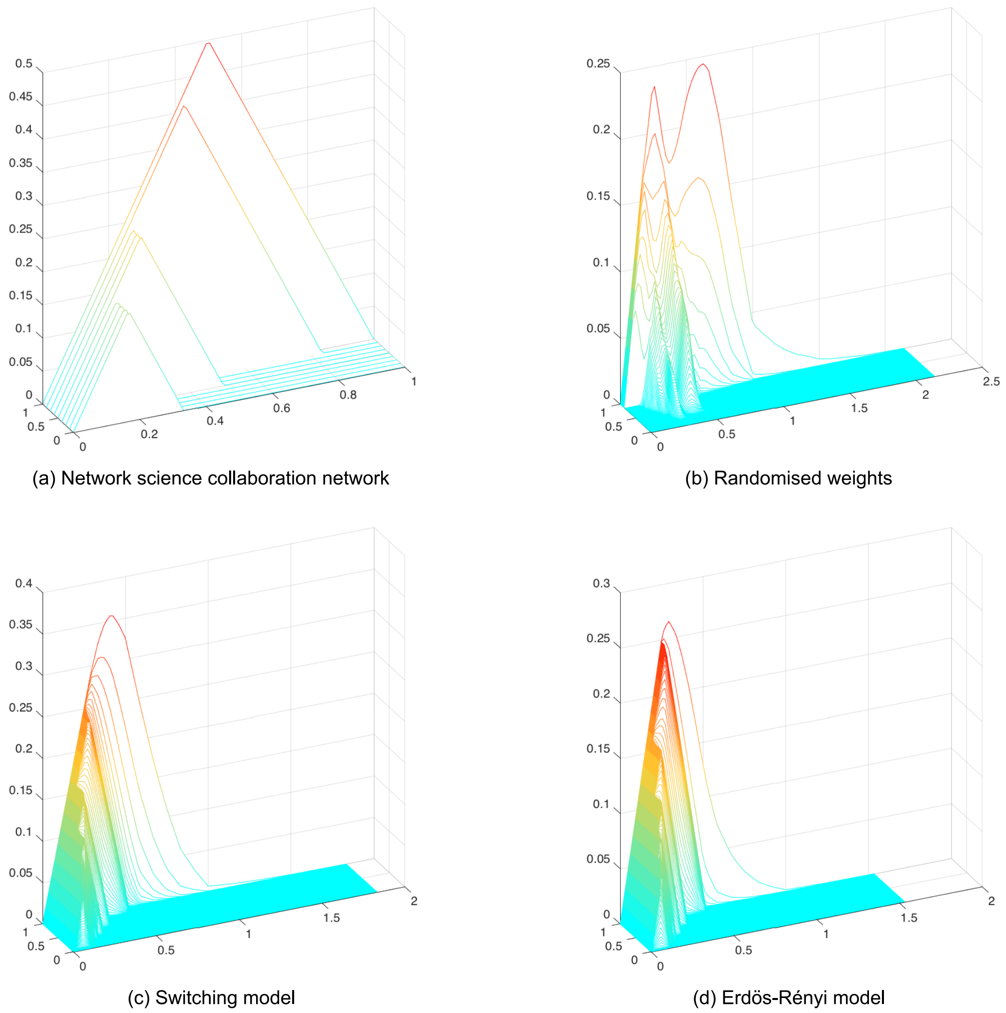
Using the L_1 distance between the persistence landscapes, we found that the average pairwise distance between the 1-dimensional landscape of the original networks and the random networks was 0.67 (s.d. 0.32), 12.07 (s.d 0.32), and 11.99 (s.d. 0.22) for the weight reshuffled, switching model randomised and Erdős-Rényi random networks respectively. Again, on average the distance from the original network to the random networks with just the weights reshuffled is smaller than the distance from the original network to either the random networks produced by the switching model or Erdős-Rényi model. The distance to the networks produced by the switching model and Erdős-Rényi model is very similar.

The average pairwise distance between the 1-dimensional persistence landscapes of networks within each of the random network classes was 0.39 (sd 0.15), 0.70 (sd 0.28) and 0.41 (sd 0.19) for the weight reshuffled, switching model randomised and Erdős-Rényi random networks respectively. Notice that it is difficult to tell the original network from the weight reshuffled networks using the 1-dimensional persistence landscape. For the other two classes of random networks, the average distance between networks within each class is much smaller than the average distance from the original network to the randomised networks. Hence, we can use the 1-dimensional persistent homology to distinguish the real network from random networks produced by the switching model or Erdős-Rényi model.

Figure 6.11 illustrates the average 1-dimensional landscape for each of the random net-

[5] Notice that these averages differ from the average number of intervals appearing throughout the filtration that we presented earlier. We now only measure the Betti numbers at $w^* = 0$

work classes, as well as that for the real network.



(a) Network science collaboration network

(b) Randomised weights

(c) Switching model

(d) Erdös-Rényi model

Figure 6.11.: (a) The 1-dimensional persistence landscape of the original network science network. (b) The average 1-dimensional persistence landscape of 1000 networks with randomised weights. (c) The average 1-dimensional persistence landscape of 1000 networks randomised with the switching model. (d) The average 1-dimensional persistence landscape of 1000 Erdős-Rényi random networks.

6.5.2. Physics collaboration networks

In this section we analyse three larger collaboration networks (Appendices A.1, A.4 and A.6). Again we restrict our attention to the largest connected component of each network. Table 6.2 lists the number of nodes and edges for the largest connected component of these networks. We computed the persistent barcodes in dimensions zero, one and two. In all three cases, we found that β_0 stayed high for the largest part of the filtration and then quickly decreased to 1 at the end of the filtration. In each case, the smallest weight was needed to create the single largest connected component (see Table 6.2). This behaviour differs slightly from that of the network scientist collaboration network, since in that case, the connected component formed before reaching the end of the filtration.

Network	n	m	w^*_{con}	$\lvert E_{w^*_{con}}\rvert$	$\lvert E_{<w^*_{con}}\rvert$
Network Science	379	914	0.143	47	10
Condensed Matter	36458	171735	0.034	315	0
High-energy Theory	5835	13815	0.056	171	0
Astrophysics	14845	119652	0.018	357	0

Table 6.2.: Properties of the four collaboration networks. For each network we have indicated the number of nodes n, the number of edges m, the threshold value w^*_{con} at which β_0 becomes equal to 1 in the weighted clique rank filtration, the number of edges with weight equal to this threshold value $|E_{w^*_{con}}|$, and the number of edges with weight smaller than this threshold value $|E_{<w^*_{con}}|$.

We would again like to compare the persistent homology of these collaboration networks to that of random networks. However, the persistent homology computations for these larger networks makes it infeasible to compute the persistent homology of 1000 random networks numerically. Instead, we use theoretical results to show that the persistent homology of Erdős-Rényi random networks is very different from that of these collaboration networks.

Let $G(n, p)$ be an Erdős-Rényi random network with n nodes and p the probability of an edge being present. Erdös and Réyni showed that if $p \gg \log n/n$ then $G(n,p)$ is almost always connected [39]. In [63], Kahle shows that there are analogous results for higher dimensional connectivity of the clique complexes of these random networks. In particular, if we define α by $p = n^{\alpha}$, Kahle shows that the k-th homology group of a clique complex of an Erdős-Rényi random network is almost always zero if α is outside the interval $(\frac{-1}{k}, \frac{-1}{2k+1})$, that is $(-1, -1/3)$ for $k = 1$ and $(-1/2, -1/5)$ for $k = 2$.

A filtration of a random network corresponds to increasing p over time, or increasing α from $-\infty$ to $\log_n p$. For the clique complex of a random network $G(n,p)$ we expect the second Betti number to be zero for $\alpha < -0.5$. In Table 6.3 the values α for the three collaboration networks can be found. For all three networks the value of α satisfies this condition. In fact, we find a large number of intervals for both the condensed matter network and the astrophysics network. This clearly distinguishes these networks from Erdős-Rényi random networks.

Network	**total** β_1	**total** β_2	p	α
Condensed Matter	11361	274	0.00026	-0.79
High-energy Theory	1389	2	0.00081	-0.82
Astrophysics	4879	222	0.0011	-0.71

Table 6.3.: This table lists the total number of intervals that appear throughout the weighted clique rank filtration in dimensions one and two (total β_1 and β_2) of the three physics collaboration networks. It also shows the edge density $p = 2m/n(n-1)$ of the networks and the value α for which $p = n^{\alpha}$.

6.6. Conclusion

Even though persistent homology was developed for the analysis of point clouds, it is quite natural to use it in the analysis of networks. Since a network is itself a simplicial complex, persistent homology can be used to analyse any network filtration. There are several constructions available to enrich a network's structure by building a higher dimensional simplicial complex from it. The clique complex of a network is basically equivalent to the Vietoris-Rips complex of a point cloud (for a certain radius ϵ). Using Zachary's karate club as an illustrative example, we discussed in detail the connectivity properties that can be extracted from the Betti numbers of the 0, 1 and 2-dimensional skeleta of the clique complex of a network.

The labelled complex of a bipartite network offers a different way to construct a simplicial complex from a (bipartite) network. This simplicial complex is particularly interesting since it enriches the commonly used one-mode projection of a bipartite network, and retains more of the information present in the bipartite network.

The idea of representing relational data as a simplicial complex instead of a network seems to be gaining popularity. We discussed several recent works in the literature where this is done. It would be interesting to use persistent homology and the weighted labelled complex filtration, to analyse bipartite networks. This is left as future work.

We showed that persistent homology is a versatile tool for network analysis, as it can be used for several classes of networks. Specifically, it is very suited to the analysis of weighted networks, evolving networks and bipartite networks.

We presented a case study where we used persistent homology to analyse several weighted collaboration networks. By inspecting the 1-dimensional homology classes and their generators we discovered that the weight distribution in these networks is very different from those in Granovetter's weak tie theory. We could explain how the construction of the analysed networks, as a one-mode projection of a bipartite network, caused the weight distribution to differ from that in Granovetter's theory. This result indicates that collaboration networks with the given weight assignment may not be as good a proxy for social networks as previously believed [91].

We used JavaPlex to compute the persistent homology of networks. There are faster implementations of the persistent homology algorithm available that we have not used. These software packages would likely make it possible to compute the persistent homology of randomised versions of the larger collaboration networks. This is left as future work.

Persistent homology is a valuable measure of the structure of a network, that can be used in addition to more traditional network measures.

7. Topological data analysis of lipid formulations

In this chapter we describe our work on the classification of lipid formulations using topological data analysis. This work is related to the techniques introduced in Section 1.5 and used in Chapter 6. In this chapter we analyse the persistent homology of spatial point clouds. To do so, we convert spatial point clouds to spatial networks and higher dimensional simplicial complexes, using the Vietoris-Rips construction.

The work discussed in this chapter has been done in collaboration with Dr. Dallas Warren (Pharmaceutical Science, Monash University) and Prof. Craig Westerland (Mathematics, University of Minneapolis). The problem that we address originates in pharmaceutical science: we investigate the potential of persistent homology to classify different phases in chemical systems. This problem was suggested by Dr. Dallas Warren. The experiments discussed in this chapter were designed in collaboration with both Dr. Dallas Warren and Prof. Craig Westerland. I was responsible for all the programming described in this chapter. Furthermore I did all the data analysis described in Sections 7.3 and 7.4 and part of the data analysis in Section 7.5.

We initially use persistent homology as an exploratory data analysis tool. In order to do this I wrote HomViz, a graphical user interface for JavaPlex. This software allows us to visualize 3-dimensional point-clouds and their associated Vietoris-Rips complex as well as the corresponding persistence barcode or diagram. We also use HomViz to highlight the generators of 1-dimensional homology classes.

In our exploratory data analysis, we discuss the persistent homology of four different lipid formulations in detail. These four systems were each chosen to represent a distinct phase behaviour of lipid formulations. We investigate how topological and geometrical properties, as measured by persistent homology, can help to classify systems from these four different phases.

All lipid formulations that we study have previously been assigned a phase by experts. We show that we obtain reasonable classification results, by assigning the remaining lipid formulations to the class of the representative that they are most similar to, in terms of their persistent homology.

Furthermore, so as not to be dependent on the choice of representatives, we show that we get similar results by using hierarchical clustering of the lipid formulations based on their persistent homology. Our results indicate that persistent homology can distinguish fairly well between different phase behaviours.

Ultimately our goal is to label lipid formulations automatically. This will assist practitioners in assigning the correct phase to each system. A better understanding of the phase behaviour of lipid formulations will help recognize the conditions for which a drug is absorbed best.

This chapter is organised as follows. Section 7.1 sets the problem that we address in context in the literature. In Section 7.2 we describe the data sets that we have analysed. Section 7.3 presents results of our exploratory data analysis. In Section 7.4 we present classification results of a simple classifier. Section 7.5 presents classification results based on hierarchical clustering. Finally in Section 7.6 we discuss our results and point out directions for future research.

My contributions to the literature this chapter are

- Introducing a novel approach to phase recognition of lipid formulations (paper in draft), in collaboration with Dr. Dallas Warren and Prof. Craig Westerland.

- Developing the software HomViz (see Appendix B.1.2), a graphical user interface for JavaPlex.

- Introducing periodic boundary conditions to algorithms for persistent homology.

7.1. Context and problem description

There is a growing number of drugs that are poorly soluble in water [34]. Such drugs are typically inefficiently absorbed by the human body, i.e. these drugs have low *bioavailability*. A multitude of methods has been developed in attempts to mitigate this problem [125], one of which is to dissolve the drug within a lipid formulation [102]. In doing so, the bioavailability of the drug can be improved [125].

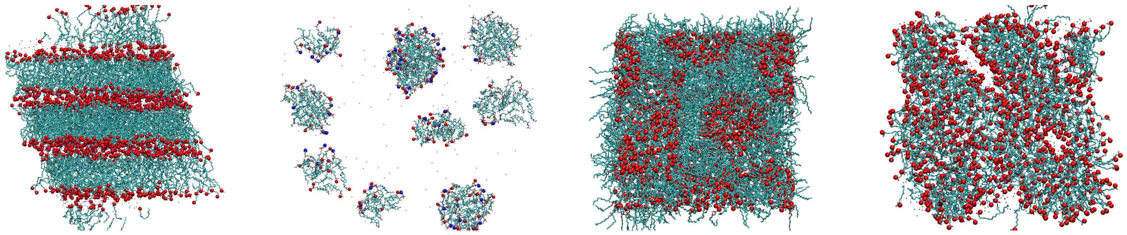

(a) Lamellar (b) Micellar (c) Phase Separated (d) Wormy Micellar

Figure 7.1.: Example output of molecular dynamics simulations of lipid formulations. Each figure is representative of specific phase behaviour: lamellar, micellar, phase separated and wormy micellar. The atoms are coloured by type: cyan = carbon, red = oxygen and blue = nitrogen (nitrogen appears in (b) only). Water has been omitted for clarity.

Lipid formulations generally consist of a drug dissolved in two or more *excipients*, inactive substances that serve as a vehicle for the drug [102]. These excipients are usually surfactants: molecules that comprise of a hydrophobic (water hating) and hydrophilic (water loving) part, see Figure 7.2. As a result, surfactants can form complex structures in the presence of water. The structure formed depends on the molecular shapes, concentration and composition of the surfactants [72, 104]. Figure 7.1 illustrates four lipid formulations that form quite diverse structures.

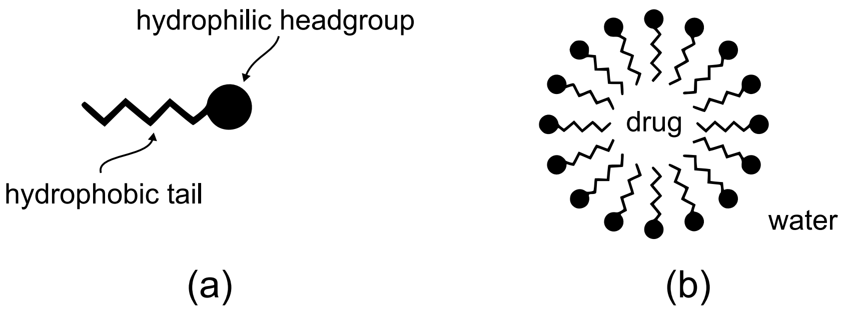


Figure 7.2.: (a) Schematic representation of a surfactant, with a hydrophilic (water loving) headgroup, and hydrophobic (water hating) tail. (b) Schematic representation of a micelle. The headgroups of the surfactants face the water molecules, whereas the hydrophobic tails point away from the water, inside the micelle.

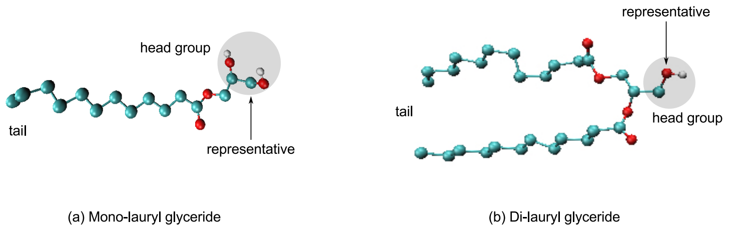
The structure formed by the excipients is meant to ensure that the drug is in a dissolved state. The effectiveness of a lipid formulation for drug-delivery purposes depends on its behaviour when mixing with the aqueous environment in the gut. This behaviour can be predicted using molecular dynamics simulations with increasing amounts of water.

Molecular dynamics simulation is an important tool used to study the aqueous phase behaviour of complex systems. It is increasingly being used for predicting the aqueous phase behaviour of lipid-based drug formulations [8]. Several distinct liquid phases have been observed and studied using molecular dynamics [14, 68, 119, 9, 58].

The current method for determining the phases formed within molecular dynamics simulations is simply to visualise the coordinate files manually and make a judgement on what type of liquid phase it appears to resemble. To observe the geometric structure, the atoms are usually plotted with varying radii. This process is both labour intensive and can be biased, due to the results expected by the observer. The classification of liquid phases would benefit from a more quantitative, automated and independent method. We believe persistent homology is well suited to this classification problem, as it is a robust measure of the structure underlying a point cloud at varying scales.

7.2. Lipid formulation data set

From a mathematical perspective, the outcome of a molecular dynamics simulation is a 3-dimensional labelled point cloud in a cube (homeomorphic to the unit cube I^3). The molecular dynamics simulations use periodic boundary conditions to approximate a large system, by simulating just a small part: the unit cell. That is, the left and right side, top and bottom, and front and back side of the cube are identified. In effect, the point cloud lives in a space homeomorphic to the 3-torus T^3.



(a) Mono-lauryl glyceride

(b) Di-lauryl glyceride

Figure 7.3.: Examples of surfactant molecules, consisting of hydrophobic and hydrophilic regions, and atoms from the hydrophilic head group selected for analysis. Atom colouring; cyan = carbon, red = oxygen, white = polar hydrogens, non-polar hydrogens have been omitted.

Each point corresponds to an atom and is labelled by the atom type and atom index as well as the corresponding molecule type and molecule index. Hence, for each atom, we know what kind of atom it is (e.g. carbon, hydrogen or oxygen), which molecule it is part of and what kind of molecule that is. Furthermore, the atom index provides us with the 'location' of the atom within the molecule, e.g. whether it is part of the head group of a surfactant or of the tail (see Figure 7.2).

We analysed a total of 46 lipid formulations. A detailed description of the data can be found in Appendix D. For each system, we extracted two point clouds, one to represent the surfaces formed by the head groups of surfactants and another to represent the location of the water in the system. The former point cloud was obtained by selecting a specific atom from the head group of each surfactant in the systems. Figure 7.3 illustrates the selected atom for two different surfactants. The latter point cloud was obtained by selecting the oxygen atom of each water molecule in the system. The sizes of the resulting point clouds are listed in Table 7.1.

#	N_{surf}	N_{water}	#	N_{surf}	N_{water}	#	N_{surf}	N_{water}	#	N_{surf}	N_{water}
1	1200	24,687	13	2152	9,075	25	136	31,339	37	1344	16,560
2	1200	73,110	14	1568	3,312	26	96	31,272	38	1072	19,872
3	567	89,100	15	1368	13,248	27	1200	6,093	39	1389	13,200
4	338	32,349	16	808	22,910	28	1200	10,444	40	1157	16,500
5	2416	6,624	17	536	26,079	29	1200	16,24	41	1042	18,160
6	1880	6,624	18	272	29,466	30	1200	246	42	1075	14,850
7	1608	9,936	19	347	28,050	31	1200	1,283	43	800	18,096
8	2083	3,312	20	400	26,496	32	1200	2,708	44	984	16,560
9	2416	9,936	21	200	29,360	33	1200	4,301	45	584	23,184
10	1678	13,248	22	267	105,806	34	250	89,100	46	1509	89,937
11	1760	3,300	23	502	100,073	35	1964	86,149			
12	1272	6,600	24	300	105,806	36	1472	14,850			

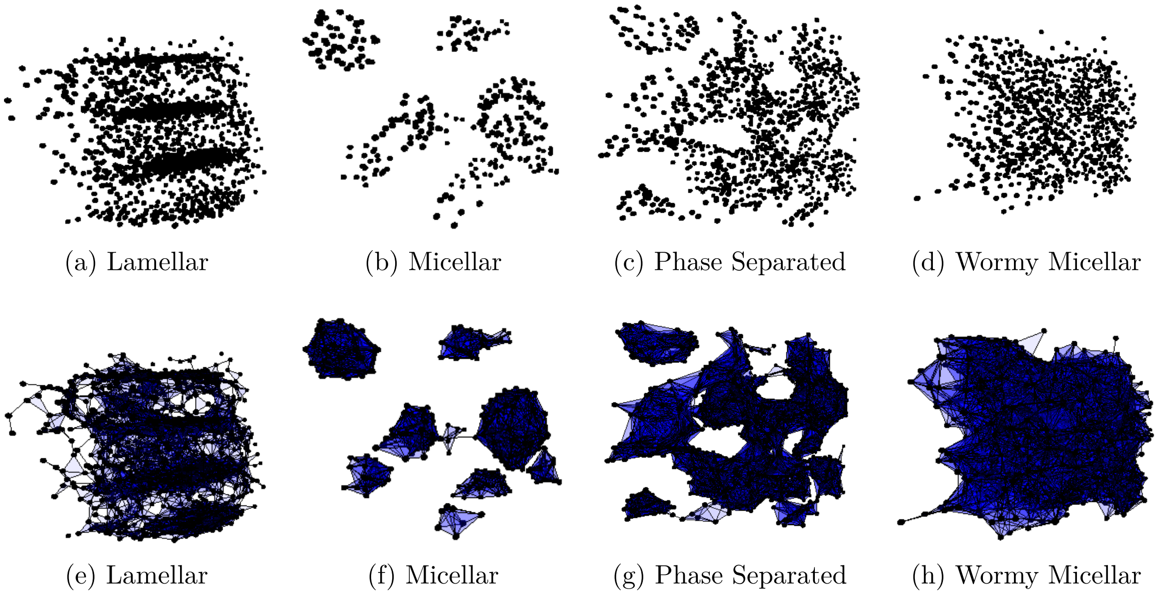
Table 7.1.: Number of points in the surfactant N_{surf} point cloud and the water N_{water} point cloud for each of the 46 systems. The number of points corresponds to the number of surfactant molecules or water molecules in the lipid formulation. The highlighted lines correspond to systems that we chose as representatives of the four different phases: lamellar (red), micellar (yellow), phase separated (green) and wormy micellar (blue). The greyed out systems (1, 2, 3, 4, 26, 27, 28, 29 and 34) were not used in our classification experiments (Sections 7.3 and 7.5).

