View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by University of Queensland eSpace


THE UNIVERSITY OF QUEENSLAND

AUSTRALIA

# Sequential Monte Carlo for random graphs

*MORGAN GRANT, B.Sc*

A thesis submitted for the degree of Master of Philosophy at
The University of Queensland in 2018
School of Mathematics and Physics

# Abstract

Real-world networks, such as the Internet, can be studied by modelling the network as a random graph. Often the process for generating such a random graph provides insight into how the real-world network was formed. For example, if the real-world network is generated by a "rich-get-richer" scheme, where graph nodes receive more connections if they are already well connected, then a good random graph model would also feature this mechanism. However, random graphs are high-dimensional, complicated objects. Computing probabilities related to the properties of random graphs are often computationally expensive or outright impossible. Furthermore, basic Monte Carlo techniques for estimating such probabilities often fail, despite producing results that appear reasonable.

This motivates the use of advanced sequential Monte Carlo techniques to estimate quantities of interest with acceptable speed and accuracy. In this thesis we investigate the application of sequential Monte Carlo to estimating tail probabilities of expected typical distance for preferential attachment random graphs, fitting a general graph process given time series data, and simulating from an exponential random graph model. We show that it is possible to construct estimators that perform better than the rudimentary Monte Carlo techniques on a variety of estimation problems. These estimators can be used by practioners to quickly obtain probabilities regarding a variety of random graph models, and therefore draw inference about the related real-world networks.

# Declaration by author

This thesis is composed of my original work, and contains no material previously published or written by another person except where due reference has been made in the text. I have clearly stated the contribution by others to jointly-authored works that I have included in my thesis.

I have clearly stated the contribution of others to my thesis as a whole, including statistical assistance, survey design, data analysis, significant technical procedures, professional editorial advice, financial support and any other original research work used or reported in my thesis. The content of my thesis is the result of work I have carried out since the commencement of my higher degree by research candidature and does not include a substantial part of work that has been submitted to qualify for the award of any other degree or diploma in any university or other tertiary institution. I have clearly stated which parts of my thesis, if any, have been submitted to qualify for another award.

I acknowledge that an electronic copy of my thesis must be lodged with the University Library and, subject to the policy and procedures of The University of Queensland, the thesis be made available for research and study in accordance with the Copyright Act 1968 unless a period of embargo has been approved by the Dean of the Graduate School.

I acknowledge that copyright of all material contained in my thesis resides with the copyright holder(s) of that material. Where appropriate I have obtained copyright permission from the copyright holder to reproduce material in this thesis and have sought permission from co-authors for any jointly authored works included in the thesis.

# Publications included in thesis

The following publication is included in Chapter 5 of this thesis:
Grant, Morgan R., and Dirk P. Kroese. "Efficient estimation of tail probabilities of the typical distance in preferential attachment models." *Winter Simulation Conference (WSC), 2016.* IEEE, 2016.

| Contributor | Contribution |
|---|---|
| Morgan Grant | Conception and design (90%)<br>Analysis and interpretation (90%)<br>Drafting and production (90%) |
| Dirk Kroese | Conception and design (10%)<br>Analysis and interpretation (10%)<br>Drafting and production (10%) |

# Submitted manuscripts included in this thesis

No manuscripts submitted for publication.

# Other publications during candidature

### Peer-reviewed papers

Grant, Morgan R., and Dirk P. Kroese. "Efficient estimation of tail probabilities of the typical distance in preferential attachment models." *Winter Simulation Conference (WSC), 2016.* IEEE, 2016.

## Contributions by others to the thesis

Robert Salamone contributed to the conception and design of the work presented in Chapter 6. Thomas Taimre contributed to the conception and design of the model in Chapter 4. Dirk Kroese provided valuable feedback and edits to the drafts of this thesis.

## Statement of parts of the thesis submitted to qualify for the award of another degree

No works submitted towards another degree have been included in this thesis.

## Research Involving Human or Animal Subjects

No animal or human participants were involved in this research.

# Acknowledgements

Special thanks to Thomas Taimre and Robert Salamone for their contributions to the fourth and sixth chapter of this thesis, respectively. Additional thanks to Liam Hodgkinson for the stimulating conversions about mathematics, probability and statistics. Finally, I would like to thank Dirk Kroese for his support and patience in supervising my MPhil studies.

# Financial support

This research was supported by an Australian Government Research Training Program Scholarship.