Clearly, there is great variability in the sizes of these point clouds, which is one of the main challenges in analysing this data set. For the water point clouds, we analyse a random sub-sample of 1000 points (except for system 30), to mitigate this. At this stage, we use the whole head group point clouds as they are listed in Table 7.1.

The 46 systems had previously been labelled by pharmaceutical scientists as specific phases. These phases were grouped into four coarser categories. Table D.1 in Appendix D provides more details on this classification. Figures D.1, D.2, D.3 and D.4 illustrate all 46 systems.

7.3. Analysing four distinct phases

In this section we discuss an exploratory data analysis of four systems from our data set. We selected a representative for each of the four categories of systems: lamellar, micellar, phase separated and wormy micellar. We discuss the results of analysing these systems by using the (3-skeleton of the) Vietoris-Rips filtration (Definition 1.5.14) and its persistent homology. Figure 7.1 illustrates the four representatives: system 7 represents lamellar, system 24 represents micellar, system 32 represents phase separated and system 38 represents wormy micellar.


(a) Lamellar (b) Micellar (c) Phase Separated (d) Wormy Micellar

(e) Lamellar (f) Micellar (g) Phase Separated (h) Wormy Micellar

Figure 7.4.: (a)-(d) Point clouds extracted for the surfactant headgroups in systems 7, 24, 32 and 37. These are the same systems as illustrated in Figure 7.1 and as highlighted in Table 7.1. These systems look different from the systems in Figure 7.1 because the 3D system is plotted from a different angle. (e)-(h) The 2-skeletons of the Vietoris-Rips complexes for each pointcloud at filtration values 1.3, 2.4, 1.4 and 1.9 respectively.

We first focus on the head group point clouds of these systems (see Figure 7.4(a)-(d)). For each of these point clouds, we used HomViz (see Appendix B.1.2) to manually decide on a suitable maximum filtration value f_{max}. In our computations we actually use the 3-skeleton of the Vietoris-Rips complex, $V(X, f_{max})$. In Figure 7.4(e)-(h) we plot the

2-skeleton for simplicity.

There are two reasons for choosing the maximum filtration values manually. The first reason is a practical one. It is computationally infeasible to use high filtration value for persistent homology computations[1]. The reason for this, is that the number of simplices grows very rapidly when increasing the maximum filtration value, especially for higher dimensional simplices. That is, for n points that are within distance ϵ from one another, the number of k-simplices equals $\binom{n}{k+1}$. Running computations with high filtration values causes out of memory exceptions.

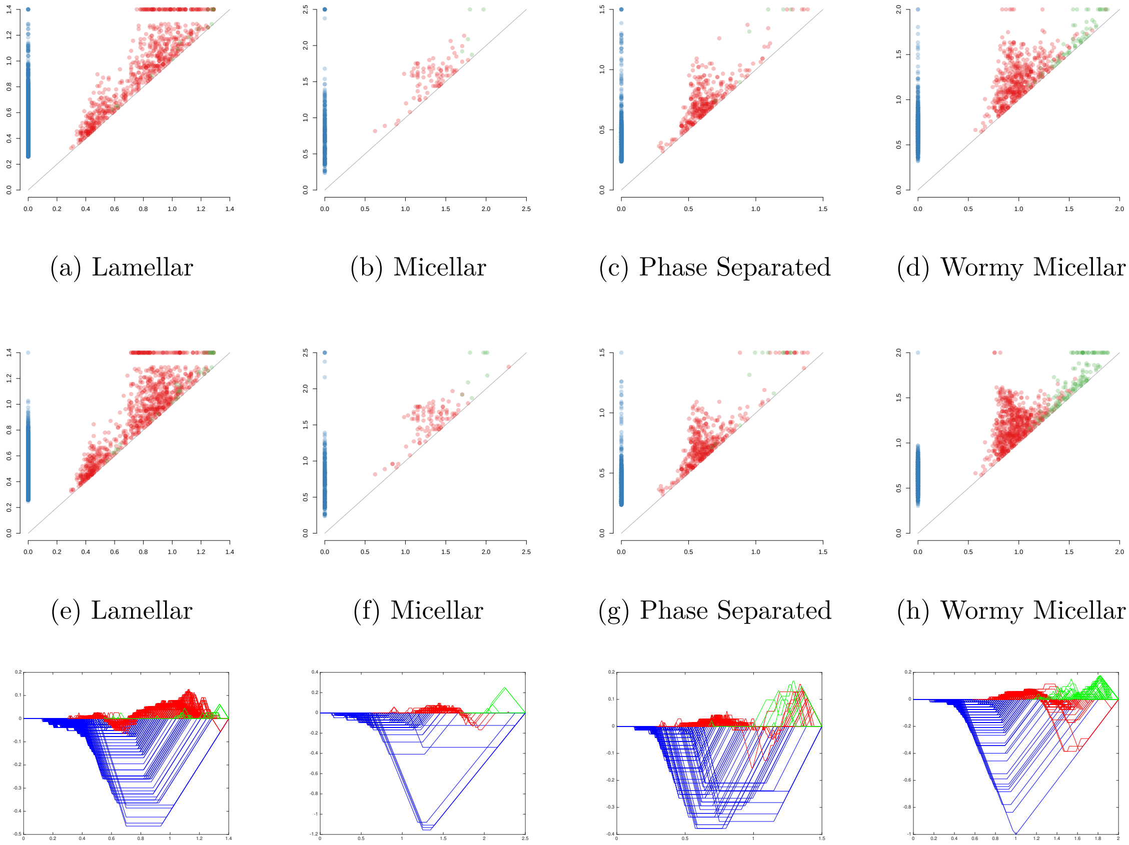
The second reason for us to limit the maximum filtration value is that our main interest is the surfaces formed by the surfactant head groups. The surfactant molecules are a given size, hence increasing the threshold too far above this scale does not reflect the geometry or topology of the physical system.

The lipid formulations are modelled in boxes with periodic boundary conditions. The molecules near the boundary of this box interact with the molecules on the opposing side of the box. We introduce these periodic boundary conditions in the computation of persistent homology. To do so we extended the JavaPlex code base, such that the abstract Vietoris-Rips complex $V(X, \epsilon)$ includes simplices that span the boundary of the box (see Appendix B.1.3).

Figures 7.5(a)-(d) show the persistence diagrams of these four systems. Figures 7.5(e)-(h) show the persistence diagrams of these systems, but this time taking into account periodic boundary conditions. Finally Figure 7.5(i)-(l) shows the landscape difference between the persistence landscapes with periodic boundary conditions and without periodic boundary conditions.

Let us first discuss the lamellar system and its persistence diagrams. This system is characterised by several box-spanning layers formed by the surfactants. In this specific system we see four distinct layers. The layers are thick, in fact they are double layers, formed by head groups with their tails pointing in both directions away from the double

[1] There are faster implementations available of the persistent homology algorithm. We have not yet attempted to use these.



(a) Lamellar (b) Micellar (c) Phase Separated (d) Wormy Micellar

(e) Lamellar (f) Micellar (g) Phase Separated (h) Wormy Micellar

(i) Lamellar (j) Micellar (k) Phase Separated (l) Wormy Micellar

Figure 7.5.: (a)-(d) Persistence diagrams of the surfactant point clouds for systems 7, 24, 32 and 37. Blue points correspond to 0-dimensional homology classes, red point to 1-dimensional homology classes and green points to 2-dimensional homology classes. (e)-(h) Persistence diagrams of the same four systems, with periodic boundary conditions. (i)-(l) Persistence landscapes of the *difference* between the persistence landscape with periodic and without periodic boundary conditions.

layer (see Figure 7.1(a)). There are water molecules trapped in between these double layers. If there were no noise in this system, we expect the homology classes at the maximum filtration value to be fairly trivial: a box-spanning layer is contractible, and thus has Betti numbers $\beta_0 = 1$, $\beta_1 = 0$ and $\beta_2 = 0$. If we take into account periodic boundary conditions however, then the homology classes are a bit more interesting, since in this case a box-spanning layer becomes topologically equivalent to a torus. Hence the corresponding Betti numbers would be $\beta_0 = 1$, $\beta_1 = 2$ and $\beta_2 = 1$. Thus for the ideal lamellar system with k layers, we would expect to see long-lived homology classes with Betti numbers $\beta_0 = k$, $\beta_1 = 0$, and $\beta_2 = 0$ for the non-periodic case, and Betti numbers $\beta_0 = k$, $\beta_1 = 2k$ and $\beta_2 = k$ for the periodic case.

System 7 displays more noise than an ideal lamellar phase. It hence does not show the expected profile for either non-periodic or periodic boundary conditions. However, Figure 7.5(a),(e) and (i) show that there are more long-lived first homology classes (appearing at about 0.7) for homology with periodic boundary conditions as compared to normal homology. This is likely caused by the box-spanning layers, forming cycles through the identified boundaries.

The micellar phases are characterised by a collection of sphere like formations. The head groups of the surfactants form the surface of these spheres and the tails point inwards towards the centres. In an ideal system, a micellar phase with k spheres has Betti numbers $\beta_0 = k$, $\beta_1 = 0$ and $\beta_2 = k$. It is likely for some of the spheres to be 'divided' by the boundaries of the box, and hence we expect to see a higher first Betti number and lower second Betti number when periodic boundary conditions are not being taken into account compared to when they are being taken into account.

System 24 has a quite reasonable profile at the maximum filtration value (see Figure 7.5(b),(f),(j)). The Betti numbers are $\beta_0 = 9$, $\beta_1 = 0$ and $\beta_2 = 2$ without periodic boundary conditions and $\beta_0 = 2$, $\beta_1 = 0$ and $\beta_2 = 4$ with periodic boundary conditions. Visually, it appears that the spheres in this system are of different sizes, which may explain why we only see four 2-dimensional homology classes persist until the end of the filtration. Furthermore, some of the micelles may consist of too few molecules, and hence never form a complete surface (recall from Section 6.3 that the smallest non-trivial 2-dimensional homology class in a Vietoris-Rips complex is the octahedron).

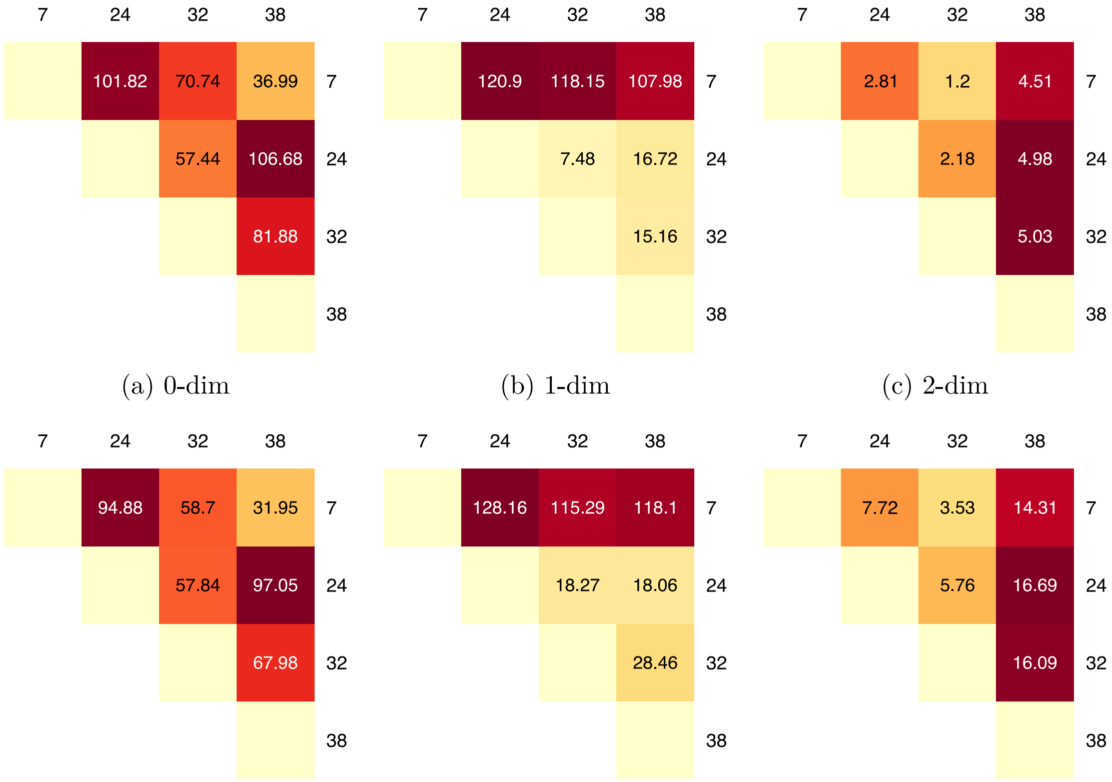
The representative of the phase separated systems that we chose is part of the subcategory of reverse micellar systems. As the name suggests, these systems are characterised by inverted micelles. Again the head groups form spheres, but in this case, the tails of the surfactants point out. The structure formed by the head groups is thus again a collection of spheres and in an ideal system we expect to see exactly the same homology classes as expected for the micellar phase.

Visually, system 32 indeed has a similar persistence diagram to system 24. However, due to the greater number of points, 1200 instead of 300, there is more noise in terms of 0-dimensional and 1-dimensional homology classes near the diagonal. Furthermore, there are some persisting 1-dimensional classes. This is likely due to the higher point density, creating spurious cycles.

The final category consists of wormy micellar systems. These systems are characterized by elongated micelles. Sometimes these are literally stretched out micelles, whereas other times these form cylindrical shapes, stretching across the whole box. In an ideal system, each stretched out micelle would contribute 1 to β_0 and 1 to β_2, just like spherical micelles. The box-spanning cylinders would contribute 1 to β_0 and 1 to β_1 without periodic boundary conditions, and 1 to β_0, 2 to β_1 and 1 to β_2 when periodic boundary conditions are taken into account, since the cylinder then closes up, making it equivalent to a torus.

We see many 2-dimensional homology groups appear in this system. These may be due to the wormy micelles having thicker and thinner parts, where the thinner part forms part of the surface of an internal 2-sphere within the worm. There are many more 2-dimensional homology classes in the periodic persistence diagram than in the non-periodic persistence diagram. This could partially be due to cylinders 'closing up' across the boundary.

Figures 7.6(a)-(f) show heat maps corresponding to the pair-wise L^1 landscape distances between the four systems in dimensions 0,1 and 2. In general, the heat maps are very similar for the periodic and non-periodic cases. From Figure 7.6(a) we can derive that the connectivity of the lamellar (system 7) and wormy micellar (system 38) systems are most similar, followed by the micellar (system 24) and phase separated (system 32) phases. When using periodic boundary conditions (Figure 7.6(d)) the distance between the lamellar and wormy micellar phases decreases slightly, whereas the distance between the micellar and phase separated systems remains roughly equal. Notice that even though, theoretically, we would say that reverse micellar and micellar should have the most sim-



(a) 0-dim

(b) 1-dim

(c) 2-dim

(d) 0-dim periodic

(e) 1-dim periodic

(f) 2-dim periodic

Figure 7.6.: (a)-(c) Heat maps of L^1 landscape distances in dimensions 0,1 and 2, between the persistence landscapes of the systems that we chose as representatives (d)-(f) Heat maps of L^1 landscape distances in dimensions 0,1 and 2, between the persistence landscapes with periodic boundary conditions of the systems that we chose as representatives.

ilar topological information, we do not find that in the 0-dimensional persistence. This is most likely due to the difference in number of points in these two systems, since this information is particularly present in the connectivity (i.e. the 0-dimensional homology) of the Vietoris-Rips complex during the filtration.

The 1-dimensional homology groups demonstrate a big difference between the lamellar phase and the other phases. The reason for this is that there are many 1-dimensional homology classes in the lamellar system that persist, whereas the other systems do not have this feature.

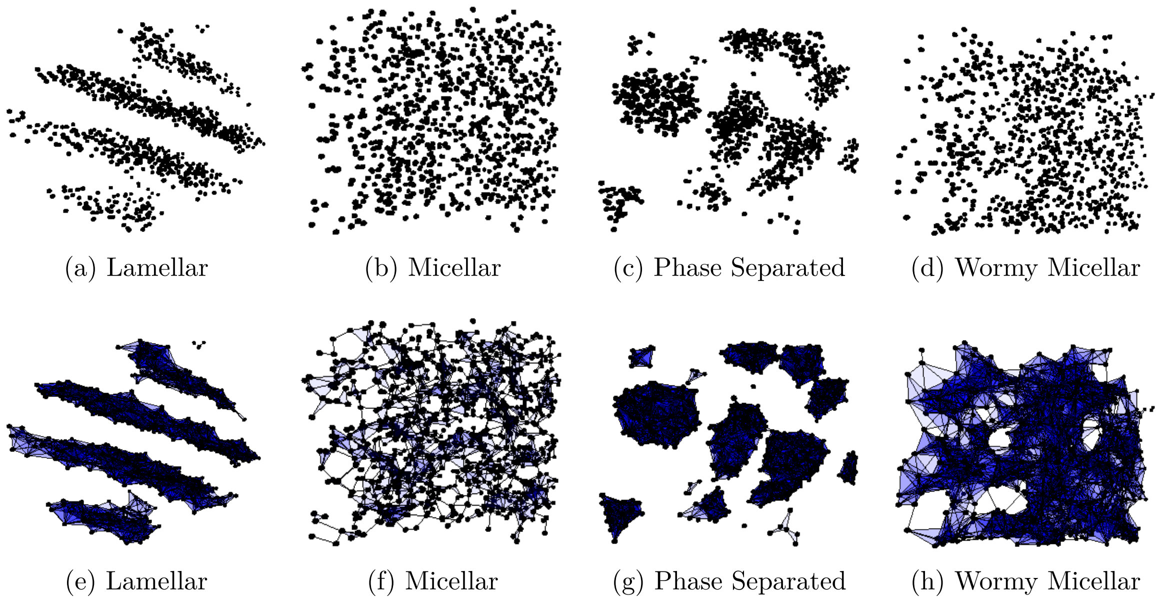
The distances between the 2-dimensional landscapes show that 38 differs from the other three. The reason for this is that the wormy micellar systems contain more and longer lived 2-dimensional homology classes than the other systems.

From this investigation, we can see that there are differences in the persistence diagrams of the systems. However, it is difficult to tell if these correspond to actual geometric aspects of the shape underlying the points or if they are due to the differences in size of the point clouds, the noise in the systems, and the choice of maximum filtration value.

We next investigate the water point clouds for these four systems, as illustrated in Figure 7.7(a)-(d). Each of these systems contains 1000 points that were chosen as a random subsample of the N_{water} points. Indeed we clearly see that there is water trapped in between the layers formed by the head groups of the surfactants in the lamellar phase. Hence, for the lamellar system, the water point cloud forms a collection of box-spanning layers, just like the head group point cloud. For the micellar phases, recall that the hydrophobic tails are inside the spheres (see Figure 7.2), hence the water is outside of the spheres, making it hard to see that there are in fact void spaces inside, corresponding to the micelles. For the reverse micellar system we see the opposite behaviour. In this case the hydrophobic tails point out from the spherical micelles, and water is trapped inside the reverse micelles. Hence we clearly see groups of water. The wormy micellar phase is similar to the micellar phase, water is surrounding the elongated micelles, leaving voids where the wormy micelles are located.

We compute the 0-,1- and 2-dimensional persistent homology groups up to a maximum filtration value of 1.5. This value was chosen manually as a trade off between computational constraints and structural features present in the Vietoris-Rips complex. Computationally, the main difficulty is with the phase separated systems, since they have very dense regions, causing out of memory exceptions for high maximum filtration values. Figure 7.7(e)-(h) illustrates the 2-skeleton of the four point clouds at $\epsilon = 1.5$.