# Keywords

random graph, sequential monte carlo, preferential attachment

# Australian and New Zealand Standard Research Classifications (ANZSRC)

010404 Probability Theory, 40%
010406 Stochastic Analysis and Modelling, 30%
010104 Combinatorics and Discrete Mathematics, 30%

# Fields of Research (FoR) Classification

0104 Statistics, 70%
0101 Pure Mathematics, 30%

# Contents

**List of Figures** **xiii**

**List of Tables** **xv**

**1 Introduction** **1**

**2 Random Graph Theory** **5**

2.1 Graph Theory . . . . . 5

2.1.1 Graph Statistics . . . . . 10

2.2 Random Graphs . . . . . 12

2.3 Erdős-Rényi Random Graphs . . . . . 14

2.4 Preferential Attachment Models . . . . . 17

2.5 Exponential Random Graph Models . . . . . 22

**3 Monte Carlo Techniques** **25**

3.1 Splitting . . . . . 26

3.2 Importance Sampling . . . . . 27

3.3 Markov Chain Monte Carlo . . . . . 28

3.4 Stratified Splitting Algorithm . . . . . 30

**4 Generalized Dynamic Networks** **33**

4.1 Definition and Examples . . . 34

4.2 Simulation Software . . . 42

4.3 Parameter Fitting . . . 42

4.3.1 Approximate Maximum Likelihood Estimation . . . 44

4.4 Discussion . . . 47

**5 Expected Typical Distance Estimation** **51**

5.1 Theoretical Results . . . 52

5.2 Numerical Results . . . 54

5.3 Splitting . . . 58

5.3.1 Bounding Scheme . . . 58

5.4 Sequential Importance Sampling . . . 62

5.4.1 Efficient Expected Typical Distance Evaluation . . . 63

5.5 Discussion . . . 67

**6 Exponential Random Graph Model Simulation** **71**

6.1 Basic Methods . . . 72

6.1.1 Exact Simulation . . . 73

6.1.2 Markov Chain Monte Carlo . . . 76

6.2 Stratified Splitting Algorithm . . . 76

6.3 Models and Results . . . 77

6.3.1 Edge Valley . . . 78

6.3.2 Alternating Stars and Triangles . . . . . 78

6.3.3 Strictly Business . . . . . 79

6.4 Discussion . . . . . 82

**7 Conclusion** **85**

**8 Bibliography** **87**

# List of Figures


2.1 3-star example. . . . . 8

2.2 3-triangle example . . . . . 8

2.3 Realizations of the ER$(30, m)$ model . . . . . 15

2.4 Realizations of the ER$(30, p)$ model . . . . . 16

2.5 Realizations of PA$(50, 1, \alpha)$ . . . . . 18

2.6 Realizations of PA$(30, m, 0)$ . . . . . 19

4.1 Time slices of the GDN defined in Example 4.1. . . . . 36

4.2 Empirical demonstration of Example 4.1 convergence to stationarity . . . . . 37

4.3 Formation of a dominant opinion in an opinion process. . . . . 39

4.4 Realization of an opinion process after $t = 100$ time units. . . . . 39

4.5 Results from running the approximate likelihood scheme on Example 4.1 . . . . 45

4.6 Results from running the approximate likelihood scheme on Example 4.2 . . . . 46

5.1 Kernel density estimates for $\alpha = 0, 3$ preferential attachment models. . . . . 56

5.2 ETD over time for preferential attachment models. . . . . 57

5.3 Time required for each algorithm to complete (CMC, SIS and Naive SIS). . . . . 67

6.1 Estimates of the expected number of edges for the edge valley model. . . . . 78

6.2 Estimates of the expected number of 3-stars in the alternating stars/triangles model. . . . . 79

6.3 Estimates of the expected number of edges for the strictly business model. . . . 81

6.4 Estimates of the expected number of employees in the highest degree organization for the strictly business model. . . . . 81

# List of Tables

5.1 Performance comparison of estimators for $\mathbb{P}(D(G_{nm}) > \gamma)$. . . . . 66

5.2 SIS estimator results for varying parameter $p$. . . . . 66

5.3 Comparison of CMC and SIS estimation results for varying $\alpha$ and $\gamma$ . . . . . 68

6.1 Table comparing the mean squared error for various statistics of the models of interest. . . . . 82

# Chapter 1

# Introduction

Networks are prevalent in modern life, from technological networks such as the Internet and the World Wide Web [20], to biological networks including protein interaction [67] and social interaction networks for humans and other animals. By studying the structure of these networks we gain insight into how they were formed, how they might change in the future and how they can be influenced in our favour. For example, in an epidemological network a person with more social connections has more opportunities to spread an infection. The study in [18] uses the *friendship paradox* to find and monitor these well-connected individuals in an effort to detect epidemics sooner than simply studying randomly chosen people from the population.

Real-world networks of interest are often too difficult to study directly — they are expensive to study accurately and change frequently. Furthermore, it is unethical to experiment with many real-world networks, such as introducing an infection into a population to study its rate of propagation. Rather than working with real-world networks, it is possible to generate random graph models to emulate reality. This alleviates the aforementioned problems and motivates studying random graph theory, an emerging field of mathematical study.

A major random graph model of interest was introduced by Barabási and Albert in 1999, demonstrating the power of *preferential attachment* for modelling real-world networks; see [7]. Informally, preferential attachment describes any mechanism for building a graph where the probability of attaching an edge to a vertex is dependent on the degree of said vertex.

The Barabási–Albert model gained attention due to its ability to create graphs with degree sequences that follow a *power law* — a property found in many real-world networks [25].

Specifically, the probability that a vertex has a particular degree $d$ is proportional to $d^{-3}$. This contrasts with earlier graph models such as the Erdős–Rényi [24] and Waxman [79] random graphs, which do not have power-law degree sequences, and thus are unsuitable for modelling real-world networks similar to the World Wide Web [55].

A key result for Barabási–Albert models concerns the *typical distance*, which can be thought of as the expected number of 'hops' between a randomly chosen pair of connected vertices. The asymptotic typical distance was found to bounded by an $O(\log n)$ function, where $n$ is the number of vertices in the graph [11]. This is a very important result, as it satisfies the *small-world* criterion. The precise definition of a small-world network varies in the literature; informally we describe it as a network for which the average path length between any two vertices is of the order $\log n$. This small-world phenomenon is exhibited by many real-world networks, such as social networks [64], the World Wide Web [2], and neurological networks [78].

While the asymptotic result for the typical distance is well studied, its distribution is not. In this thesis we will use kernel density estimates [66] to study the probability that the expectation of the typical distance is larger than some constant level $\gamma$. As this becomes a rare-event probability for large $\gamma$, we use the Monte Carlo techniques of *sequential importance sampling* [45] and *splitting* [33] to reduce the variance of our estimators. The application of these techniques to random graphs has not been studied in-depth, providing new challenges.

The drawback of the preferential attachment models is that they do not explicitly create substructures that are common in real-world networks, such as cliques and triangles [3]. The *exponential random graph* (ERG) model [61, 77] (alternatively called the p* model) is a family of random graph distributions where substructures of interest can be directly weighted, allowing for a high level of control over the random graph's behaviour. When observed network data are available, maximum likelihood estimation techniques applied to the model can give statistical evidence that the occurrence of a substructure is more than just random chance. Applications of the model include hierarchical networks [76], social networks in a school environment [37] and business connections [61].

The ERG model operates on a vertex set of fixed size. A random graph $G$ in the ERG family has a density where the log-probability of each graph is proportional to the linear combination of selected graph statistics multiplied by model parameters. See Definition 2.9 for the specific formulation. By choosing appropriate graph statistics and weighting them as desired, graphs

with the desired traits will appear with high probability. For example, if a statistic which counts the number of components in a graph is weighted negatively, connected graphs would be generated with greater frequency.

A major drawback for the ERG model is that there is currently no way to exactly simulate from non-trivial models. Calculating the normalizing constant for an ERG on $n$ vertices requires $2^{\binom{n}{2}}$ calculations, which is completely infeasible for modern computers when $n > 10$. An alternative approach is to get approximate samples using *Markov chain Monte Carlo* (MCMC) [68]. This method has questionable efficacy, as it was demonstrated in [9] that the time required to obtain accurate samples using MCMC scales exponentially with the number of vertices in the graph.

In this thesis we apply the *stratified splitting algorithm* (SSA) [73] to the ERG model to create a method which is faster than full enumeration and more reliable than MCMC. We demonstrate that for small models SSA is the superior choice for general ERG model specifications. Larger models tend to require too many splitting levels which causes the algorithmic error to explode.