We expect to see a similar persistence diagram for the lamellar water point cloud as we saw for its head group point cloud, since the underlying geometric object is basi-

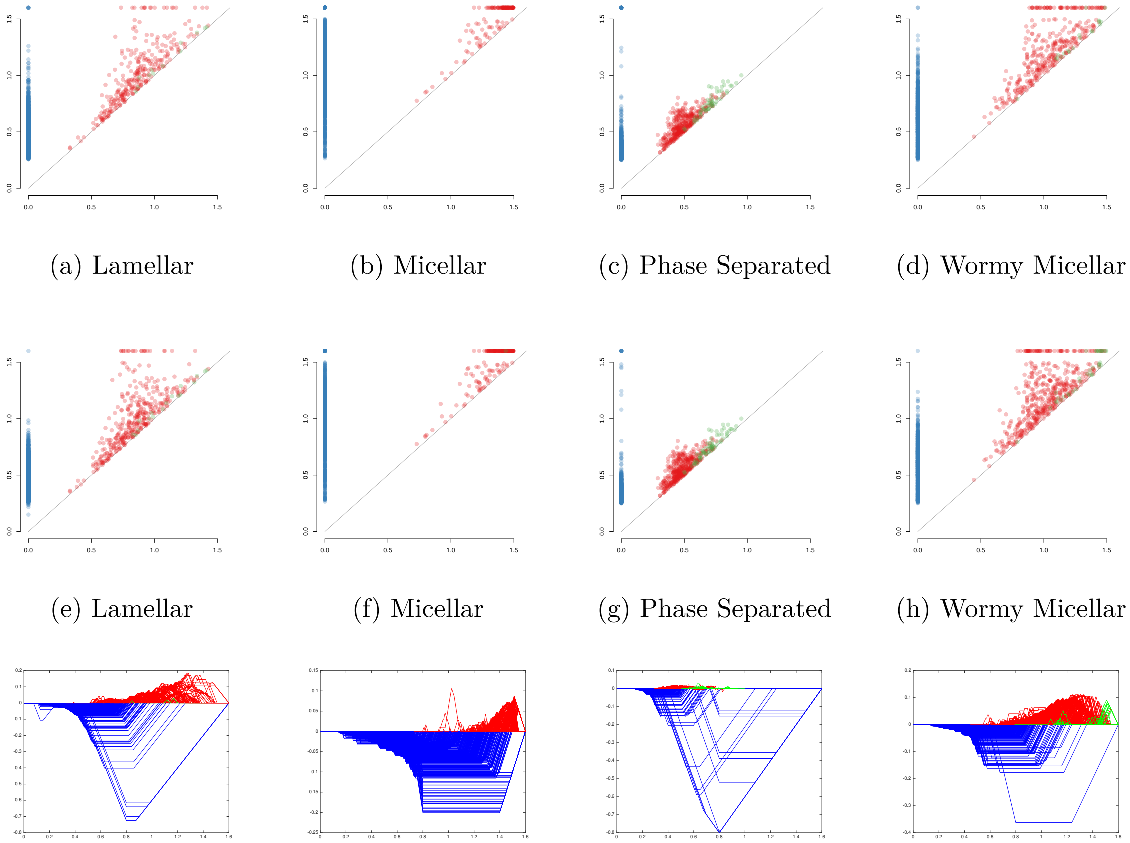

(a) Lamellar (b) Micellar (c) Phase Separated (d) Wormy Micellar

(e) Lamellar (f) Micellar (g) Phase Separated (h) Wormy Micellar

Figure 7.7.: (a)-(d) Point clouds extracted to represent the water molecules in systems 7, 24, 32 and 37. These are the same systems as illustrated in Figure 7.1, for each system, 1000 oxygen atoms were selected at random from the 9,936, 105,806, 2,708 and 16,560 water molecules respectively. (e)-(h) The 2-skeletons of the Vietoris-Rips complexes for each pointcloud at filtration value 1.5.

cally equivalent. However, there are some differences between the two point clouds and their persistence diagram. Surprisingly, we do not find any persisting 2-dimensional periodic homology classes for the water point cloud representation of the lamellar system, which we do expect to see since the box-spanning layers correspond to tori when periodic boundary conditions are taken into account. The fact that we do not find any 2-dimensional homology may be due to the layers not having been sampled densely enough, since this can lead to holes in the tori, which implies they no longer enclose a void.

For the micellar system, no interesting features appear in the persistence diagrams. This is due to f_{max} being quite low and the points being spaced quite evenly throughout the box. For higher filtration values we did find 2-dimensional homology classes corresponding to the voids left by the micelles. We leave the maximum filtration value at 1.5 here to be able to compare the four systems better.



(a) Lamellar (b) Micellar (c) Phase Separated (d) Wormy Micellar

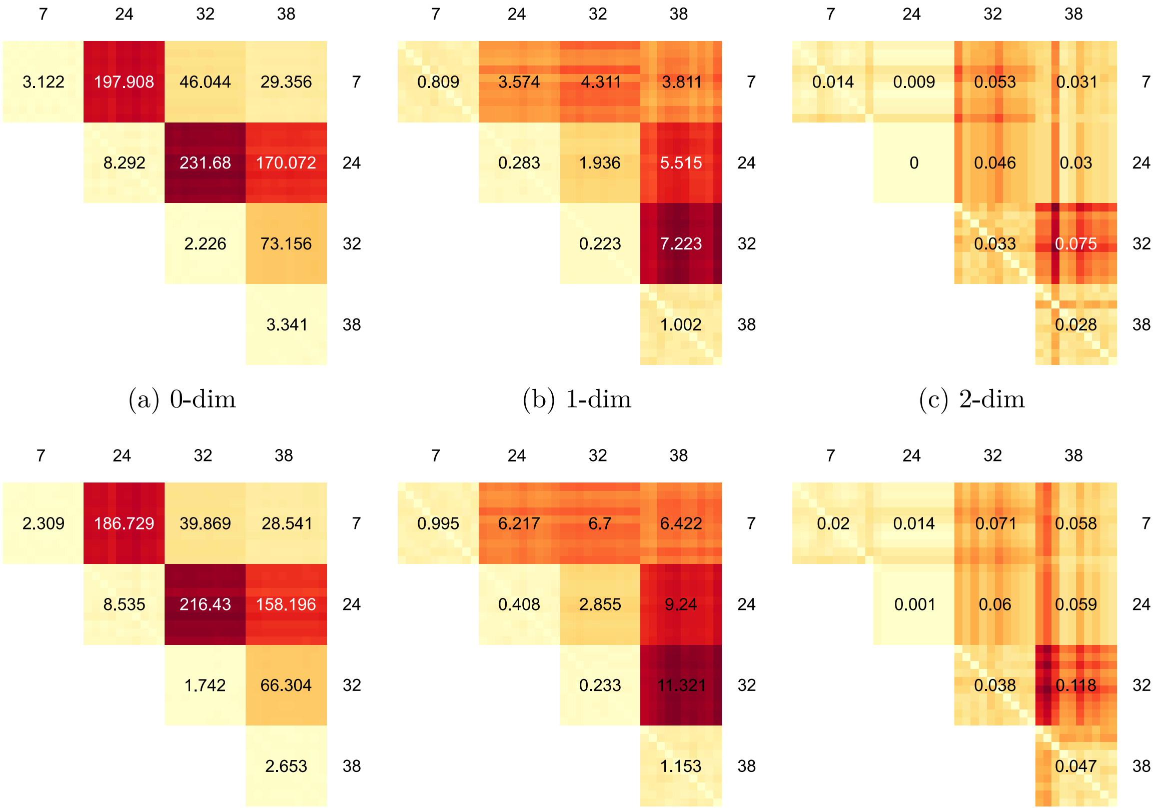
(e) Lamellar (f) Micellar (g) Phase Separated (h) Wormy Micellar

(i) Lamellar (j) Micellar (k) Phase Separated (l) Wormy Micellar

Figure 7.8.: (a)-(d) Persistence diagrams of the water point clouds for systems 7, 24, 32 and 37. Blue points correspond to 0-dimensional homology classes, red point to 1-dimensional homology classes and green points to 2-dimensional homology classes. (e)-(h) Persistence diagrams of the same four systems, with periodic boundary conditions. (i)-(l) Persistent landscapes of the *difference* between the persistence landscape with periodic and without periodic boundary conditions for the same four systems.

The phase separated system has a very distinct persistence diagram. It clearly shows multiple connected components persisting throughout the whole filtration, whereas higher dimensional homology classes basically only appear as noise. This corresponds exactly to the system, since each connected component is a solid sphere, which only has non-trivial 0-dimensional homology, i.e. $\beta_0 = 1$.

Finally the wormy micellar systems shows many 2-dimensional homology classes born towards the end of the filtration. Otherwise it looks quite similar to the persistence diagrams of the lamellar phase.



(a) 0-dim (b) 1-dim (c) 2-dim

(d) 0-dim periodic (e) 1-dim periodic (f) 2-dim periodic

Figure 7.9.: (a)-(c) Heat maps of L^1 landscape distances in dimensions 0,1 and 2, between persistence landscapes of ten subsamples of each of the water point clouds for systems 7, 24, 32 and 38. The numbers equal the average pairwise distance for that block. For diagonal elements, this is the average landscape distance between sub samples of the same system. (d)-(f) The same heat maps for homology with periodic boundary conditions.

Figures 7.9(a)-(f) show heat maps corresponding to the pair-wise L^1 landscape distances in dimensions 0, 1 and 2, between the ten subsamples of the water point clouds for systems 7, 24, 32 and 38. The colours represent the dissimilarity; the darker red the more dissimilar. They are scaled for each heat map. The numbers equal the average distance within a given block. For the off-diagonal blocks this is the average of 100 pairwise distances between 10 1000 point subsamples of one system and 10 1000 point subsamples of another system. For the diagonal blocks it is the average of 45 pair-wise distances between 10 1000 point subsamples of one system.

In general, the heat maps also are very similar for periodic and non-periodic homology. For 0-dimensional homology, the average distance between subsamples from the same system (i.e. the diagonal elements in Figures 7.9(a) and (d)) is much smaller than the average distance between subsamples of different systems (i.e. the off-diagonal elements in Figures 7.9(a) and (d)). For 1-dimensional homology, the distances are closer together, but still significantly larger between subsamples from different systems than between subsamples from the same system. For 2-dimensional homology, the distances between subsamples of the same systems and subsamples of different systems are very similar. Hence, we do not discuss the 2-dimensional heat maps any further.

Figures 7.9(a) and (d) both show that system 24 is very different from the other three systems in terms of its 0-dimensional persistent homology. Visually, system 24 is the most random of the four point clouds (Figure 7.7(e)-(h)). Compared to the other systems, there are more long living components (see Figures 7.8(b) and (f)). This is in agreement with more randomly spaced and less clustered points.

Figures 7.9(b) and (c) show that 1-dimensional homology differentiates system 38, the wormy micellar system, well from the others. The persistence diagrams show that this may be caused by the large number of persistent 1-dimensional homology classes in system 38 (see Figure 7.7(d) and (h)). These differences are more pronounced for homology with periodic boundary conditions. System 7 is also quite different from the others, especially when periodic boundary conditions are taken into account.

These initial investigations show that persistent homology provides valuable information on the structure of the four different phases. However, we made several parameter choices manually. Further research is needed to obtain methods for automatic parameter selection and to test robustness.

7.4. A simple classifier

We now focus on our main goal, the classification of lipid formulations using persistent homology. We remove some of the systems from this task, because they have too limited a number of points in the point cloud (system 26) or because visually the system does not resemble the representative of its class. We remove systems 1, 2, 3 and 4 from the lamellar phase and systems 27, 28, 29 and 34 from the phase separated phase. We are left with 37 systems.

In this section all persistent homology computations are presented without periodic boundary conditions, since this gives slightly better classification results.

The first simple classifier that we employ attempts to classify the 33 systems not equal to the representatives chosen in Section 7.3. We compare each of the remaining systems to the representatives and give it the same label as the representative that it is most similar to. We do this for nine distance measures and compare the results. The first four distance measures are based on the landscape distance between persistence diagrams of the surfactant head group point clouds: d_0^{surf}, d_1^{surf}, d_2^{surf} and d^{surf}, where d_i^{surf} is the i-dimensional L^1 landscape distance and $d^{surf} = \sum_{i=0}^{2} \bar{d}_i^{surf}$ with $\bar{d}_i^{surf}$ the normalised distances.

For the water point clouds we use slightly more complicated distance measures. For each system, we take ten random sub samples of 1000 points and compute the corresponding persistence landscapes in dimensions 0, 1 and 2. We then compute the average of those ten landscapes [20] and use it to represent the water point cloud of the system. We base our distance measures on the L^1 landscape distances between the average landscapes. That is $d_i^{water}(X, Y)$ is the landscape distance between the average i-dimensional landscape of ten samples of 1000 points of X and the average i-dimensional landscape of ten samples of 1000 points of Y. We define $d^{water} = \sum_{i=0}^{2} \bar{d}_i^{water}$ with $\bar{d}_i^{water}$ the normalised distances. Our final distance measure is d^{total} which is defined as the sum of d^{surf} and d^{water}.

Definition 7.4.1. The **accuracy** [113] of a multi-class classifier is defined as the number of correctly identified objects divided by the total number of identified objects.

Distance	Accuracy	Distance	Accuracy
d_0^{surf}	0.76	d_0^{water}	0.64
d_1^{surf}	0.61	d_1^{water}	0.58
d_2^{surf}	0.42	d_2^{water}	0.39
d^{surf}	0.70	d^{water}	0.70
d^{total}	0.52		

Table 7.2.: The accuracy of nine classifiers based on different distance measures.

	Ground truth			
Prediction	Lamellar	Micellar	Phase Sep	Wormy Mic
Lamellar	2	0	0	0
Micellar	0	8	0	1
Phase Sep	7	0	3	0
Wormy Mic	1	1	0	10

Table 7.3.: Confusion matrix for classifier based on d^{surf}.

The accuracy of the classifiers based on these nine distance measures is listed in Table 7.2.

Even though d_0^{surf} is the best performing classifier, we prefer to use the measures d^{surf} and d^{water} since they take into account the 0-, 1- and 2-dimensional homology features of the systems. Summing these distances leads to much worse performance.

The confusion matrices [114] of the classifiers corresponding to d^{surf} and d^{water} are shown in Tables 7.3 and 7.4. Both perform well on the wormy micellar systems and the phase separated systems very well. However, the classifier based on surfactants has very poor performance for the lamellar phase, whereas the water based classifier performs perfectly for these systems. By contrast6, the surfactant based classifier has near perfect performance for the micellar phase, whereas the water based classifier misclassified nearly all of these systems.

This leads us to belief that an approach which combines the structural properties of the surfactant point clouds and the water point clouds could lead to an excellent classi-

	Ground truth			
Prediction	Lamellar	Micellar	Phase Sep	Wormy Mic
Lamellar	10	0	0	1
Micellar	0	2	0	2
Phase Sep	0	0	3	0
Wormy Mic	0	7	0	8

Table 7.4.: Confusion matrix for classifier based on d^{water}.

fier.

7.5. Hierarchical clustering

In this final section we discuss an experiment that again uses landscape distances between persistence landscapes to classify the different systems. However, we no longer want our results to depend on the choice of representative states. Instead, we use a hierarchical clustering approach to analyse if we can obtain good clustering without specifying a representative for each class.

We first focus on the head group point clouds, representing the surfaces formed by the head groups of the surfactants. We compute the d^{surf} distance between each pair of persistence landscapes for the 37 systems discussed in Section 7.4. Figure 7.10 shows the heat map together with the dendrogram obtained by using average linkage clustering [57], that is at each step in the dendrogram the clusters which have closest average distance are merged, for this distance matrix. We used average linkage clustering because it combines clusters based on the distance between all elements in the clusters.

The ordering of the columns and rows is determined by the dendrogram. The clustering in the dendrogram is very good for the micellar phase (yellow): the only system that is not part of the cluster is system 16. Close inspection of this point cloud (see Figure D.2(d)) reveals that it indeed looks more like a wormy micellar system than a micellar

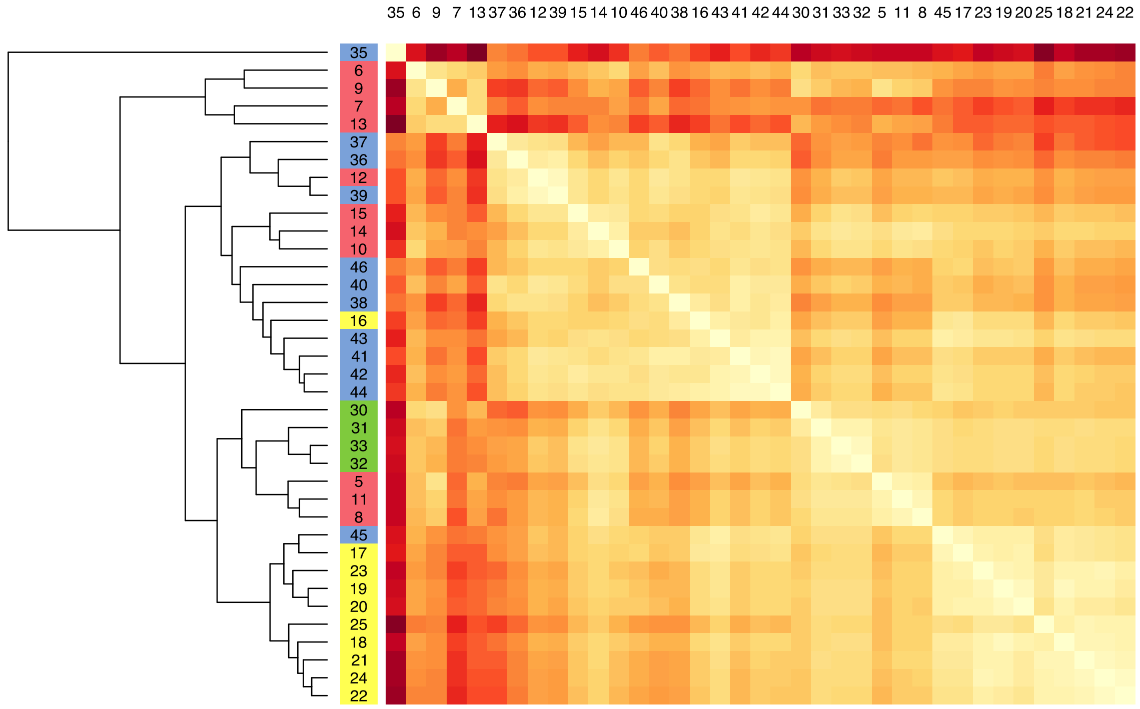


Figure 7.10.: Heat map for d^{surf} distances. Light yellow corresponds to small distances, dark red to large distances. The dendrogram is made using average linkage.

system. This is exactly what it is grouped with in the dendrogram.

System 45 is grouped with the micellar systems, and on close inspection is borderline between micellar and wormy micellar (see Figure D.4(i)).

In this dendrogram there is no clear distinction between the phase separated and lamellar systems. This is not a complete surprise since the simple classifier in Section 7.2 similarly misclassified lamellar systems as phase separated. To separate these phases we require further information.

Next we investigate the clustering based on the d^{water} distances between the average persistence landscapes of sets of 10 sub samples of 1000 points in the water point clouds. Figure 7.11 shows the heat map corresponding to this distance matrix together with the average clustering dendrogram based on these distances.

Again we notices that system 16 is clustered with the wormy micellar systems (blue),

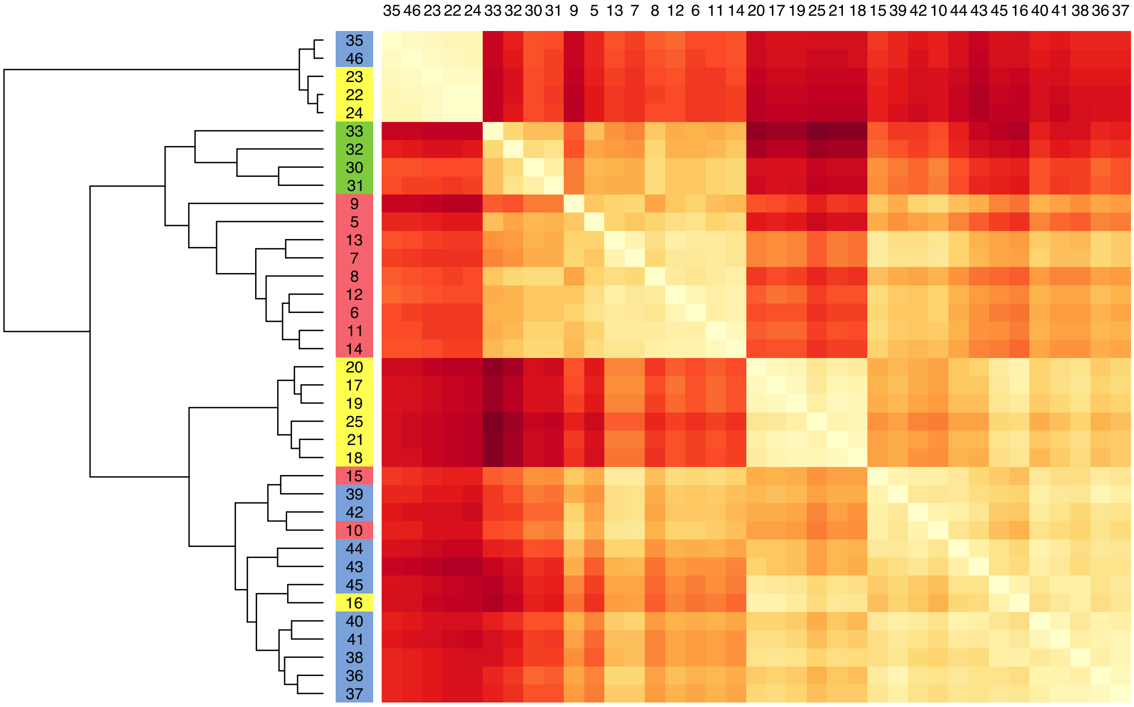


Figure 7.11.: Heat map for d^{water} distances. Light yellow corresponds to small distances, dark red to large distances. The dendrogram is made using average linkage.

instead of the micellar systems. However, this time system 45 is clustered with the wormy micellar systems. The micellar systems are split into two distinct groups. Systems 22,23 and 24 correspond to spaced micelles, whereas systems 17,18,19,21 are closely packed micelles. Since we are investigating the geometry and topology of the complement of these micelles, it makes sense that the systems where the micelles occupy a smaller space (the spaced micelles) form a cluster separate from the micelles that occupy a large part of the space (closely packed micelles). Even though system 25 is classified as spaced micellar, it is clustered with the densely packed micelles. Looking at a visualisation of this system (see Figure D.3(a))it is not so clear if it should be classified as densely packed or spaced.