The dynamical behaviour of real-world networks is not captured by ERG models. Furthermore, models such as preferential attachment are too simple for accurate model simulation. These problems led to the creation of the *generalized dynamic network* (GDN) model [71], a Markovian, continuous-time weighted graph process. The weights are usually a vector of real numbers associated with each vertex, which allows for much more sophisticated models. For example, the weights can create a susceptible-infected-susceptible epidemic model running atop the random graph. Analytical work on similar graph processes is seen in [38] and [13]. Our work in Chapter 4 focusses on simulations and applications, allowing greater flexibility in how the weights evolve over time.

When dynamically adding and removing vertices is permitted, the state space and transitions become very complicated, as each state of the Markov chain is described by an adjacency matrix whose size depends on the current number of vertices. Work in [71] has concluded that handling these vertex transitions, along with general edge transitions, makes analytical results too difficult to achieve with current techniques. This necessitates simulation studies.

Preliminary work in [75] demonstrates the power of the model by using discrete event simulation to embed a contact process evolving over the graph simultaneously with the graph

being rewired. The results revealed that a contact process evolving over changing network spreads slower compared to the same contact process on a static network. The discrepancy highlights how GDN can provide new insight into natural phenomena such as viruses in social networks.

This thesis introduces a GDN package written for C++. This package is designed to allow for swift construction and observation of models by abstracting out many tedious parts of writing a simulation. A special software companion was also developed to allow for the dynamic graphs to be visualized, as existing software lacked the features we desired. The package also features a filtering algorithm for estimating the likelihood of a model given data snapshots. This can then be used with a noisy optimiser such as the cross-entropy method [45] to provide a maximum likelihood estimate.

# Chapter 2

# Random Graph Theory

The study of random graphs can be traced back to [24], where the random graph model was introduced as a tool for proving graph theoretic results. However, it was the paper by Barabási and Albert [7] which ignited study into random graphs for modelling real-world complex network behaviour.

As noted in [23] and [74] many papers on random graphs lack mathematical rigour. For example, the original paper [7] by Barabási and Albert does not specify how to begin the construction of the random graph. Other incidences involve using a mean-field approximation to obtain results, despite the mean-field approach being designed for latices rather than the more chaotic random graphs of interest [36]. To address this, we open the chapter by describing what a graph is and what statistics are of interest. This is then followed by a precise definition of random graphs and random graph processes. These sections form the fundamentals for the rest of the chapter which defines and explains the three most relevant random graphs for this thesis: those by Erdős-Rényi, Barabási-Albert and the exponential random graph.

## 2.1 Graph Theory

We begin with formal definitions of the graph theory nomenclature that will be used throughout this thesis.

**Definition 2.1.** A (*simple*) *graph* $g = (\mathcal{V}, \mathcal{E})$ is a pair of two sets: the *vertex* set $\mathcal{V} = \{1, 2, \ldots, n\}$ and the *edge* set $\mathcal{E} = \{\{v_1, v_2\}, \{v_3, v_4\}, \ldots\}$, for all $v_i \in \mathcal{V}$. For each $\{v_i, v_{i+1}\} \in \mathcal{E}$, $v_i \neq v_{i+1}$. ♦

As $\mathcal{E}$ is a set, duplicate edges are not permitted (See Definition 2.2 for *multigraphs* which permit duplicate edges). Every graph considered in this thesis will be undirected. We denote the set of all such graphs by $\mathcal{G}$, and let $\mathcal{G}_i$ be the set of all graphs on $i$ vertices. Some fields refer to graphs as *networks*, vertices as *nodes* and edges as *ties*. Note that our use of lower-case letters to denote graphs is non-standard; this is to ensure a clear distinction between deterministic and random graphs.

If $\{u, v\} \in \mathcal{E}$ we say that the vertices $u$ and $v$ are *adjacent*. For any pair of vertices, either an edge exists between them or it does not. Thus the structure of a graph can be encoded in a symmetric, binary matrix. Such a matrix $A = \{a_{ij}\}$ is called an *adjacency matrix*. As Definition 2.1 ensures that the vertices have integer labels, the vertices themselves can be used directly as the indices for the adjacency matrix. If vertices $v$ and $u$ are adjacent, then $a_{uv} = a_{vu} = 1$. Similarly, if $v$ and $u$ are not adjacent, then $a_{uv} = a_{vu} = 0$. Note that we do not consider $v$ to be adjacent with itself, so the diagonal of $A$ will be all zeroes.