The phase separated systems are again clustered together nicely. Most of the lamellar systems are grouped together. However, system 10 and 15 are grouped with the wormy micellar systems. This is possibly caused by the chaotic defects in these lamellar systems.

Finally the wormy micellar systems are split into two clusters, one that just contains system 35 and 46 and one that contains all other systems. System 35 and 46 contain wormy micelles that are spaced out quite nicely. This results in the grouping with the three spaced micellar systems.

Just as we found with the basic classifier in Section 7.4, the water based classifier has trouble distinguishing the micellar and wormy micellar systems. However, we have seen that it can distinguish between densely packed and spaced configurations.

7.6. Conclusion

In this chapter we discussed a classification problem originating in the field of pharmaceutical science. The problem is related to the development of efficient drug delivery capsules for drugs that are poorly soluble in water. As part of investigating the potential of a lipid formulation to be used in a drug delivery capsule, pharmaceutical scientists run molecular dynamics simulations. The outcome of these simulations predicts the structure that will be formed by the lipid formulation. Currently the resulting configurations are inspected manually. This is both time consuming and prone to error. We investigated the potential of persistent homology to automatically classify the phase behaviour of these configurations.

We analysed 46 different outcomes of molecular dynamics simulations. We represented each system by two point clouds, one based on the locations of the surfactant head groups and one based on the location of water in the systems. We were able to show that persistent homology distinguishes different phases, depending on whether we analyse the head group point cloud or the water point cloud. Using very simple classifiers we got reasonable classification results. This research indicates that persistent homology is a promising approach for the classification of the phase behaviour in lipid formulations.

We have not yet developed this technique to its full potential and expect that it is possible to built better classifiers based on the information extracted from the point clouds using persistent homology. However, we think further exploratory data analysis is needed to avoid over-fitting and to make sure that this method is robust.

The dataset that we worked with presents several challenges. First, the data set is relatively small. It would be highly desirable to have access to additional lipid formulations. This is difficult, since molecular dynamics simulations take a very long time to run (in the order of weeks on a super computer). Second, most of the systems have a significant amount of noise so applying a noise reduction algorithm prior to the topological data analysis might improve our results. Third, there is a large variety in the number of points in the different systems. It is important to obtain robustness results for varying the number of points in the data set. Finally, some of the systems that have the same phase label, have significantly different geometry or topology. For example, micellar phases can consist of few large micelles or many small micelles. We like a classifier to assign the same label both systems. Exploratory data analysis is crucial to understand which properties need to be used in a good classifier.

Another direction for future research is the use of alpha shapes [36] instead of the Vietoris-Rips complex. Alpha shapes are an alternative way to associate a simplicial complex to a 3-dimensional point cloud. Very recently we discovered that this is a common way to describe a system of molecules [126].

Finally, our approach should benefit from the use of more advanced classifiers, using machine learning.

We believe our approach to the classification of lipid formulations to be a promising one and a novel application of persistent homology.

8. Conclusion

Network science is the field of research in which relational data is represented and analysed as networks. In this thesis we present results on three distinct but related topics in network science: random network models that fix the degree sequence of a network; local network properties, in particular network motifs and neighbourhood distinctiveness; and the application of topological data analysis to networks.

Random network models Several important techniques in network analysis rely on random network models: algorithms that generate networks using a random process. For instance, these models are used in identifying motifs in networks and in detecting communities or clusters. A crucial property of random network models is that they are unbiased. It has turned out to be challenging to develop fast and unbiased random network models. In this thesis we focused on random network models that fix the degree sequence of a network.

We demonstrated that the ambiguity in the definition of the switching model has lead to different interpretations and implementations of the switching model. This, and the fact that the switching model has been used to randomise several different classes of networks without addressing their differences, has in some cases lead to biased sampling.

We introduced a precise definition of a family of switching models with respect to nine classes of networks. We showed that using this definition, there is no ambiguity as to how to implement the models. We derived conditions under which the different versions of the switching model converge to the uniform distribution. In some cases, it was necessary to introduce acceptance probabilities to ensure unbiased sampling.

We compared the run-time of the switching model to several existing random network models. We demonstrated that even though the configuration model runs faster, it always samples from the classes of multigraphs (i.e. $\mathcal{G}_6$ and $\mathcal{G}_8$). We demonstrated that it is unlikely for other Markov chains that use the same set of moves as the switching model (i.e. allowed switches) to improve on the run-time of the switching model.

The recently introduced Curveball algorithm does improve on the run-time of the switching model. We proved that this algorithm samples without bias; a position that was previously implied, but unproven. We introduced several variations of the Curveball algorithm. Of particular importance are the Simple Curveball algorithm and the Simple Undirected algorithm. We showed that these algorithms sample simple directed and simple undirected networks without bias respectively.

As a result of our findings in Chapters 2 - 4 we recommend using the configuration model to randomise multi-graphs with self-loops (classes $\mathcal{G}_6, \mathcal{G}_8$), variations of the Curveball algorithm to randomise networks without multiple edges or without self-loops (classes $\mathcal{G}_1 - \mathcal{G}_5$ and $\mathcal{G}_7$) and the ordered switching model to randomise directed acyclic networks.

We pointed out two advantages of the configuration model as compared to Markov chain approaches for network randomisation. First, the random networks produced by the configuration model are in no way related to the original network and second it runs much faster.

For Markov chain methods on the contrary, it is unclear how many steps are necessary to ensure the generated network is not correlated to the original network. Usually a large number of steps is made to ensure the chain has converged to its stationary distribution. This is the main cause for Markov chain methods to run more slowly.

The Markov chain approach has two advantages as compared with the configuration model approach. It offers flexibility in terms of which class of network is being sampled from as well as which distribution is being sampled from (using acceptance probabilities).

The configuration model on the other hand, always samples from the class of multigraphs and does not offer flexibility in terms of the sampling distribution.

The recently introduced Expand and Contract method combines the configuration model and Markov chain approaches. It generates networks that are in no way related to the original network, just like the configuration model, and it offers the same flexibility to sample from different classes of networks as Markov chain approaches. In this sense it really is the best of both worlds.

However, it is no longer as fast as the configuration model approach. We implemented the Expand and Contract method and showed experimentally that in several cases its run-time is similar to that of the switching model and in some cases it is much longer. Further research is needed to establish under which conditions the Expand and Contract method is a practical method for network randomisation.

The Expand and Contract method was also claimed to sample without bias. However, we argue for thoroughly checking this claim before adopting the Expand and Contract method.

Currently the Expand and Contract method combines the configuration model with a Markov chain similar to that of the switching model. For directed networks we recommend to speed up the Expand and Contract method by replacing this Markov chain by the Curveball algorithm with respect to $\mathcal{G}_8$. For undirected networks, we recommend to develop a Curveball algorithm with respect to $\mathcal{G}_6$.

The Curveball algorithm has been shown to improve on the speed of the switching model experimentally. However, there are proofs of rapid mixing of the switching model for specific classes of networks and these proofs break down when attempting to transfer them to the Curveball algorithm. Proving that the Curveball algorithm mixes rapidly is an interesting open question.

Local network properties Local network properties are an important method for the analysis of networks and many existing measures rely on them. For instance, motifs are a popular method of describing the building blocks of a network.

Existing motif finding techniques were developed for simple directed and simple undirected networks. In this thesis we have extended the theory of motif detection to the class of directed acyclic networks. To do so we developed the ordered switching model, a model that produces random directed acyclic networks with a given degree sequence. We derived formulas relating the number of occurrences of 3-node patterns in a directed acyclic network to its degree sequence. In our experiments, we compared motifs detected by three different null-models. We showed that the choice of null-model had no effect on the patterns that were determined to be motifs. The real networks differ so significantly from all three random networks, that their mutual differences do not seem to matter in this case. However, the results may differ in other networks, and therefore we recommend using the ordered switching model; it is the only model that produces directed acyclic networks with fixed degree sequences and fixed topological ordering uniformly at random.

The neighbourhood of a node can be used as a local network measure. We analysed a novel application of neighbourhood analysis, namely using the neighbourhood of a node for identification of the node itself across networks. Our motivation comes from aiming to identify entities that wish to remain hidden, in particular in dark social networks.

We demonstrated that the influence neighbourhood is a distinctive characteristic of a node in both the Enron email database and the CiteSeer citation network. We showed that the influence neighbourhood has potential for good matching performance across databases. Building on this work, my collaborators showed that indeed the influence neighbourhood performs well at matching entries across databases.

Topological data analysis

Persistent homology is a relatively new data analysis technique and is becoming increasingly popular. We surveyed the recent literature on applications of persistent homology in network analysis. We conclude that it has three nice features that distinguish it from other network analysis techniques. Firstly, homology reveals information on the mesoscopic structure of a network that local statistics do not capture. Secondly, the simplicial complexes used in persistent homology enrich the representation of relational data as networks, and finally persistent homology analysis a parametrised family of networks as a single object.

We demonstrated that this final property makes persistent homology particularly applicable to weighted networks and evolving networks. Our experiments on weighted collaboration networks showed that the persistent homology of the weighted clique rank filtration detects both structural differences and difference in weight distribution between networks. We have left the analysis of the temporal clique rank complex and the analysis of bipartite networks using persistent homology as future research.

We used persistent homology for a classification problem in pharmaceutical science. This is a novel application of persistent homology. Our analysis demonstrates that persistent homology is a promising approach for the classification of the phase behaviour of lipid formulations. We have not yet developed this technique to its full potential and believe there are many directions for improvement of the simple classifier that we presented. This is left as future work.

Appendices

A. Data Sets

A.1. Astrophysics: collaboration network

Source: http://www-personal.umich.edu/∼mejn/netdata
Accessed: Downloaded September 2012
Type: undirected weighted simple network, obtained as a projection from a bipartite network
Number of vertices: 14845
Number of edges: 119652
Description: A network of coauthorships between scientists posting preprints on the Astrophysics E-Print Archive between Jan 1, 1995 and December 31, 1999 [91].
Processing: We use the largest connected component of this network. The vertex and edge count corresponds to this component.

A.2. C. Elegans: neural network

Source: http://www-personal.umich.edu/∼mejn/netdata
Accessed: Downloaded June 2015
Type: directed weighted network
Number of vertices: 297
Number of edges: 2359
Description: Compiled by Duncan Watts and Steven Strogatz [120] from original ex-

perimental data by White et al. [122]. The file celegansneural.gml describes a weighted, directed network representing the neural network of C. Elegans. The data were taken from the web site of Prof. Duncan Watts at Columbia University[1]. The nodes in the original data were not consecutively numbered, so they have been renumbered to be consecutive. The original node numbers from Watts' data file are retained as the labels of the nodes. Edge weights are the weights given by Watts.
Processing: NA

A.3. CiteSeer Archive: citation network

Source: http://citeseer.ist.psu.edu/oai.html
Accessed: Downloaded May 2010
Type: directed network
Number of vertices: 383,535
Number of edges: 1,740,303
Description: The CiteSeer Archive is a database describing papers primarily in computer and information science and published before 2005. The database consists of 2.1GB of data in a format that is close to valid XML format. The archive contains 716,772 records of publications, each including a CiteSeer identifier, title, author information, publication date and references.
Processing: This dataset was downloaded by a former member of the Information Security and Network Science Group at RMIT University. The processing described here is my own work. Due to inconsistencies in the XML format, standard XML parsers fail to parse the data. We instead parsed the file line by line using custom code, written in the R language. We extracted a paper citation network, where the nodes correspond to papers and the edges to citations. An edge is directed from the citing paper to the cited paper. Nearly half of the papers in the database did not have any citations or contain references. These papers were removed from the network. Some papers contained duplicate references or self-citations. Such spurious citations were also removed, so that the resulting network does not have multiple edges or loops.

[1] http://cdg.columbia.edu/cdg/datasets

A.4. Condensed matter: collaboration network

Source: http://www-personal.umich.edu/∼mejn/netdata
Accessed: Downloaded September 2012
Type: undirected simple network, obtained as a projection from a bipartite network
Number of vertices: 36458
Number of edges: 171735
Description: A network of coauthorships between scientists posting preprints on the Condensed Matter E-Print Archive between Jan 1, 1995 and March 31, 2005 [91].
Processing: We use the largest connected component of this network. The vertex and edge count corresponds to this component.

A.5. Enron: email communication network

Source: http://dl.dropbox.com/u/1800572/enron/enron-cleaned.edges
Accessed: Downloaded November 2014
Type: temporal directed network
Number of vertices: 22477
Number of edges: 53285
Description: This data set is a record of emails sent within the Enron company between 01/01/1997 and 31/12/2005. We used a cleaned version of the Enron email corpus. See http://sociograph.blogspot.com.au/2011/04/communication-networks-part-1-enron-e.html for an elaborate description of the data cleaning.
Processing: We only use the structural features of the network, ignoring the timestamps.

A.6. High energy physics - theory: collaboration network

Source: http://www-personal.umich.edu/∼mejn/netdata
Accessed: Downloaded September 2012
Type: undirected simple weighted network, obtained as a projection from a bipartite network
Number of vertices: 5835
Number of edges: 13815
Description: A network of coauthorships between scientists posting preprints on the High-Energy Theory E-Print Archive between Jan 1, 1995 and December 31, 1999 [91].
Processing: We use the largest connected component of this network. The vertex and edge count corresponds to this component.

A.7. High energy physics - theory: citation network

Source: http://www.cs.cornell.edu/projects/kddcup/datasets.html
Accessed: downloaded May 2013
Type: directed temporal network
Number of vertices: 27770
Number of edges: 351389
Description: This data set is based on a collection of papers from the arXiv in the theoretical high energy physics category. For most papers in the data set, the corresponding upload/publication date is available. These dates range from 1976 to 2003.
Processing: We removed all papers that have neither citations nor references (within the database). We also removed papers for which no publication date is available. For papers that occur multiple times with different dates we used the earliest date and disregarded the other dates. There are few citations that do not respect the time ordering on the papers, i.e. papers that cite papers that were published on the same date as, or after their own publication date. Such citations were removed. The resulting network is a directed acyclic network, each vertex has an associated date and all edges $u \to v$ have the property that u is more recent than v. The number of vertices and edges in the

network listed corresponds to the network post processing.

A.8. High energy physics - phenomenology: citation network

Source: http://snap.stanford.edu/data/cit-HepPh.html
Accessed: downloaded May 2013
Type: directed temporal network
Number of vertices: 30501
Number of edges: 344617
Description: This data set is based on a collection of papers from the arXiv in the phenomenology high energy physics category. For most papers in the data set, the corresponding upload/publication date is available. These dates range from 1992 to 2003.
Processing: We removed all papers that have neither citations nor references (within the database). We also removed papers for which no publication date is available. For papers that occur multiple times with different dates we used the earliest date and disregarded the other dates. There are few citations that do not respect the time ordering on the papers, i.e. papers that cite papers that were published on the same date as, or after their own publication date. Such citations were removed. The resulting network is a directed acyclic network, each vertex has an associated date and all edges $u \to v$ have the property that u is more recent than v. The number of vertices and edges in the network listed corresponds to the network post processing.

A.9. Network Science: collaboration network

Source: http://www-personal.umich.edu/∼mejn/netdata
Accessed: Downloaded September 2012
Type: undirected simple network, obtained as a projection from a bipartite network

Number of vertices: 379
Number of edges: 914
Description: A coauthorship network of scientists working on network theory and experiment. Compiled by M. Newman in May 2006. [92].
Processing: We use the largest connected component of this network. The vertex and edge count corresponds to this component.

A.10. US airport data: transport network

Source: R package *igraphdata*
Accessed: Nov 2015
Type: directed temporal multigraph
Number of vertices: 755
Number of edges: 8228
Description: The network of passenger flights between airports in the United States. The data set was compiled based on flights in 2010 December. This network is directed and edge directions correspond to flight directions. Each edge is specific to a single carrier aircraft type. Multiple carriers between the same two airports are denoted by multiple edges. Most of this information was downloaded from The Research and Innovative Technology Administration (RITA)[2].
Processing: We flatten the network to a static directed network and simplify it to remove multiple edges.

A.11. Zachary's karate club: social network

Source: http://www-personal.umich.edu/∼mejn/netdata
Accessed: June 2015
Type: undirected network

[2] See http://www.rita.dot.gov/about_rita/ for details

Number of vertices: 34
Number of edges: 78
Description: A friendship network between 34 members of a karate club at a US university in the 1970s [132].
Processing: NA

B. Software

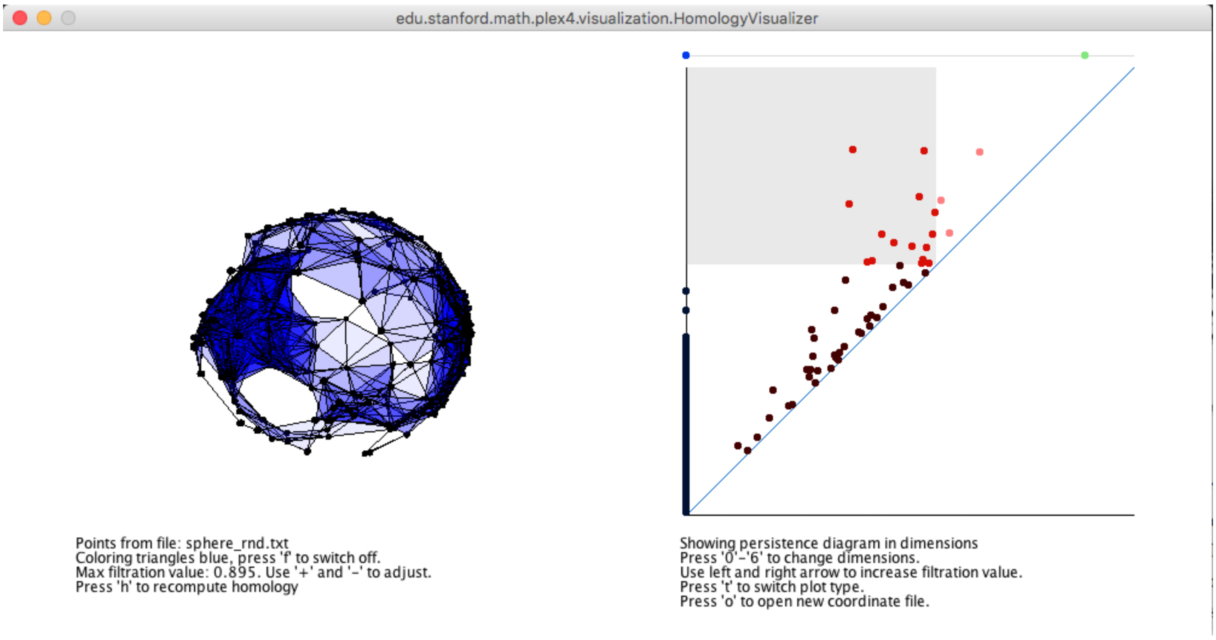
B.1. Developed

B.1.1. Curveball style algorithms

Description: This GitHub repository contains code to reproduce the results presented in [26]. In particular it contains an implementation of the original Curveball algorithm and the Good Shuffle algorithm. Furthermore it contains implementations of the Simple Curveball algorithm, the Simple Undirected Curveball algorithm and the other versions of the Curveball algorithm that are discussed in Chapter 4. **Source code:** http://github.com/queenBNE/Curveball

B.1.2. HomViz: Homology Visualizer

Description: HomViz is a graphical user interface for persistent homology and powered by JavaPlex. It provides an interactive visualization of the Vietoris-Rips complex of a point cloud and its corresponding persistent diagram at different filtration values. Furthermore it has the capacity to highlight generators of homology classes.
URL: http://jacobiencarstens.com



Points from file: sphere_rnd.txt
Coloring triangles blue, press 'f' to switch off.
Max filtration value: 0.895. Use '+' and '–' to adjust.
Press 'h' to recompute homology

Showing persistence diagram in dimensions
Press '0'–'6' to change dimensions.
Use left and right arrow to increase filtration value.
Press 't' to switch plot type.
Press 'o' to open new coordinate file.

Figure B.1.: Screenshot of HomViz software. On the left: a visualization of the 2-skeleton of the Vietoris-Rips complex, $V(X, \epsilon)$, of a random sample of points X from the surface of a sphere. The visualization can be rotated using the mouse. On the right: the corresponding persistence diagram, blue points correspond to 0-dimensional homology classes, red points to 1-dimensional homology classes and green points to 2-dimensional homology classes. The grey box indicates the current value of ϵ in the visualization on the left, and the brightness of each point indicates if the class is already present in the filtration at this stage: light means 'not yet born', normal means 'born', dark means 'died'. The arrow keys can be used to increase and decrease ϵ.

B.1.3. JavaPlex pbc extension

Description: This repository contains code that extends JavaPlex. It allows the computation of the Vietoris-Rips complex of a point cloud that is embedded in a cube with periodic boundary conditions.

Source code: https://github.com/queenBNE/Periodic-Boundary-Conditions-JavaPlex (to appear)

B.1.4. Persistence Landscapes Wrapper

Description: Persistence landscapes offer a convenient topological summary of persistent homology. The Persistence Landscape Wrapper provides a Matlab interface for the C++ library 'persistent landscape toolkit' that was developed by Paweł Dłotko.
Source code: https://github.com/queenBNE/Persistent-Landscape-Wrapper.

B.1.5. Random directed acyclic networks

Description: This software repository contains implementations of the ordered switching model and the Karrer-Newman model for the randomisation of directed acyclic networks. There are two implementations of the ordered switching model, one written in R (slow) and one incorporated into MFinder (fast).
URL: http://github.com/queenBNE/DirectedAcyclicNetworks

B.1.6. Switching Models

Description: This software repository contains implementations of the switching model with respect to classes $\mathcal{G}_1 - \mathcal{G}_8$ as discussed in Chapter 2. This includes the adjusted versions as discussed.
Source code: https://github.com/queenBNE/SwitchingModelFamily

B.2. Used

B.2.1. Curveball algorithm

Description: Several implementations of the original Curveball algorithm in the Python and R programming languages. We used the R implementation, i.e. Supplementary Software 5.
URL: http://www.nature.com/ncomms/2014/140611/ncomms5114/full/ncomms5114.html#/sup
information

B.2.2. Gephi

Description: Gephi is an interactive visualization and exploration platform for all kinds of networks and complex systems, dynamic and hierarchical graphs.
URL: http://gephi.github.io/

B.2.3. igraph

Description: igraph is a collection of network analysis tools with the emphasis on efficiency, portability and ease of use. igraph is open source and free. igraph can be programmed in R, Python and C/C++.
URL: http://igraph.org

B.2.4. JavaPlex

Description: The JavaPlex library implements persistent homology and related techniques from computational and applied topology, in a library designed for ease of use, ease of access from Matlab and java-based systems, and ease of extensions for further research projects and approaches.

JavaPlex is mainly developed by the Computational Topology workgroup at Stanford University, and is based on previous similar packages from the same group.
URL: https://github.com/appliedtopology/javaplex

B.2.5. MFinder

Description: MFinder is a software tool for network motifs detection, developed by Nadav Kashtan, Shalev Itzkovitz, Ron Milo and Uri Alon.
URL: http://www.weizmann.ac.il/mcb/UriAlon

B.2.6. Persistence Landscape Toolbox

Description: The Persistence Landscape toolbox is a collection of algorithms to work with persistence landscapes. The persistence landscape is an alternative topological summary to the persistence diagram. It allows performing common statistics, such as computing a mean and performing t-tests. Developed by Paweł Dłotko.
URL: https://www.math.upenn.edu/ dlotko/persistenceLandscape.html

B.2.7. VMD

Description: VMD (Visual Molecular Dynamics) is a molecular visualization program for displaying, animating, and analyzing large biomolecular systems using 3-D graphics and built-in scripting. VMD supports computers running MacOS X, Unix, or Windows, is distributed free of charge, and includes source code.
URL: http://www.ks.uiuc.edu/Research/vmd/

C. State graphs

In this appendix we illustrate (parts of) the state graphs of the Curveball algorithm for regular directed networks on three to six vertices. We will denote these state graphs by Ψ. Notice that the state graph of the 1-regular directed network and the 2-regular directed network on three vertices are the same (see C.1.1). The reason for this is that the adjacency matrix of a 1-regular network is exactly the binary complement[1] of that of the adjacency matrix of a 2-regular network. Or, in other words, the corresponding networks are complements.

We may think of the adjacency matrix of a directed network as the incidence matrix of a bipartite network as illustrated in Figure C.1.

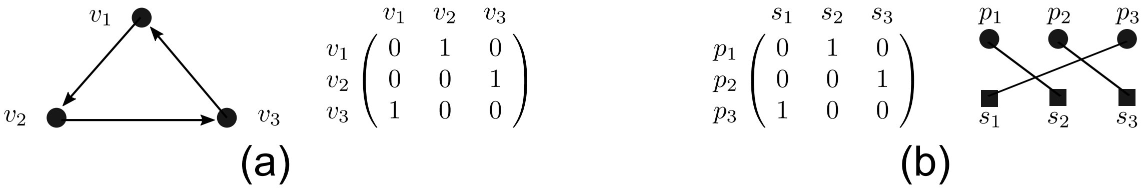

v_1
v_2
v_3
$\begin{array}{c|ccc} & v_1 & v_2 & v_3 \\ \hline v_1 & 0 & 1 & 0 \\ v_2 & 0 & 0 & 1 \\ v_3 & 1 & 0 & 0 \end{array}$
(a)
$\begin{array}{c|ccc} & s_1 & s_2 & s_3 \\ \hline p_1 & 0 & 1 & 0 \\ p_2 & 0 & 0 & 1 \\ p_3 & 1 & 0 & 0 \end{array}$
p_1
p_2
p_3
s_1
s_2
s_3
(b)

Figure C.1.: (a) A 1-regular network on three vertices and its adjacency matrix. (b) The same matrix, but interpreted as the incidence matrix of a 1-biregular bipartite network on six vertices.

The number of isomorphism classes of k-biregular bipartite networks is smaller than the number of k-regular directed networks for $k > 0$ and $n > 1$, since neither diagonal elements nor symmetric entries play a special role in the incidence matrix, but they do in the adjacency matrix (self-loops and bidirectional edges). We will see that the state graphs simplify when we think about them from the bipartite point of view.

[1] Replacing all zeroes by ones and vice versa.

There are eight different state graphs for regular directed networks on three to six vertices as listed in Table C.1. For the first example we show the isomorphism classes of both 1-regular and 2-regular networks on three vertices. However, in the other examples where the state graph represents either randomising k-regular or $n-k$-regular networks, we just show the isomorphism classes for either the k-regular or the $n-k$-regular networks.

G			Ψ		G			Ψ		G			Ψ		G			Ψ	
n	k	$n-k$	n_Ψ	m_Ψ	n	k	$n-k$	n_Ψ	m_Ψ	n	k	$n-k$	n_Ψ	m_Ψ	n	k	$n-k$	n_Ψ	m_Ψ
3	1	2	6	15	4	1	3	24	96	5	1	4	120	600	6	1	5	?	?
					4	2		90	774	5	2	3	2040	33540	6	2	4	?	?
															6	3		?	?

Table C.1.: There are eight different state graphs for regular directed networks on three to six vertices. In this table G is the k-regular or $n-k$-regular directed graph on n vertices. The corresponding state graph Ψ has n_Ψ vertices and m_Ψ edges.

We never show the self-loops of the state graphs, since this complicates the visualisation. However, each state does have a non-zero probability of being repeated and hence a self-loop. The value m_Ψ^- corresponds to the number of edges in Ψ minus the self-loops, that is m_Ψ^- equals the number of edges that is visualised $(m_\Psi - n_\Psi)$.

C.1. Regular directed networks on three vertices

C.1.1. 1-regular and 2-regular networks on three vertices

State graph size: $n_\Psi = 6$ and $m_\Psi^- = 9$

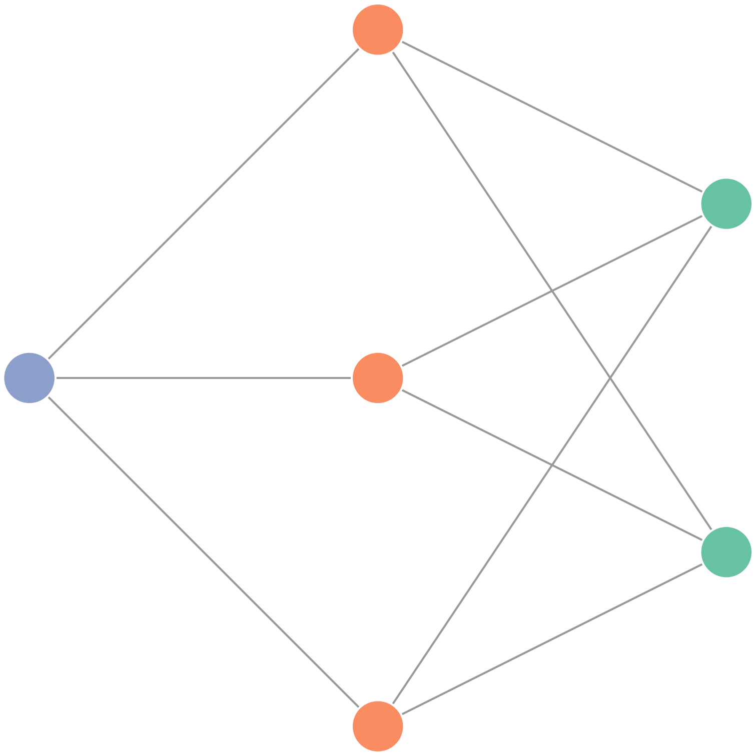


Figure C.2.: The state graph of the Curveball algorithm for 1-regular and 2-regular directed networks on three vertices. States with equal colours correspond to isomorphic networks. For simplicity we do not show self-loops. The three isomorphism classes are illustrated in Figure C.3

Isomorphism classes: There are three distinct isomorphism classes of directed 1-regular networks on three vertices. Similarly there are three distinct isomorphism classes of 2-regular directed networks on three vertices. The number of networks in each isomorphism class equals two, three, and one from left to right. Figure C.3 illustrates the different isomorphism classes.

From the perspective of 1-biregular and 2-biregular bipartite networks on six vertices, there is just a single isomorphism class. Figure C.4 illustrates the state graph for the bipartite case and the 1-biregular as well as the 2-biregular isomorphism class.

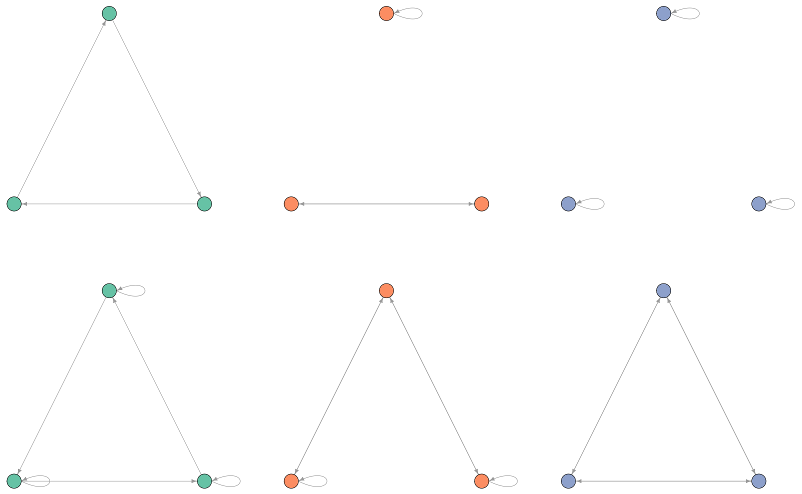


Figure C.3.: Isomorphism classes of 1-regular (top) and 2-regular (bottom) directed networks on three vertices.

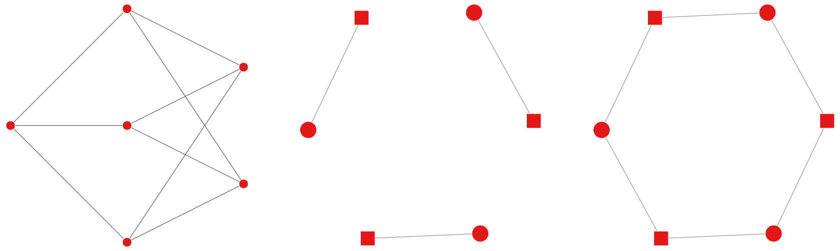


Figure C.4.: State graph of the Curveball algorithm with respect to 1-biregular and 2-biregular bipartite networks on six vertices. There is just one isomorphism class in each case (middle for 1-biregular, right for 2-biregular).

C.2. Regular directed networks on four vertices

C.2.1. 1-regular and 3-regular on four vertices

State graph size: $n_\Psi = 24$ and $m_\Psi^- = 72$

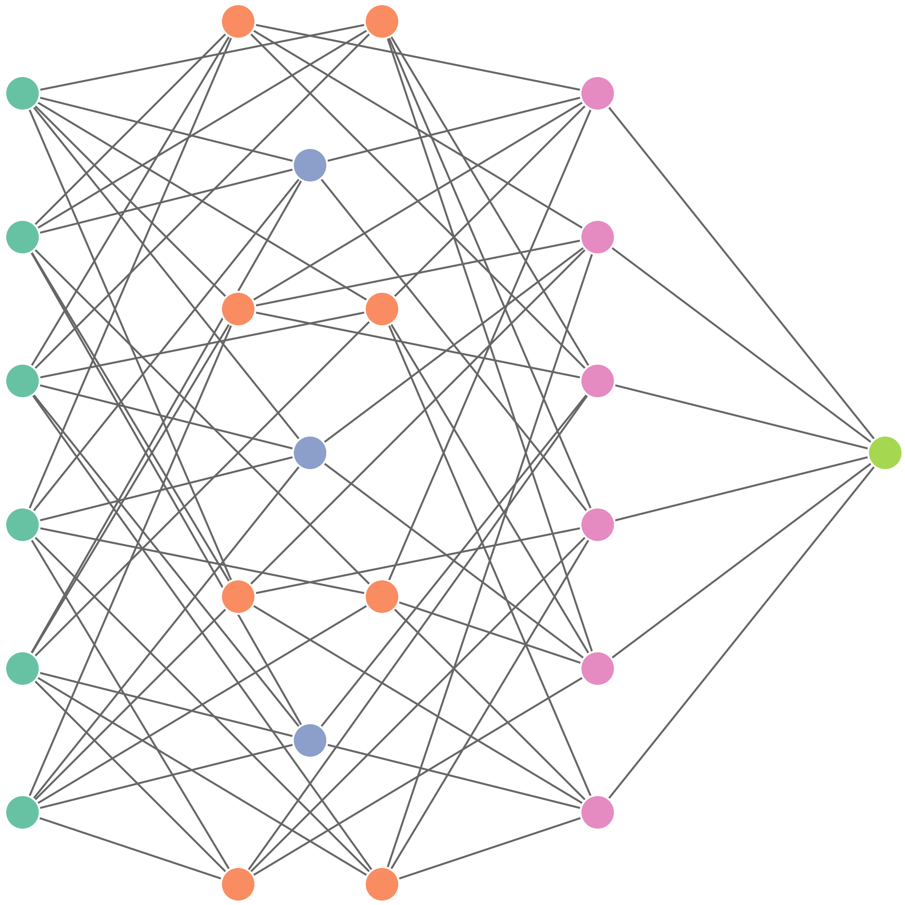


Figure C.5.: The state graph of the Curveball algorithm for 1-regular and 3-regular networks on four vertices. States with equal colours correspond to isomorphic networks. The 5 isomorphism classes are illustrated in Figure C.6

Isomorphism classes: There are 5 distinct isomorphism classes of 3-regular directed networks, see Figure C.6.

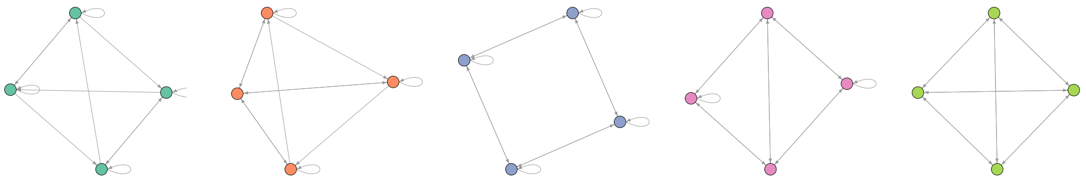


Figure C.6.: Isomorphism classes of 3-regular directed networks on four vertices.

The state graph simplifies when thinking about the adjacency matrices as the incidence matrix of bipartite networks. In this case, there is just one isomorphishm class. Figure C.7 illustrates this.

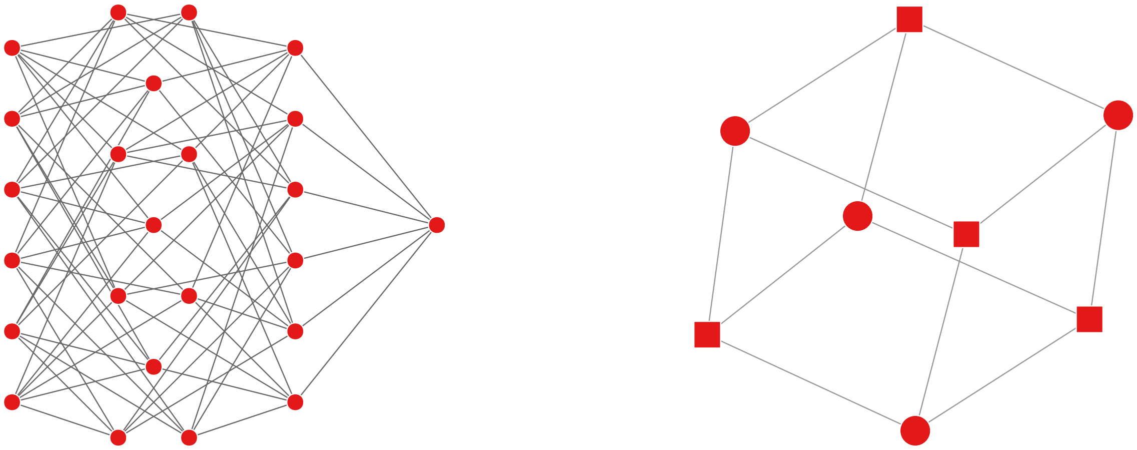


Figure C.7.: State graph of the Curveball algorithm with respect to 1-biregular and 3-biregular bipartite networks on eight vertices. There is one isomorphism class, as illustrated on the right.

C.2.2. 2-regular on four vertices

State graph size: $n_\Psi = 90$ and $m^-_\Psi = 684$

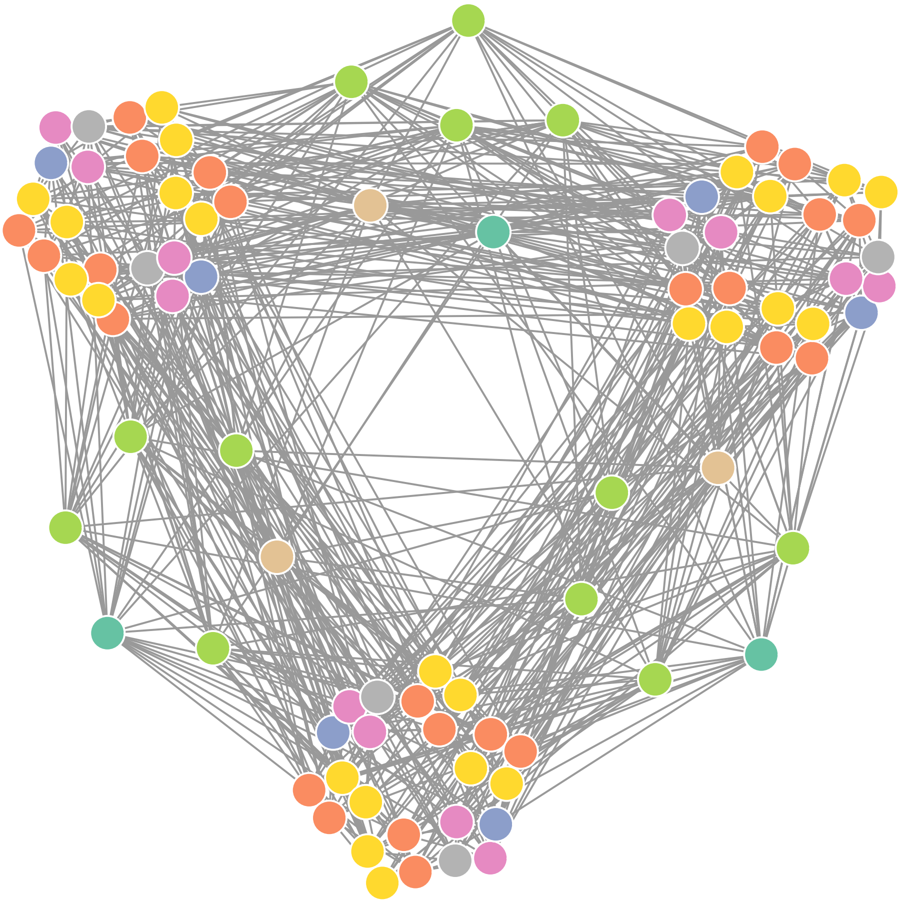


Figure C.8.: The state graph of the Curveball algorithm for 2-regular directed networks on four vertices. States with equal colours correspond to isomorphic networks. The 8 isomorphism classes are illustrated in Figure C.9

Isomorphism classes: There are 8 distinct isomorphism classes of directed networks with degree sequences $k_{in} = (2, 2, 2, 2)$ and $k_{out} = (2, 2, 2, 2)$, see Figure C.9. The number of networks in each isomorphism class equals 6, 24, 3, 12, 12, 24, 6 and 3 from left to right, top to bottom.

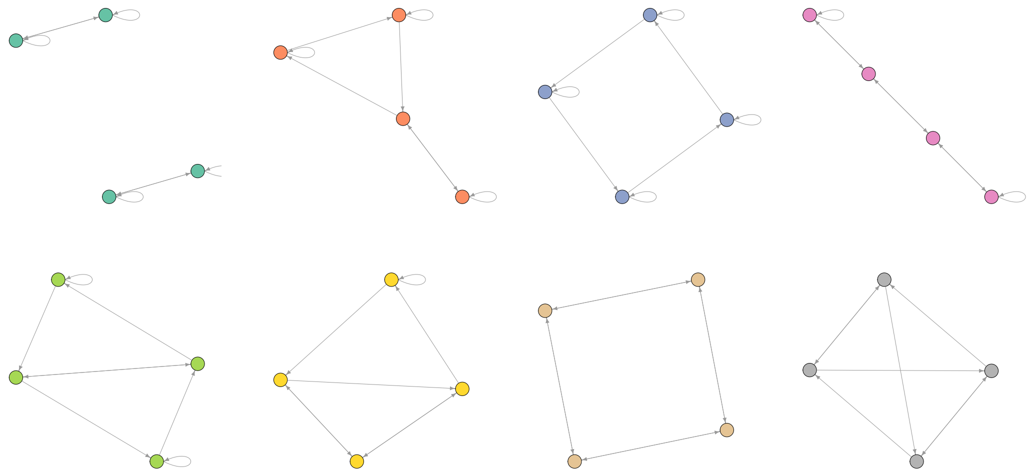


Figure C.9.: Isomorphism classes of 2-regular directed networks on four vertices.

The state graph simplifies when thinking about the adjacency matrices as the incidence matrix of bipartite networks. There are just two isomorphishm classes as illustrated in Figure C.10.

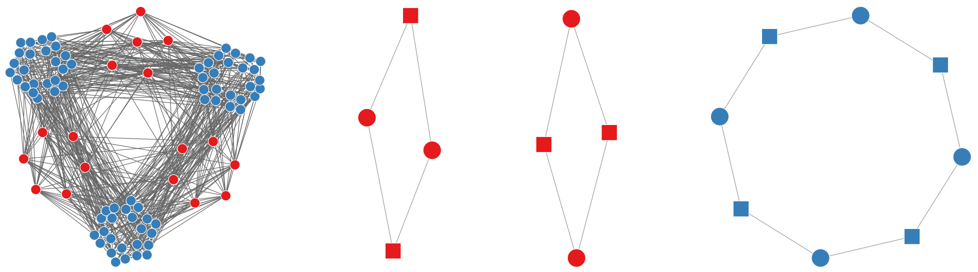


Figure C.10.: State graph of the Curveball algorithm with respect to 2-biregular bipartite networks on eight vertices. There are two isomorphism classes, as illustrated in red and blue.