Take $\mathcal{M}$ to be the set of all symmetric, binary matrices of finite size and zero diagonal. Let $\text{Adj} : \mathcal{G} \to \mathcal{M}$ be the function which maps a graph to the unique adjacency matrix that has columns and rows ordered by vertex value. We are often interested in the connectivity behaviour of a given vertex $v$. The set of vertices adjacent to $v$ in graph $g$ is called the *neighbourhood* of $v$, and is denoted by $\mathcal{N}_g(v)$. Another important concept is the *degree* of $v$,

$$\deg_g(v) := \sum_{u \in \mathcal{V}} \text{Adj}(g)_{uv}. \tag{2.1}$$

The *degree sequence* is the vector of all vertex degrees; that is, $(\deg_g(1), \deg_g(2), \ldots, \deg_g(n))$. Comparing the degree sequences of two graphs can provide insight into their structural differences.

An ordered sequence of vertices $u_1, u_2, \ldots, u_k$ is called a *path* if $u_i$ and $u_{i+1}$ are adjacent for $i \in \{1, \ldots, k-1\}$. The *length* of this path is $k - 1$; counting the edges traversed, not the number of vertices. The *distance* between two vertices $u$ and $v$ in graph $g$, denoted $d_g(u, v)$, is the length of the shortest path from $u$ to $v$. If there is no path from $u$ to $v$ then by convention $d_g(u, v) = \infty$.

A *cycle* is a sequence of vertices $u_1, u_2, \ldots, u_k, u_1$ that has consecutive vertices adjacent and all $u_i, i \in \{1, 2, \ldots, k\}$ are unique. Of interest to this thesis are graphs without cycles, called *trees*.

**Theorem 2.1.** *For vertices* $u, v, w \in \mathcal{V}_g$,

*i)* $d_g(v, v) = 0$
*ii)* $d_g(u, v) = d_g(v, u)$
*iii)* $d_g(u, v) + d_g(v, w) \geq d_g(u, w)$.

*Proof.* For (i), the shortest path from $v$ to itself is contains only $v$. Thus $d_g(v, v) = 0$ directly from definition. To show (ii), we simply note that edges are undirected, so all paths from $u$ to $v$ can be reversed into paths from $v$ to $u$ and vice versa. It follows that $d_g(u, v) = d_g(v, u)$. Finally for (iii), as there exists a path between between $u$ and $w$ of length $d_g(u, v) + d_g(v, w)$, we have an upper bound on shortest path between $d_g(u, w)$. □

We say that $\mathcal{C}_g(v) = \{u \mid d_g(v, u) < \infty\}$ is the *component* of vertex $v$. Furthermore, $u$ and $v$ are said to be *connected* if $\mathcal{C}_g(v) = \mathcal{C}_g(u)$. As only undirected graphs will be considered in this thesis, $u \in \mathcal{C}_g(v)$ implies that $\mathcal{C}_g(v) = \mathcal{C}_g(u)$.

Often the symbols we use to represent each vertex are not important; they are merely identification tags. If a graph $g = (\mathcal{V}_g, \mathcal{E}_g)$ is a simple vertex relabelling of another graph $h = (\mathcal{V}_h, \mathcal{E}_h)$, we say these graphs are *isomorphic*. Put formally: $g$ and $h$ are isomorphic if there exists a function $f : \mathcal{V}_g \to \mathcal{V}_h$ where for all $u, v \in \mathcal{V}_g$, $\{u, v\} \in \mathcal{E}_g$ if and only if $\{f(u), f(v)\} \in \mathcal{E}_h$. That is, $f$ is an adjacency-preserving bijection.

If $\mathcal{V}_g \subset \mathcal{V}_h$ and $\mathcal{E}_g \subset \mathcal{E}_h$ then $g$ is a *subgraph* of $h$, written $g \trianglelefteq$ h. We will say $g^*$ is an isomorphic subgraph of $h$ if $g^*$ is isomorphic to some $g$ which satisfies $g \trianglelefteq h$. A *k-star* is a subgraph of $g$ that is isomorphic to a graph where one vertex has degree $k$, all others have degree one. See Figure 2.1 for an example. A *k-triangle* is a subgraph of $g$ that is isomorphic to a graph where there exists an edge $\{v, u\}$ and $v, u$ are mutually adjacent to $k$ other vertices. See Figure 2.2 for an example.

Sometimes it is prudent to relax the definition of a graph by changing the edge set, as well as edges themselves, to multisets.