C.3. Regular directed graph on five vertices

C.3.1. 1-regular and 4-regular on five vertices

State graph size: $n_{\Psi} = 120$ and $m_{\Psi}^{-} = 600$

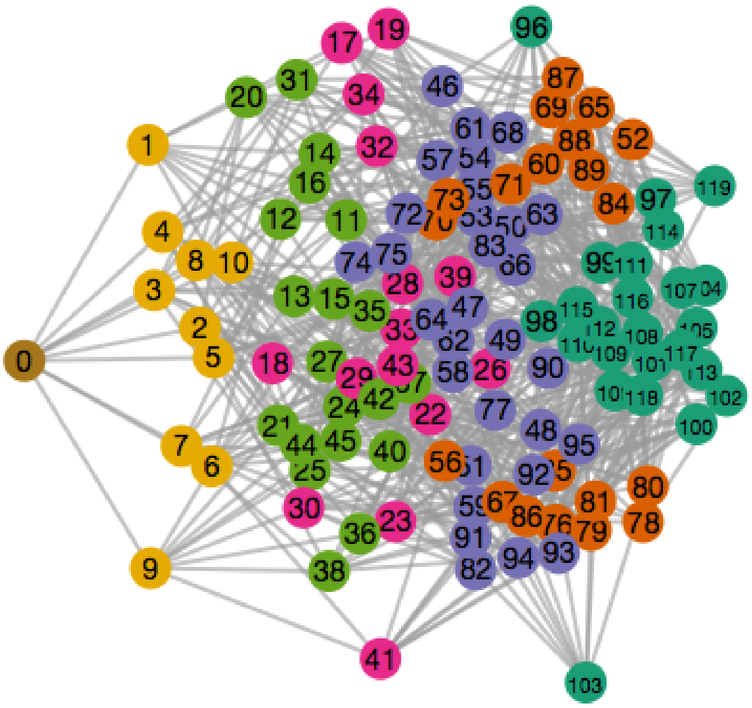


Figure C.11.: The state graph of the Curveball algorithm for 1-regular and 4-regular networks on five vertices. The seven isomorphism classes are illustrated in Figure C.12

Isomorphism classes: There are seven distinct isomorphism classes of 1-regular and 4-regular directed networks on five vertices, see Figure C.12.

The bipartite point of view is very simple, there is only 1 isomorphism class. This is not illustrated.

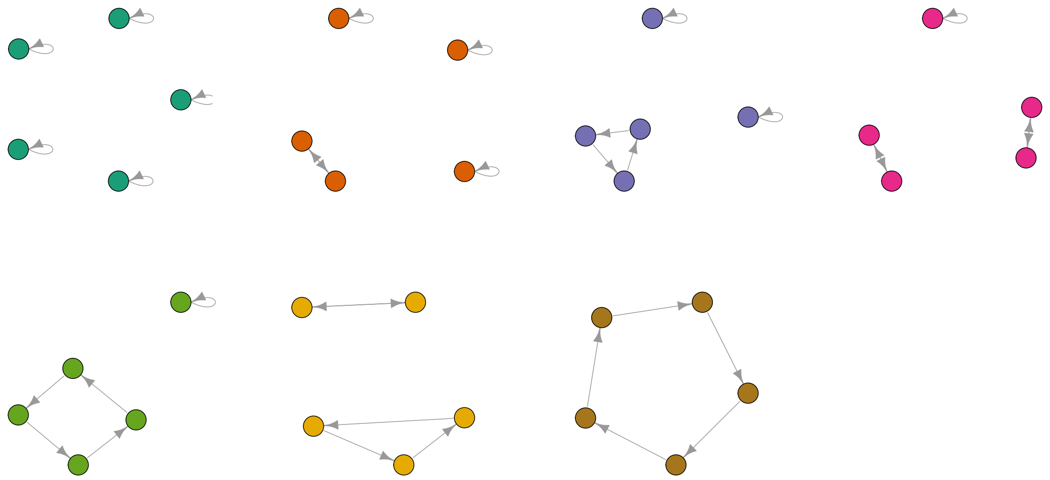


Figure C.12.: Isomorphism classes of 1-regular directed networks on five vertices.

C.3.2. 2-regular and 3-regular on five vertices

State graph size: $n_{\Psi} = 2040$ and $m_{\Psi}^{-} = 31500$
The state graph of 2-regular and 3-regular directed networks on five vertices, or 2-biregular and 3-biregular bipartite networks on ten vertices is too densely connected to profit from visualisation. However, we do see very nice symmetric structures in the 1-neighbourhood and 2-neighbourhood of each state in the state graph.

There are 18 isomorphism classes for the 2-regular and 3-regular directed networks. There are only two isomorphism classes for 2-biregular and 2-biregular networks. Figure C.13 illustrates two isomorphism classes for both the directed and the bipartite networks.

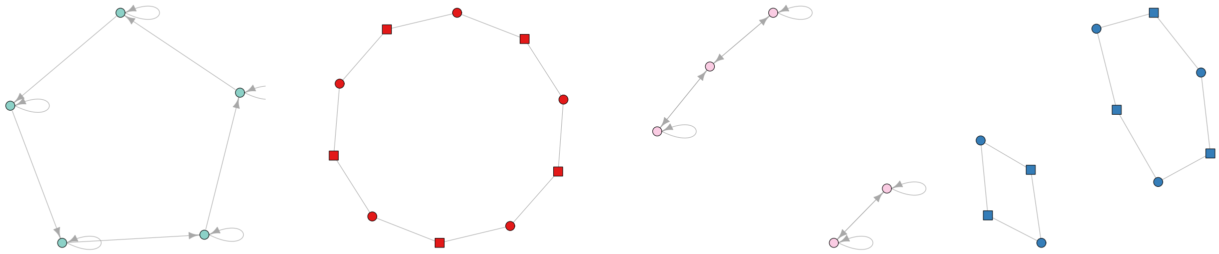


Figure C.13.: Two representatives of two isomorphism classes of 2-regular networks on five vertices and 2-biregular bipartite networks on ten vertices.

The fact that there are only 2 distinct isomorphism classes of 2-biregular bipartite networks on five vertices implies that there are also just two types of neighbourhoods in the state graph. That is the neighbourhoods of the different isomorphism classes may differ, but the neighbourhoods of isomorphic states have to be isomorphic.

Figure C.14 shows the 1-neighbourhoods of the isomorphism classes from Figure C.13. Similarly Figure C.15 shows the 2-neighbourhoods of these two isomorphism classes.

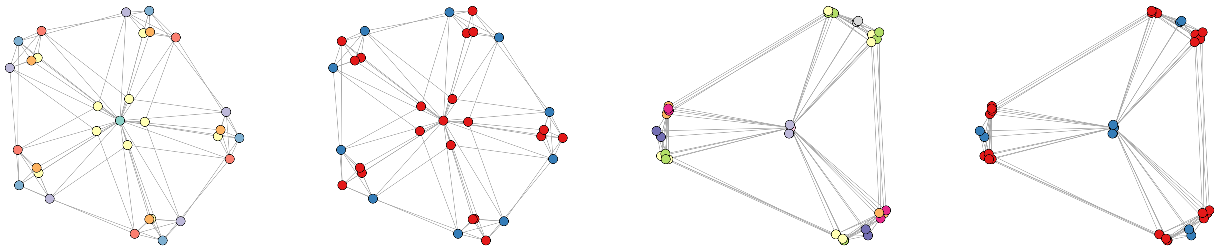


Figure C.14.: The 1-neighbourhoods of two different states in the state graph of the Curveball algorithm for 2-regular directed networks and 2-biregular bipartite networks. States with equal colours correspond to isomorphic networks.

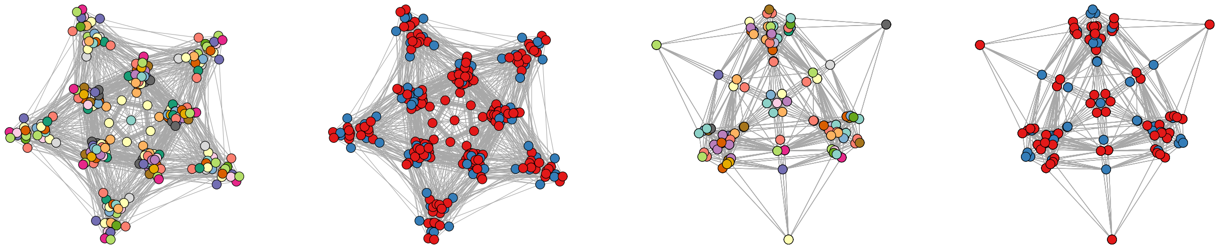


Figure C.15.: The 2-neighbourhoods of two different states in the state graph of the Curveball algorithm for 2-regular directed networks and 2-biregular bipartite networks. States with equal colours correspond to isomorphic networks.

C.4. Regular directed networks on six vertices

C.4.1. 2-regular and 4-regular networks on six vertices

State graph size: unknown

We do not know the exact size of the state graph for 2-regular networks on six vertices. Here we only investigate the state graph from a bipartite point of view. There are four isomorphism classes of 2-biregular bipartite networks on 12 vertices, these are shown in Figure C.16. The 1-neighbourhood of these four states in the state graph again show a great deal of symmetry as shown in Figure C.17.

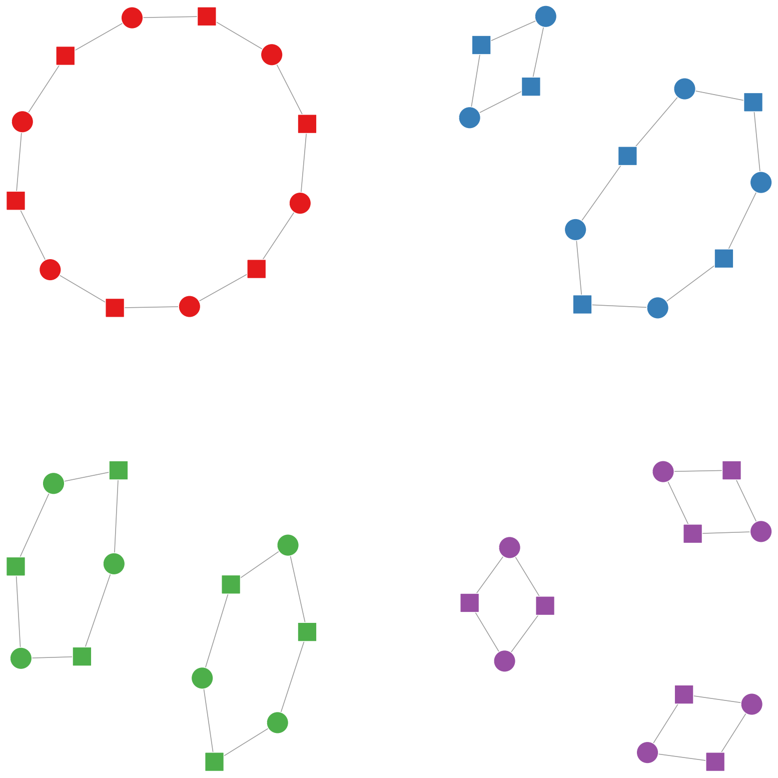


Figure C.16.: The isomorphism classes of 2-biregular bipartite networks on 12 vertices.

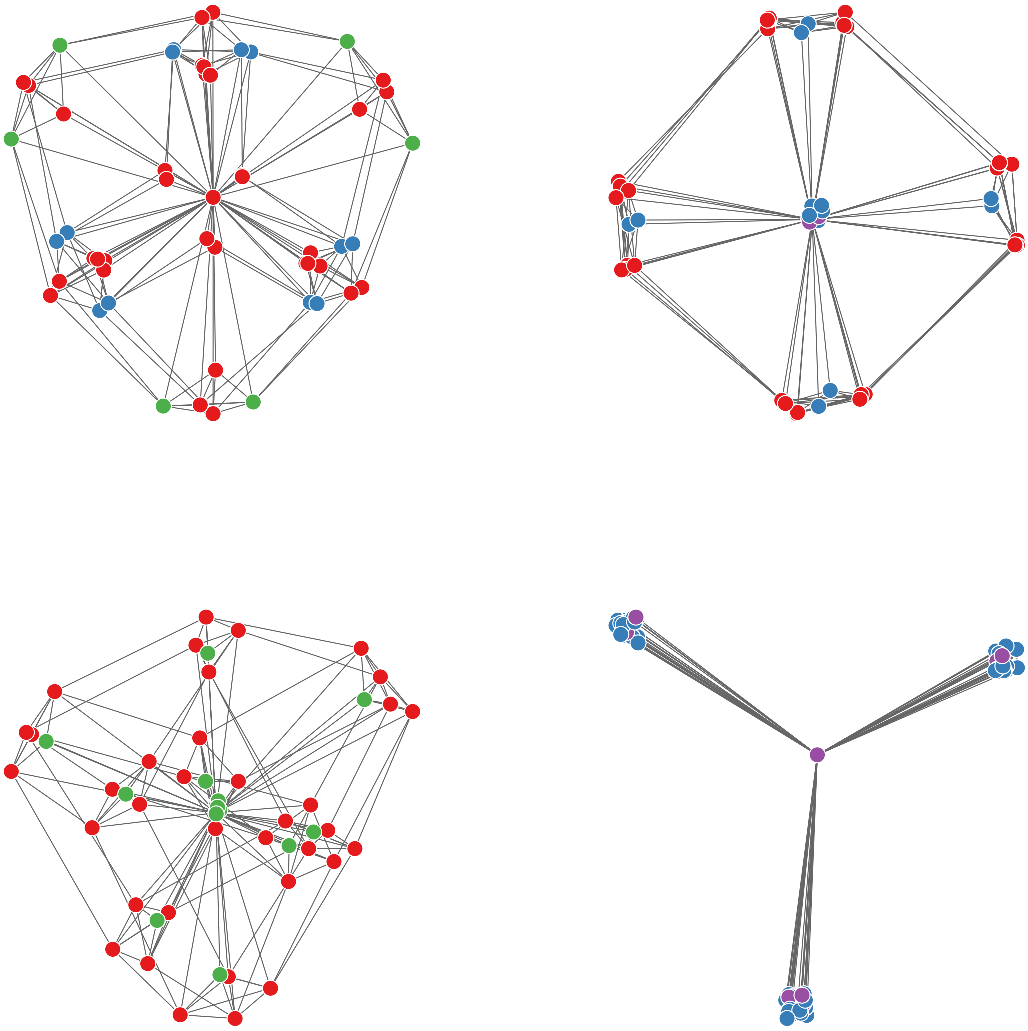


Figure C.17.: The 1-neighbourhood of the four different isomorphism classes in the state graph for 2-biregular bipartite networks on 12 vertices.

C.4.2. 3-regular networks on six vertices

State graph size: unknown

We do not know the exact size of the state graph for 3-regular networks on six vertices. Here we only investigate the state graph from a bipartite point of view. There are six isomorphism classes of 3-biregular bipartite networks on 12 vertices, these are shown in Figure C.18. The 1-neighbourhood of these six states in the state graph again show considerable symmetry as shown in Figure C.19.

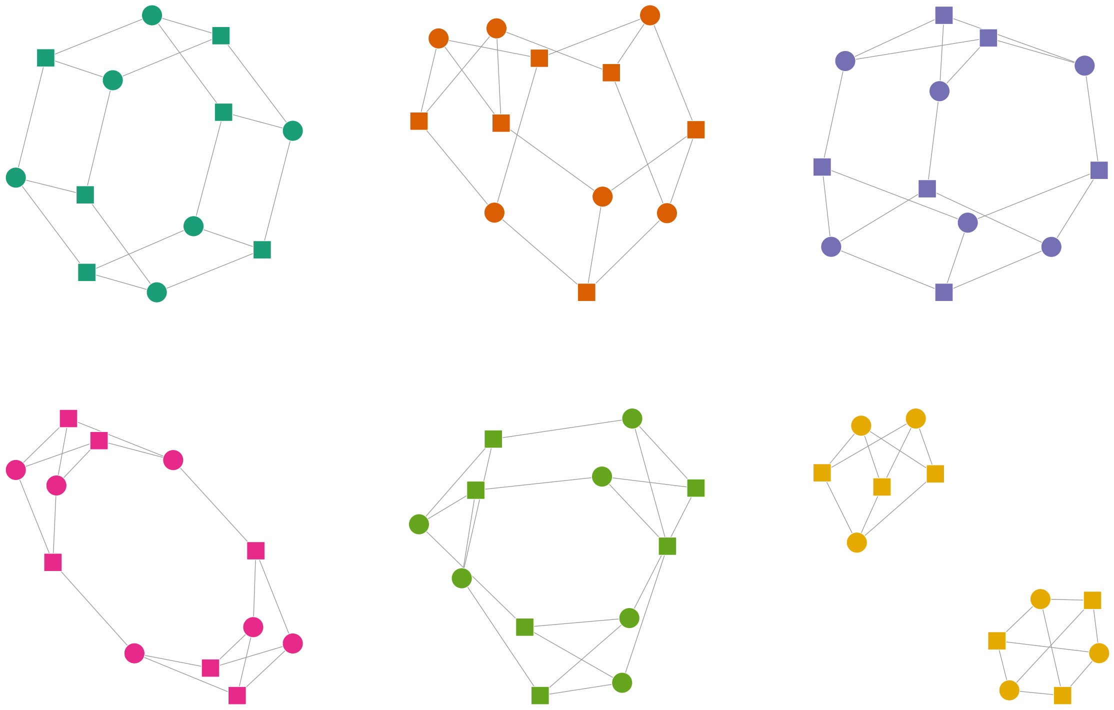


Figure C.18.: The isomorphism classes of 3-biregular bipartite networks on 12 vertices.

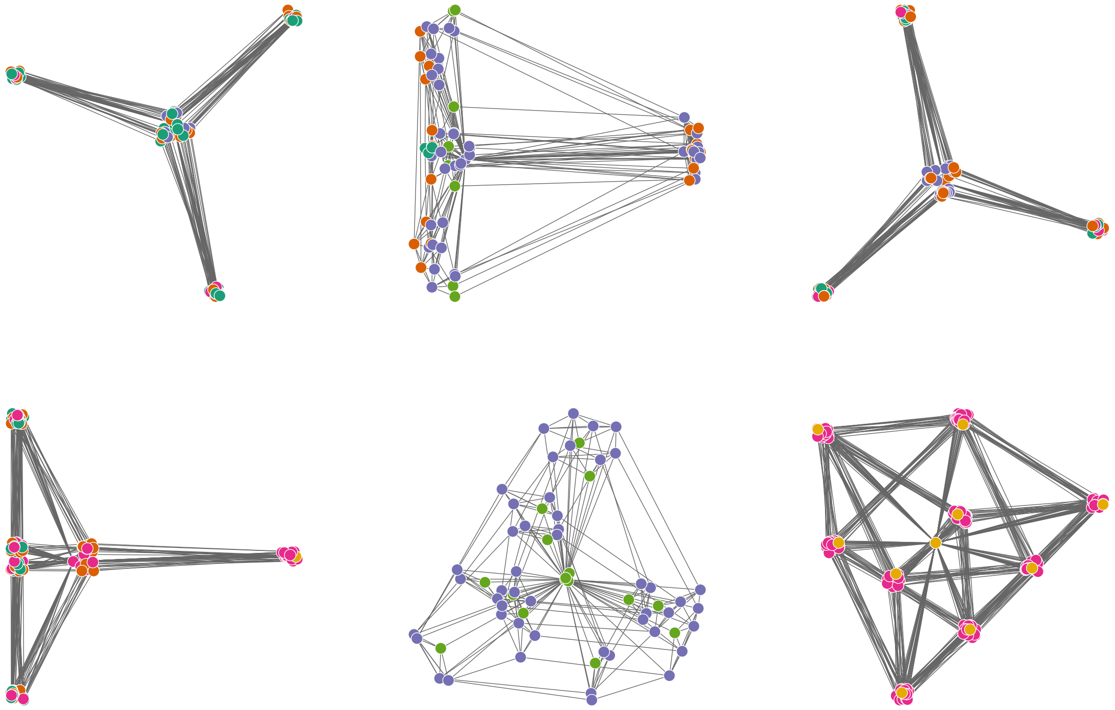


Figure C.19.: The 1-neighbourhood of the six different isomorphism classes in the state graph for 3-biregular bipartite networks on 12 vertices.