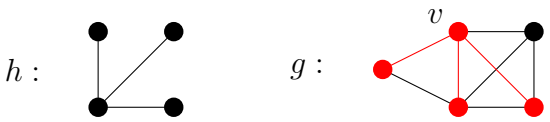



Figure 2.1: The graph $h$ is an example of a 3-star. In graph $g$ there are four 3-stars with $v$ as the central vertex, one of which is highlighted.

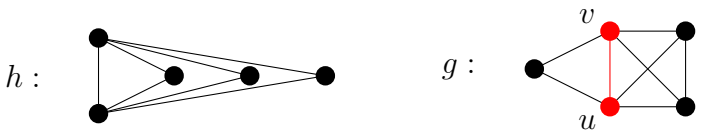



Figure 2.2: The graph $h$ is an example of a 3-triangle. There is one 3-triangle in the entirety of graph $g$ — $u$ and $v$ are mutually adjacent to every other vertex in the graph.

**Definition 2.2.** A *multigraph* $g = (\mathcal{V}, \mathcal{E})$ is a pair of two sets: the *vertex* set $\mathcal{V} = \{1, 2, \ldots, n\}$ and the *edge* multiset $\mathcal{E} = \{\{v_1, v_2\}, \{v_3, v_4\}, \ldots\}$, for some $v_i \in \mathcal{V}$. The edges $\{v_i, v_{i+1}\}$ are multisets, so $v_i = v_{i+1}$ is permitted. ♦

Edges of the form $\{v, v\}$ are often called *self-loops* or just *loops*. In a multigraph *parallel* edges may be present. These are edges which occur more than once in the the edge multiset. We will let the set of all multigraphs be denoted $\mathcal{G}^*$. As every simple graph satisfies the definition of multigraph, $\mathcal{G} \subset \mathcal{G}^*$. Furthermore, unless stated otherwise, all results and statistics which apply to multigraphs also apply to simple graphs.

The definition of adjacency is the same for multigraphs as it is for simple graphs — $u$ and $v$ are adjacent if $\{u, v\} \in \mathcal{E}$. However, the adjacency matrix changes significantly. Consider adjacency matrix $A = \{a_{ij}\}$ of multigraph $g = (\mathcal{V}, \mathcal{E})$. Then for distinct vertices $u, v \in \mathcal{V}$, $a_{uv}$ equals the number of times $\{u, v\}$ appears in $\mathcal{E}$. Special consideration is given to the diagonal elements $a_{vv}$, which now equals *double* the number of times $\{v, v\}$ appears in $\mathcal{E}$. We can extend the adjacency matrix function to be $\text{Adj} : \mathcal{G}^* \to \mathcal{M}^*$, where $\mathcal{M}^*$ is the set of all symmetric, non-negative integer matrices with even diagonal. The definition of degree follows directly from Equation 2.1, with $g$ now a multigraph and Adj the extended form of the adjacency matrix function.

The following theorem implies that working in the adjacency matrix space is equivalent to working with the graph objects directly.

**Theorem 2.2.** *The function* $\text{Adj} : \mathcal{G}^* \to \mathcal{M}^*$ *is invertible.*

*Proof.* Consider the construction of a function $\text{Adj}^{-1} : \mathcal{M}^* \to \mathcal{G}^*$. The goal is to construct said function so that $g = \text{Adj}^{-1}\text{Adj}(g)$.

For all $A \in \mathcal{M}$ define $\text{Adj}^{-1}(A) = h = (\mathcal{V}_h, \mathcal{E}_h)$, where $\mathcal{V}_h = \{1, 2, \ldots, m\}$ and $m$ is the number of rows of $A$. Next set

$$\mathcal{E}_h = \left( \bigcup_{1 \le v \le m} \bigcup_{i=1}^{A_{vv}/2} \{ \{v, v\} \} \right) \cup \left( \bigcup_{1 \le u < v \le m} \bigcup_{i=1}^{A_{uv}} \{ \{u, v\} \} \right). \tag{2.2}$$

That is, we construct a multiset which contains the edge $\{u, v\}$ repeated $A_{uv}$ times (or repeated $A_{vv}/2$ times for self-loops). Given a graph $g = (\mathcal{V}_g, \mathcal{E}_g)$ of order $n$, consider $h = \text{Adj}^{-1}\text{Adj}(g)$. We have $\mathcal{V}_g = \mathcal{V}_h$ trivially, thus $m = n$. Moreover, for all distinct $u, v \in \mathcal{V}_h$, $|\{ \{u, v\} : \{u, v\} \in \mathcal{E}_h\}| = |\bigcup_{i=1}^{A_{u,v}} \{ \{u, v\} \}| = A_{uv} = |\{ \{u, v\} : \{u, v\} \in \mathcal{E}_g\}|$. As order is not important for multiset equality, $\mathcal{E}_h = \mathcal{E}_g$. □

The definition of multigraph isomorphism extends and encapsulates the definition for simple graph isomorphism. Multigraphs $g$ and $h$ are isomorphic if there exists a function $f : \mathcal{V}_g \to \mathcal{V}_h$ where for all $u, v \in \mathcal{V}_g$, $\text{Count}(\{u, v\}, \mathcal{E}_g) = \text{Count}(\{f(u), f(v)\}, \mathcal{E}_h)$. The $\text{Count}(a, \mathcal{B})$ function simple counts the number of times element $a$ appears in multiset $\mathcal{B}$.

We conclude this section with a relevant and surprising theorem which gives mathematical support to a common observation: "my friends have more friends than me."

**Theorem 2.3** (Friendship Paradox)**.** *Let $g = (\{1, 2, \ldots, n\}, \mathcal{E}_g)$ by any finite multigraph with degree sequence $(\deg_g(1), \deg_g(2), \ldots, \deg_g(n))$. Let $V$ be a random vertex chosen uniformly from the vertex set of $g$ and $U$ be a random vertex chosen uniformly from $\mathcal{N}_g(V)$. Then $\mathbb{E}\ \deg_g(V) \le \mathbb{E}\ \deg_g(U)$.*

*Proof.* We follow the proof given in [74]. For notational simplicity, let $\deg(v) = \deg_g(v)$. The joint probability mass function is found through applying the law of total probability followed by Bayes:

$$\begin{aligned}
\mathbb{P}(\deg(V) = k, \deg(U) = l) &= \sum_{(u,v)} \Big( \mathbb{P}(\deg(V) = k, \deg(U) = l \mid U = u, V = v) \\
&\qquad \times \mathbb{P}(U = u)\mathbb{P}(V = v \mid U = u) \Big) \\
&= \sum_{(u,v)} \mathbb{I}\{\deg(u) = k, \deg(v) = l\}(n\deg(u))^{-1},
\end{aligned}$$

where the sums are taken over all edges but with order induced. That is, if $\{u, v\} \in \mathcal{E}$, then $(u, v)$ and $(v, u)$ will appear in the sum. Clearly

$$\mathbb{E}\ \deg(U) = \sum_l l \sum_k \sum_{(u,v)} \mathbb{I}\{\deg(u) = k, \deg(v) = l\}(n\deg(u))^{-1},$$

but by pulling the sum over vertex pairs out front then the sums over $k$ and $l$ have only one non-zero term. Hence $\mathbb{E}\ \deg(U) = \frac{1}{n}\sum_{(u,v)} \frac{\deg(v)}{\deg(u)}$. A similar approach gives $\mathbb{E}\ \deg(V) = \frac{1}{n}\sum_{(u,v)} 1$.

To finish the proof, use the Cauchy-Swartz inequality.

$$\begin{aligned}
\mathbb{E}\ \deg(V) &= \frac{1}{n}\sum_{(u,v)} \sqrt{\frac{\deg(u)}{\deg(v)}}\sqrt{\frac{\deg(v)}{\deg(u)}} \\
&\leq \sqrt{\sum_{(u,v)} \frac{\deg(u)}{\deg(v)}}\sqrt{\sum_{(u,v)} \frac{\deg(v)}{\deg(u)}} \\
&= \frac{1}{n}\sum_{(u,v)} \frac{\deg(u)}{\deg(v)} \\
&= \mathbb{E}\ \deg(U),
\end{aligned}$$

with the second equality possible thanks to the symmetry of summing over all ordered edges. □

### 2.1.1 Graph Statistics

Graphs are high-dimensional, highly sophisticated objects. This makes them very difficult to visualize and compare in a meaningful way. It is therefore helpful to consider *graph statistics* — functions of the form $\varphi : \mathcal{G}^* \to \mathbb{R}$. A collection of these functions can summarize the graph into a few characteristic numbers that highlight important structural features. Poor selection of graph statistics will cause us to miss interesting properties. Which graph statistics are appropriate for use is heavily dependent on context.