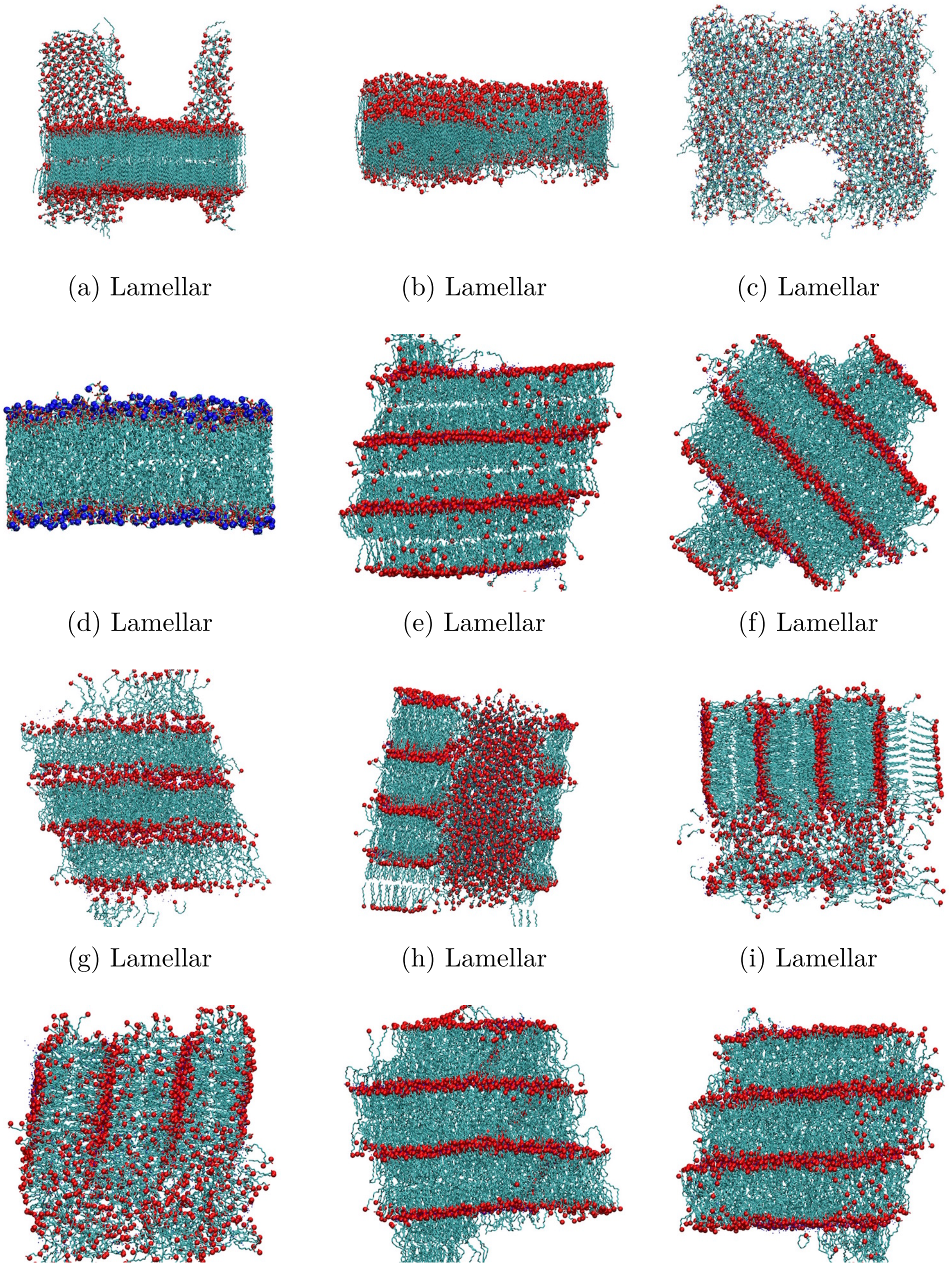
D. Lipid formulations

This appendix contains information on and visualisations of the 46 lipid formulations analysed in Chapter 7. All figures were produced by my collaborator Dr. Dallas Warren, using the VMD software package (see Appendix B.2.7).

The data set is compiled from four different sources: (A) data published in [119], (B) unpublished data from Dallas Warren, (C) [68], (D) Burri et. al. in preparation

The data set is a collection of '.gro' files. This file format is used in chemistry to represent a collection of molecules and their constituent atoms. Each line in such files represents an atom. For each atom the following information is available: the kind of atom, and the molecule that it is part, its coordinates (x, y, z) and velocity (x- ,y-, z-directions).

Table D.1 contains information on the 46 systems that we studied. The systems are illustrated in Figures D.1 - D.4.



(a) Lamellar

(b) Lamellar

(c) Lamellar

(d) Lamellar

(e) Lamellar

(f) Lamellar

(g) Lamellar

(h) Lamellar

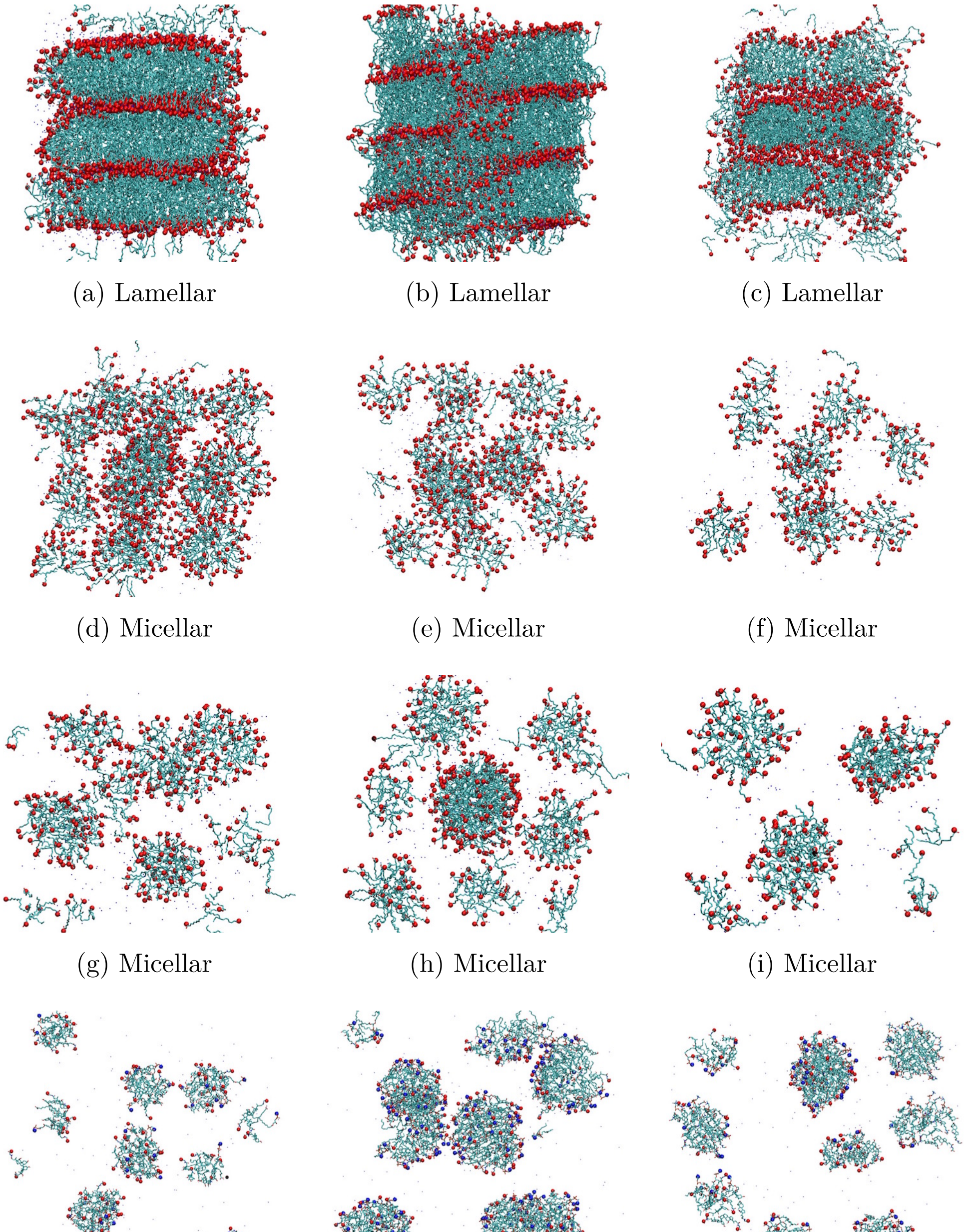
(i) Lamellar

(j) Lamellar

(k) Lamellar

(l) Lamellar

Figure D.1.: Lipid formulations 1-12, carbon in cyan, oxygen in red and nitrogen in blue. Water has been omitted for clarity.



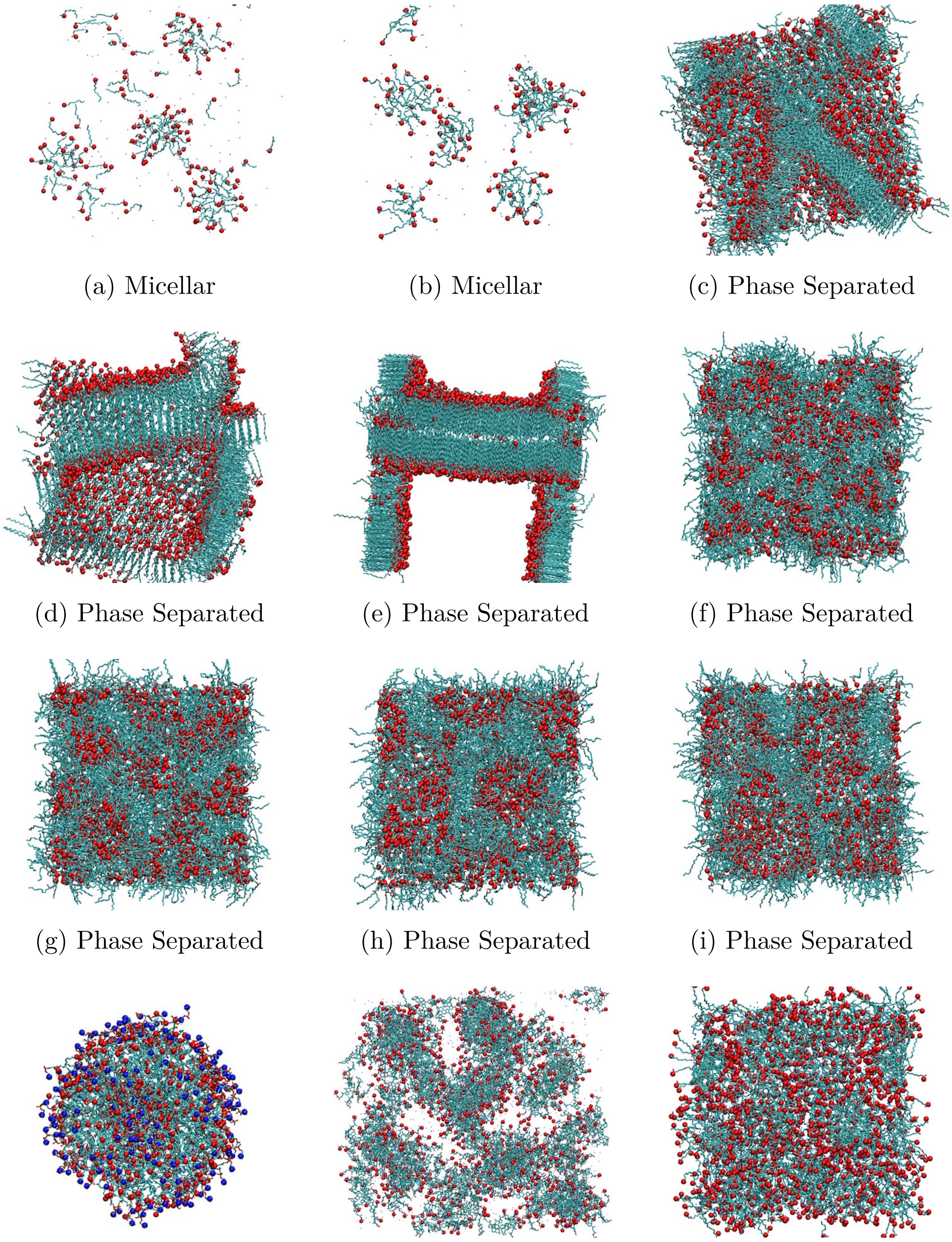
(a) Lamellar (b) Lamellar (c) Lamellar

(d) Micellar (e) Micellar (f) Micellar

(g) Micellar (h) Micellar (i) Micellar

(j) Micellar (k) Micellar (l) Micellar

Figure D.2.: Lipid formulations 13-24, carbon in cyan, oxygen in red and nitrogen in blue. Water has been omitted for clarity.



(a) Micellar

(b) Micellar

(c) Phase Separated

(d) Phase Separated

(e) Phase Separated

(f) Phase Separated

(g) Phase Separated

(h) Phase Separated

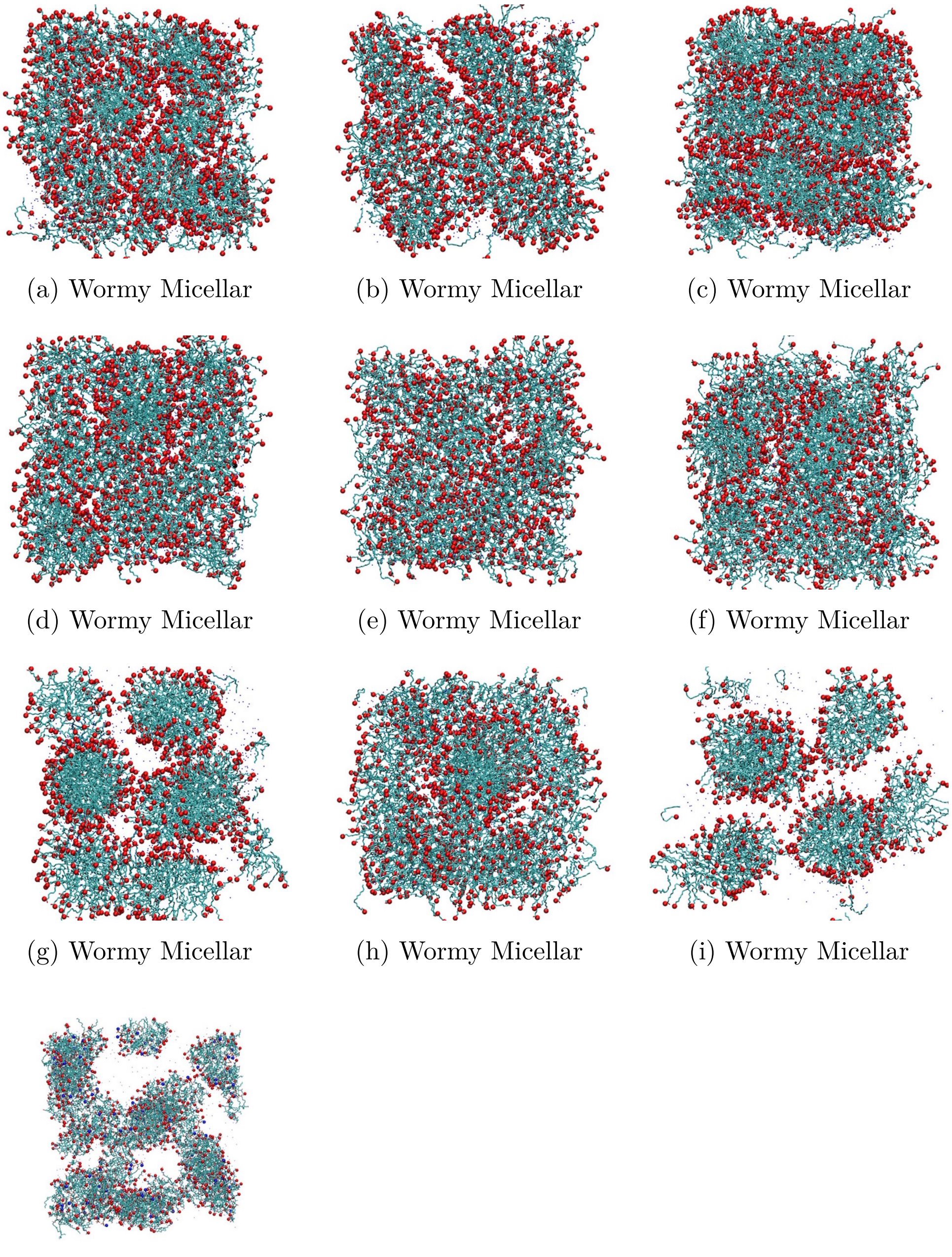
(i) Phase Separated

(j) Phase Separated

(k) Wormy Micellar

(l) Wormy Micellar

Figure D.3.: Lipid formulations 25-36, carbon in cyan, oxygen in red and nitrogen in blue. Water has been omitted for clarity.



(a) Wormy Micellar

(b) Wormy Micellar

(c) Wormy Micellar

(d) Wormy Micellar

(e) Wormy Micellar

(f) Wormy Micellar

(g) Wormy Micellar

(h) Wormy Micellar

(i) Wormy Micellar

(j) Wormy Micellar

Figure D.4.: Lipid formulations 37-46, carbon in cyan, oxygen in red and nitrogen in blue. Water has been omitted for clarity.

#	Grouping	Subgrouping	Description	Composition	Ref
1	Lamellar	Bilayer with hole	Bilayer across 2 directions	600 DGL / 600 MGL / 24,687 SOL	A
2			Bilayer with single hole	600 DGL / 600 MGL / 73,110 SOL	A
3			Bilayer with single hole	567 POPC / 89,100 SOL	B
4		Bilayer	Complete bilayer	338 POPC / 32,349 SOL	B
5		Lamellar	3 bilayers stacked	2,152 LAU / 6,624 SOL	C
6			3 bilayers stacked	1,568 OLA / 6,624 SOL	C
7			3 bilayers	1,368 OLA / 9,936 SOL	C
8		Disordered	2 sets of 3 bilayers at right angles	2,416 LAU / 3,312 SOL	C
9			3 bilayers with worm	1,800 LAU / 9,936 SOL	C
10			3 bilayers with disordered ends	1,608 LAU / 13,248 SOL	C
11			3 bilayers with disordered ends	1,204 LAU / 879 OLA / 3,300 SOL	C
12			3 bilayers with disoredered channel	1,070 LAU / 782 OLA / 6,600 SOL	C
13			3 bilayers with capped ends	970 LAU / 708 OLA / 9,075 SOL	C
14			3 bilayers with dislocation	1,760 OLA / 3,312 SOL	C
15			3 bilayers with disordered ends	1,272 OLA / 13,248 SOL	C
16	Micellar	Concentrated	Closely packed micelles	808 LAU / 22,910 SOL	C
17			Closely packed micelles	536 LAU / 26,079 SOL	C
18			Closely packed micelles	272 LAU / 29,466 SOL	C
19			Closely packed micelles	201 LAU / 146 OLA / 28,050 SOL	C
20			Closely packed micelles	400 OLA / 26,496 SOL	C
21			Closely packed micelles, bordering on dilute	200 OLA / 29,360 SOL	C
22		Dillute	Spaced micelles	118 GDX / 31 LPC / 31 OLE / 105,806 SOL	D
23			Spaced micelles	165 GDX / 192 POPC / 100,073 SOL	D
24			Spaced micelles	118 GDX / 64 POPC / 105,806 SOL	D
25			Spaced micelles	136 LAU / 31,339 SOL	C
26			Spaced micelles	96 OLA / 31,272 SOL	C
27	Phase Separated	Phase Separated	Deformed bilayers around water pools	600 DGL / 600 MGL / 6,093 SOL	A
28			Deformed bilayers around water pool	600 DGL / 600 MGL / 10,444 SOL	A
29			2 bilayers at 90o, one with hole	600 DGL / 600 MGL / 16,247 SOL	A
30		Reverse Micellar	Linked 3D channels	600 DGL / 600 MGL / 246 SOL	A
31			Linked 3D channels with pools	600 DGL / 600 MGL / 1,283 SOL	A
32			Isolated pools	600 DGL / 600 MGL / 2,708 SOL	A
33			Isolated pools, tending phase separated	600 DGL / 600 MGL / 4,301 SOL	A
34		Vesicular	Small vesicle	250 LPC / 250 OLE / 89,100 SOL	D
35	Wormy Micellar	-	Sponge like, bicontinuous	982 GDX / 86,149 SOL	C
36			Long, cylindrical micelles	1,472 LAU / 14,850 SOL	C
37			Long, cylindrical micelles	1,344 LAU / 16,560 SOL	C
38			Long, cylindrical micelles	1,072 LAU / 19,872 SOL	C
39			Long, cylindrical micelles	803 LAU / 586 OLA / 13,200 SOL	C
40			Long, cylindrical micelles	669 LAU / 488 OLA / 16,500 SOL	C
41			Long, cylindrical micelles	602 LAU / 440 OLA / 18,160 SOL	C
42			Long, cylindrical micelles	1,075 OLA / 14,850 SOL	C
43			Almost hexagonal	800 OLA / 18,096 SOL	C
44			Long, cylindrical micelles	984 OLA / 16,560 SOL	C
45			Cylindrical	584 OLA / 23,184 SOL	C
46			Long, cylindrical micelles	699 GDX / 111 POPC / 89,937 SOL	D

Table D.1.: Details for the lipid formulations studied in Chapter 7

Bibliography

[1] D. Ajwani, T. Friedrich, and U. Meyer. An algorithm for online topological ordering. *Electronic Notes in Discrete Mathematics*, 25:7–12, 2006.

[2] D. Aldous. Random walks on finite groups and rapidly mixing Markov chains. In *Séminaire de Probabilités XVII 1981/82*, pages 243–297. Springer, 1983.

[3] U. Alon. Network motifs: theory and experimental approaches. *Nature Reviews Genetics*, 8:450–461, 2007.

[4] A. Arenas, A. Fernandez, S. Fortunato, and S. Gomez. Motif-based communities in complex networks. *Journal of Physics A: Mathematical and Theoretical*, 41(22):224001, 2008.

[5] Y. Artzy-Randrup, S. J. Fleishman, N. Ben-Tal, and L. Stone. Comment on "network motifs: simple building blocks of complex networks" and "superfamilies of evolved and designed networks". *Science*, 305:1107, 2004.

[6] Y. Artzy-Randrup and L. Stone. Generating uniformly distributed random networks. *Physical Review E*, 72:056708, 2005.

[7] R. H. Atkin. From cohomology in physics to q-connectivity in social science. *International Journal of Man-Machine Studies*, 4(2):139–167, 1972.

[8] Warren D. B., Chalmers D. K., and Pouton C. W. Structure and dynamics of

glyceride lipid formulations, with propylene glycol and water. *Molecular Pharmaceutics*, 6(2):604–614, 2009.

[9] M. Bachar, P. Brunelle, D. P. Tieleman, and A. Rauk. Molecular dynamics simulation of a polyunsaturated lipid bilayer susceptible to lipid peroxidation. *Journal of Physical Chemistry B*, 108:7170–7179, 2004.

[10] L. Backstrom and J. Kleinberg. Romantic partnerships and the dispersion of social ties: A network analysis of relationship status on facebook. In *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing*, pages 831–841, New York, NY, USA, 2014. ACM.

[11] J. Bang-Jensen and G. Gutin. Alternating cycles and paths in edge-coloured multigraphs: A survey. *Discrete Mathematics*, 165-166:39–60, 1997.

[12] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[13] A. Benkouar, Y. G. Manoussakis, V. Th. Paschos, and R. Saad. On the complexity of some Hamiltonian and Eulerian problems in edge-colored complete graphs. In *ISA'91 Algorithms*, volume 557 of *Lecture Notes in Computer Science*, pages 190–198. Springer Berlin Heidelberg, 1991.

[14] S. P. Benson and J. Pleiss. Molecular dynamics simulations of self-emulsifying drug-delivery systems (sedds): Influence of excipients on droplet nanostructure and drug localization. *Langmuir*, 30(28):8471–8480, 2014.

[15] S. Benzekry, J. A. Tuszynski, E. A. Rietman, and G. Lakka Klement. Design principles for cancer therapy guided by changes in complexity of protein-protein interaction networks. *Biology Direct*, 10(1):32, 2015.

[16] A. Berger and M. Müller-Hannemann. Uniform sampling of undirected and directed graphs with a fixed degree sequence. arXiv preprint arXiv:0912.0685, 2009.

[17] A. Berger and M. Müller-Hannemann. Uniform sampling of digraphs with a fixed degree sequence. In *Graph Theoretic Concepts in Computer Science*, Lecture Notes in Computer Science, pages 220–231. Springer Berlin Heidelberg, 2010.

[18] S. Boyd, P. Diaconis, and L. Xiao. Fastest mixing Markov chain on a graph. *SIAM Review*, 46(4):667–689, 2004.

[19] P. Bubenik. Statistical topological data analysis using persistence landscapes. *Journal of Machine Learning Research*, 16:77–102, 2015.

[20] P. Bubenik and P. Dłotko. A persistence landscapes toolbox for topological statistics. *arXiv:1501.00179 [cs, math, stat]*, 2014. arXiv: 1501.00179.

[21] H. Bunke, P. J. Dickinson, M. Kraetzl, and W. D. Wallis. *A graph-theoretic approach to enterprise network dynamics.* Birkhauser, Basel, 2007.

[22] G. Carlsson. Topology and data. *Bulletin of the American Mathematical Society*, 46:255–308, 2009.

[23] G. Carlsson, T. Ishkhanov, V. de Silva, and A. Zomorodian. On the local behavior of spaces of natural images. *International Journal of Computer Vision*, 76(1):1–12, 2008.

[24] C. J. Carstens. Motifs in directed acyclic networks. In *Signal-Image Technology Internet-Based Systems (SITIS), 2013 International Conference on*, pages 605–611, 2013.

[25] C. J. Carstens. A uniform random graph model for directed acyclic networks and its effect on motif-finding. *Journal of Complex Networks*, 2:419–430, 2014.

[26] C. J. Carstens. Proof of uniform sampling of binary matrices with fixed row sums and column sums for the fast curveball algorithm. *Physical Review E*, 91:042812, 2015.

[27] C. J. Carstens and K. J. Horadam. Persistent homology of collaboration networks. *Mathematical problems in engineering*, 2013. Article ID 815035, 7 pages, 2013.

[28] S. Castano, A. Ferrara, S. Montanelli, and G. Varese. Ontology and instance matching. In *Knowledge-driven multimedia information extraction and ontology evolution*, pages 167–195. Springer, 2011.

[29] F. Chazal, D. Cohen-Steiner, L. J. Guibas, F. Mémoli, and S. Y. Oudot. Gromov-Hausdorff stable signatures for shapes using persistence. In *Computer Graphics Forum*, volume 28, pages 1393–1403. Wiley Online Library, 2009.

[30] P. Chen, H. Xie, S. Maslov, and S. Redner. Finding scientific gems with google's pagerank algorithm. *Journal of Informetrics*, 1:8–15, 2007.

[31] A. C. C. Coolen, A. De Martino, and A. Annibale. Constrained Markovian dynamics of random graphs. *Journal of Statistical Physics*, 136(6):1035–1067, 2009.

[32] P. Cunningham, M. Harrigan, G. WU, and D. O'Callaghan. Characterizing ego-networks using motifs. *Network Science*, 1(02):170–190, 2013.

[33] V. de Silva and R. Ghrist. Coverage in sensor networks via persistent homology. *Algebraic & Geometric Topology*, 7:339–358, 2007.

[34] D. Douroumis and A. Fahr, editors. *Drug Delivery Strategies for Poorly Water-Soluble Drugs*. John Wiley and Sons, 2013.

[35] H. Edelsbrunner and J. Harer. *Computational topology: an introduction*. American Mathematical Soc., 2010.

[36] H. Edelsbrunner and E. P. Mücke. Three-dimensional alpha shapes. *ACM Trans. Graph.*, 13(1):43–72, 1994.

[37] R. B. Eggleton and D. A. Holton. The graph of type $(0,\infty,\infty)$ realizations of a graphic sequence. In *Combinatorial Mathematics VI*, pages 41–54. Springer, 1979.

[38] R. B. Eggleton and D. A. Holton. Simple and multigraphic realizations of degree sequences. In *Combinatorial Mathematics VIII*, pages 155–172. Springer Berlin Heidelberg, 1981.

[39] P. Erdös and A. Rényi. On random graphs i. *Publ. Math. Debrecen*, 6:290–297, 1959.

[40] P. Erdös and A. Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hungar. Acad. Sci.*, 5:17–61, 1960.

[41] P. L. Erdös, I. Miklós, and L. Soukup. Towards random uniform sampling of bipartite graphs with given degree sequence. *arXiv preprint arXiv:1004.2612*, 2010.

[42] L. Euler. Solutio problematis ad geometriam situs pertinentis. *Commentarii Academiae Scientiarum Imperialis etropolitanae*, 8:128–140, 1736.

[43] J. Euzenat and P. Shvaiko. *Ontology matching*, volume 333. Springer, Berlin Heidelberg, 2007.

[44] R. Ghrist and A. Muhammad. Coverage and hole-detection in sensor networks via homology. In *Proceedings of the 4th international symposium on Information processing in sensor networks*, pages 254–260, 2005.

[45] M. S. Granovetter. The strength of weak ties. *American Journal of Sociology*, 78(6):1360–1380, 1983.

[46] C. Greenhill. A polynomial bound on the mixing time of a Markov chain for sampling regular directed graphs. *The Electronic Journal of Combinatorics*, 18(1):P234, 2011.

[47] B. R. Greening Jr, N. Pinter-Wollman, and N. H. Fefferman. Higher-order interactions: understanding the knowledge capacity of social groups using simplicial sets. *Current Zoology*, 61(1):114–127, 2015.

[48] O. Häggström. *Finite Markov chains and algorithmic applications.* Cambridge University Press, 2002.

[49] S. Hakimi. On realizability of a set of integers as degrees of the vertices of a linear graph. i. *Journal of the Society for Industrial and Applied Mathematics*, 10(3):496–506, 1962.

[50] A. Hatcher. *Algebraic Topology.* Cambridge University Press, Cambridge, MA, 2002.

[51] A. Hecker, C. J. Carstens, and K. J. Horadam. Neighbourhood distinctiveness: An initial study. In *Complex Networks VI*, volume 597 of *Studies in Computational Intelligence*, pages 99–110. Springer International Publishing, 2015.

[52] K. Henderson, B. Gallagher, T. Eliassi-Rad, H. Tong, S. Basu, L. Akoglu, D. Koutra, C. Faloutsos, and L. Li. Rolx: Structural role extraction & mining in large graphs. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1231–1239. ACM, 2012.

[53] P. W. Holland and S. Leinhardt. Local structure in social networks. *Sociological methodology*, 7:1–45, 1976.

[54] P. Holme and J. Saramäki. Temporal networks. *Physics Reports*, 519(3):97–125, 2012.

[55] D. Horak, S. Maletić, and M. Rajković. Persistent homology of complex networks. *Journal of Statistical Mechanics: Theory and Experiment*, page P03034, 2009.

[56] S. Itzkovitz, R. Milo, N. Kashtan, G. Ziv, and U. Alon. Subgraphs in random networks. *Physical Review E*, 68(2):026127, 2003.

[57] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data.* Prentice Hall, Englewood Cliffs, NJ, 1988.

[58] J. J. Janke, W. F. D. Bennett, and D. P. Tieleman. Oleic acid phase behavior from molecular dynamics simulations. *Langmuir*, 30(35):10661–10667, 2014.

[59] J. Jeffers, K. J. Horadam, C. J. Carstens, A. Rao, and S. Boztas. Influence neighbourhoods in CiteSeer: A case study. In *Signal-Image Technology Internet-Based Systems (SITIS), 2013 International Conference on*, pages 612–618, 2013.

[60] M. Jerrum and A. Sinclair. Approximating the permanent. *SIAM Journal on Computing*, 18(6):1149–1178, 1989.

[61] J. Jonsson. *Simplicial Complexes of Graphs*. Springer, 2007.

[62] D. Jungnickel. *Graphs, networks and algorithms.* Springer Verlag, Heidelberg, 1999.

[63] M. Kahle. Topology of random clique complexes. *Discrete Mathematics*, 309:1658–1671, 2009.

[64] A. B. Kahn. Topological sorting of large networks. *Communications of the ACM*, 5:558–562, 1962.

[65] R. Kannan. Markov chains and polynomial time algorithms. In *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on*, pages 656–671, 1994.

[66] R. Kannan, P. Tetali, and S. Vempala. Simple Markov-chain algorithms for generating bipartite graphs and tournaments. *Random Structures and Algorithms*, 14(4):293–308, 1999.

[67] B. Karrer and M. E. J. Newman. Random graph models for directed acyclic networks. *Physical Review E*, 80:046110, 2009.

[68] D. T. King, D. B. Warren, C. W. Pouton, and D. K. Chalmers. Using molecu-

lar dynamics to study liquid phase behavior: Simulations of the ternary sodium laurate/sodium oleate/water system. *Langmuir*, 27(18):11381–11393, 2010.

[69] O. D. King. Comment on "subgraphs in random networks". *Physical Review E*, 70(5):058101, 2004.

[70] H. Klein-Hennig and A. K. Hartmann. Bias in generation of random graphs. *Physical Review E*, 85(2):026101, 2012.

[71] S. Lacoste-Julien, K. Palla, A. Davies, G. Kasneci, T. Graepel, and Z. Ghahramani. Sigma: Simple greedy matching for aligning large knowledge bases. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 572–580. ACM, 2013.

[72] R. G. Laughlin. *The Aqueous phase behavior of surfactant*. Academic Press, 1994.

[73] H. Lee, M. K. Chung, H. Kang, B. N. Kim, and D. S. Lee. Computing the shape of brain networks using graph filtration and Gromov-Hausdorff metric. In *Medical Image Computing and Computer Assisted Intervention (MICCAI), 14th International Conference on*, volume 6891, pages 289–296, 2011.

[74] H. Lee, M. K. Chung, H. Kang, B. N. Kim, and D. S. Lee. Discriminative persistent homology of brain networks. In *Biomedical Imaging: From Nano to Macro, 2011 IEEE International Symposium on*, pages 841–844, 2011.

[75] H. Lee, H. Kang, M. K. Chung, B. N. Kim, and D. S. Lee. Persistent brain network homology from the perspective of dendrogram. *IEEE Transactions on Medical Imaging*, 31(12):2267–2277, 2012.

[76] S. Maletić and M. Rajković. Combinatorial Laplacian and entropy of simplicial complexes associated with complex networks. *The European Physical Journal Special Topics*, 212(1):77–97, 2012.

[77] S. Maletić, M. Rajković, and D. Vasiljević. Simplicial complexes of networks and

their statistical properties. In *Computational Science - ICCS 2008*, number 5102 in Lecture Notes in Computer Science, pages 568–575. Springer Berlin Heidelberg, 2008.

[78] S. Maslov and K. Sneppen. Specificity and stability in topology of protein networks. *Science*, 296:910–913, 2002.

[79] I. Miklós and J. Podani. Randomization of presence-absence matrices: comments and new algorithms. *Ecology*, 85:86–92, 2004.

[80] R. Milo, S. Itzkovitz, N. Kashtan, R. Levitt, S. Shen-Orr, I. Ayzenshtat, M. Sheffer, and U. Alon. Superfamilies of evolved and designed networks. *Science*, 303:1538–1542, 2004.

[81] R. Milo, N. Kashtan, S. Itzkovitz, M. E. J. Newman, and U. Alon. On the uniform generation of random graphs with prescribed degree sequences. arXiv preprint cond-mat/0312028, 2003.

[82] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: simple building blocks of complex networks. *Science*, 298:824–827, 2002.

[83] M. Mitzenmacher and E. Upfal. *Probability and computing: Randomized algorithms and probabilistic analysis.* Cambridge University Press, 2005.

[84] M. Molloy and B. Reed. A critical point for random graphs with a given degree sequence. *Random Structures & Algorithms*, 6(2-3):161–180, 1995.

[85] T. J. Moore, R. J. Drost, P. Basu, R. Ramanathan, and A. Swami. Analyzing collaboration networks using simplicial complexes: A case study. In *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on*, pages 238–243. IEEE, 2012.

[86] J. R. Munkres. *Elements of algebraic topology.* Addison-Wesley, Menlo Park, CA, 1984.

[87] F. Mémoli. Gromov–Wasserstein distances and the metric approach to object matching. *Foundations of Computational Mathematics*, 11(4):417–487, 2011.

[88] A. Narayanan and V. Shmatikov. De-anonymizing social networks. In *Security and Privacy, 2009 30th IEEE Symposium on*, pages 173–187, 2009.

[89] M. E. J. Newman. Scientific collaboration networks: I. network construction and fundamental results. *Physical Review E*, 64:016131, 2001.

[90] M. E. J. Newman. Scientific collaboration networks. ii. shortest paths, weighted networks, and centrality. *Phys. Rev. E*, 64:016132, 2001.

[91] M. E. J. Newman. The structure of scientific collaboration networks. *Proceedings of the National Academy of Sciences*, 98:404–409, 2001.

[92] M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E*, 74:036104, 2006.

[93] M. E. J. Newman. Random graphs with clustering. *Physical Review Letters*, 103:058701, Jul 2009.

[94] M. E. J. Newman. *Networks: an introduction.* Oxford University Press, 2010.

[95] M. E. J. Newman, S. H. Strogatz, and D. J. Watts. Random graphs with arbitrary degree distributions and their applications. *Physical Review E*, 64(2):026118, 2001.

[96] T. Opsahl. Triadic closure in two-mode networks: Redefining the global and local clustering coefficients. *Social Networks*, 35(2):159–167, 2013.

[97] G. Palla, I. Derényi, I. Farkas, and T. Vicsek. Uncovering the overlapping com-

munity structure of complex networks in nature and society. *Nature*, 435:814–818, 2005.

[98] P. Pedarsani, D. R. Figueiredo, and M. Grossglauser. A bayesian method for matching two similar graphs without seeds. In *Communication, Control, and Computing (Allerton), 2013 51st Annual Allerton Conference on*, pages 1598–1607. IEEE, 2013.

[99] G. Petri, P. Expert, F. Turkheimer, R. Carhart-Harris, D. Nutt, P. J. Hellyer, and F. Vaccarino. Homological scaffolds of brain functional networks. *Journal of The Royal Society Interface*, 11(101):20140873, 2014.

[100] G. Petri, M. Scolamiero, I. Donato, and F. Vaccarino. Networks and cycles: A persistent homology approach to complex networks. In *Proceedings of the European Conference on Complex Systems 2012*, pages 93–99. Springer International Publishing, 2013.

[101] G. Petri, M. Scolamiero, I. Donato, and F. Vaccarino. Topological strata of weighted complex networks. *PloS one*, 8(6):e66506, 2013.

[102] C. W. Pouton. Formulation of poorly water-soluble drugs for oral administration: physicochemical and physiological issues and the lipid formulation classification system. *European Journal of Pharmacetuical Science*, 29(3-4):278–287, 2006.

[103] A. R. Rao, R. Jana, and S. Bandyopadhyay. A Markov chain Monte Carlo method for generating random (0, 1)-matrices with given marginals. *Sankhya: The Indian Journal of Statistics, Series A*, 58:225–242, 1996.

[104] Z. Raoul. *Introduction to Surfactants and Surfactant Self-Assemblies. In Dynamics of Surfactant Self-Assemblies.* CRC Press, 2005.

[105] J. Ray, A. Pinar, and C. Seshadhri. Are we there yet? when to stop a Markov chain while generating random graphs. In *Algorithms and Models for the Web Graph*,

volume 7323 of *Lecture Notes in Computer Science*, pages 153–164. Springer Berlin Heidelberg, 2012.

[106] S. Rechner and A. Berger. ¡italic¿marathon¡/italic¿: An open source software library for the analysis of markov-chain monte carlo algorithms. *PLoS ONE*, 11(1):e0147935, 01 2016.

[107] E. S. Roberts, A. Annibale, and A. C. C. Coolen. Controlled Markovian dynamics of graphs: unbiased generation of random graphs with prescribed topological properties. In *Nonlinear Maps and their Applications*, pages 25–34. Springer, 2014.

[108] E. S. Roberts and A. C. C. Coolen. Unbiased degree-preserving randomization of directed binary networks. *Physical Review E*, 85(4):046103, 2012.

[109] V. Robins, P. J. Wood, and A. P. Sheppard. Theory and algorithms for constructing discrete Morse complexes from grayscale digital images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(8):1646–1658, 2011.

[110] H. J. Ryser. *Combinatorial mathematics.* Carus Mathematical Monographs. The Mathematical Association of America, 1963.

[111] A. Sinclair and M. Jerrum. Approximate counting, uniform generation and rapidly mixing Markov chains. *Information and Computation*, 82(1):93–133, 1989.

[112] T. A. B. Snijders. Enumeration and simulation methods for 0–1 matrices with given marginals. *Psychometrika*, 56(3):397–417, 1991.

[113] M. Sokolova and G. Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427–437, 2009.

[114] S. V. Stehman. Selecting and interpreting measures of thematic classification accuracy. *Remote Sensing of Environment*, 62(1):77–89, 1997.

[115] G. Strona, D. Nappo, F. Boccacci, S. Fattorini, and J. San-Miguel-Ayanz. A fast

and unbiased procedure to randomize ecological binary matrices with fixed row and column totals. *Nature Communications*, 5:4114, 2014.

[116] E. Szpilrajn. Sur l'extension de l'ordre partiel. *Fundamenta Mathematicae*, 16(1):386–389, 1930.

[117] R. Taylor. Constrained switchings in graphs. In *Combinatorial Mathematics VIII*, pages 314–336. Springer Berlin Heidelberg, 1981.

[118] K. Turner. Means and medians of sets of persistence diagrams. *arXiv preprint arXiv:1307.8300*, 2013.

[119] D. Warren, D. King, H. Benameur, C. Pouton, and D. Chalmers. Glyceride lipid formulations: Molecular dynamics modeling of phase behavior during dispersion and molecular interactions between drugs and excipients. *Pharmaceutical Research*, 30(12):1–16, 2013.

[120] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world'networks. *Nature*, 393:440–442, 1998.

[121] S. Wernicke. A faster algorithm for detecting network motifs. In *Algorithms in Bioinformatics*, volume 3692 of *Lecture Notes in Computer Science*, pages 165–177. Springer Berlin Heidelberg, 2005.

[122] J. G. White, E. Southgate, J. N. Thomson, and S. Brenner. The structure of the nervous system of the nematode caenorhabditis elegans. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 314(1165):1–340, 1986.

[123] A. C. Wilkerson, H. Chintakunta, H. Krim, T. J. Moore, and A. Swami. A distributed collapse of a network's dimensionality. In *2013 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 595–598, 2013.

[124] A. C. Wilkerson, T. J. Moore, A. Swami, and H. Krim. Simplifying the homol-

ogy of networks via strong collapses. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5258–5262, 2013.

[125] H. D. Williams, N. L. Trevaskis, S. A. Charman, R. M. Shanker, W. N. Charman, C. W. Pouton, and C. J. H. Porter. Strategies to address low drug solubility in discovery and development. *Pharmacological Reviews*, 65(1):315–499, 2013.

[126] J. A. Wilson, A. Bender, T. Kaya, and P. A. Clemons. Alpha shapes applied to molecular shape characterization exhibit novel properties compared to established shape descriptors. *Journal of Chemical Information and Modeling*, 49(10):2231–2241, 2009.

[127] E. Wong, B. Baur, S. Quader, and C.-H. Huang. Biological network motif detection: principles and practice. *Briefings in Bioinformatics*, 13:202–215, 2012.

[128] G. Wu, M. Harrigan, and P. Cunningham. Characterizing wikipedia pages using edit network motif profiles. In *Proceedings of the 3rd international workshop on Search and mining user-generated contents*, pages 45–52, 2011.

[129] Z. Wu, G. Menichetti, C. Rahmede, and G. Bianconi. Emergent complex network geometry. *Scientific Reports*, 5:10073, 2015.

[130] Z.-X. Wu and P. Holme. Modeling scientific-citation patterns and other triangle-rich acyclic networks. *Physical Review E*, 80:037101, 2009.

[131] Ö. N. Yaveroğlu, N. Malod-Dognin, D. Davis, Z. Levnajic, V. Janjic, R. Karapandza, A. Stojmirovic, and N. Pržulj. Revealing the hidden language of complex networks. *Scientific Reports*, 4:4547, 2014.

[132] W. W. Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33(4):452–473, 1977.

[133] J. Zhao. Expand and contract: Sampling graphs with given degrees and other combinatorial families. arXiv preprint arXiv:1308.6627, 2013.

[134] T. Zhou, J. Ren, M. Medo, and Y.-C. Zhang. Bipartite network projection and personal recommendation. *Physical Review E*, 76:046115, 2007.

[135] K. A. Zweig and M. Kaufmann. A systematic approach to the one-mode projection of bipartite graphs. *Social Network Analysis and Mining*, 1:187–218, 2011.