One of the most common and useful graph statistics is the *density* of a multigraph $g = (\mathcal{V}_g, \mathcal{E}_g)$, defined as

$$\mathrm{Den}(g) := \frac{2|\mathcal{E}_g|}{|\mathcal{V}_g|(|\mathcal{V}_g| - 1)}.$$

For a simple graph $\mathrm{Den}(g)$ tells us the proportion of vertex pairs that have an edge between them. Real-world networks often have low density, containing edges roughly proportional to the number of vertices.

A general approach to create an interesting graph statistic is to simply count the number of times a specific subgraph $h$ appears in the graph $g$. For example, if $h$ and $g$ are the graphs from Figure 2.1, we would be counting the number of 3-stars in $g$, denoted $3-\text{stars}(g)$. In this case $3-\text{stars}(g) = 10$.

We are often interested in the notion of average distance between any two randomly chosen vertices in a multigraph. The following is one such statistic.

**Definition 2.3.** We say that $D(g)$ is the *expected typical distance* (ETD) of $g$, defined by

$$D(g) := \frac{1}{|\mathcal{L}_g|} \sum_{\{u,v\} \in \mathcal{L}_g} d_g(u, v), \tag{2.3}$$

where $\mathcal{L}_g$ is the set of all vertex pairs $\{u, v\}$ where $\mathcal{C}_g(v) = \mathcal{C}_g(u)$. ♦

Often we will want to know the *expected typical distance from* $u$. That is, the expected distance from $u$ to any point uniformly chosen from its component. We denote this quantity as $D_g(u)$ and formulate it as

$$D_g(u) = \frac{1}{|\mathcal{C}_g(u)|} \sum_{v \in \mathcal{C}_g(u)} d_g(v, u). \tag{2.4}$$

For completeness we include the definition of typical distance commonly used in the literature. The *typical distance* of a multigraph, denoted $\text{TD}(g)$, is a probability distribution of the distance between two vertices selected at random from $\mathcal{V}$, given that the two vertices are connected. Note that this is not a graph statistic, which is why this thesis focusses on expected typical distance instead. This property has been investigated by researchers in many publications, including [7], [11] and [22]. Specifically,

$$\mathbb{P}(\text{TD}(g) = i) = \frac{1}{|\mathcal{L}_g|} \sum_{\{u,v\} \in \mathcal{L}_g} \mathbb{I}\{d(u, v) = i\}, \tag{2.5}$$

where $\mathcal{L}_g$ is the set of all vertex pairs $\{u, v\}$ with $\mathcal{C}_g(v) = \mathcal{C}_g(u)$. As one would expect, $D(g) =$

$\mathbb{E}(\mathrm{TD}(g))$, as

$$\begin{aligned}\mathbb{E}(\mathrm{TD}(g)) &= \sum_{i=0}^{\infty} i\mathbb{P}(\mathrm{TD}(g) = i) \\ &= \frac{1}{|\mathcal{L}_g|} \sum_{\{u,v\}\in\mathcal{L}_g} \sum_{i=0}^{\infty} i\mathbb{I}\{d(u,v) = i\} \\ &= \frac{1}{|\mathcal{L}_g|} \sum_{\{u,v\}\in\mathcal{L}_g} d(u,v) \\ &= D(g).\end{aligned}$$

## 2.2 Random Graphs

We begin our discussion of random graphs with a formal definition.

**Definition 2.4.** Consider the probability space $(\Omega, \mathcal{F}, \mathbb{P})$. We say that a measurable function $G : \Omega \to \mathcal{G}^*$ is a *random graph.* ♦

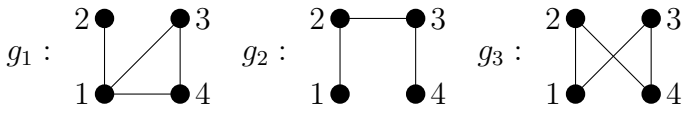
The following theorem ensures that every $\mathcal{A} \subset \mathcal{G}^*$ is measurable.

**Theorem 2.4.** *The set of all multigraphs $\mathcal{G}^*$ is countable.*

*Proof.* From Theorem 2.2 we have the invertible function Adj which maps any multigraph into its adjacency matrix. Let $\mathcal{M}_n$ be the set of all $n$-row, symmetric matrices with non-negative integer entries and even diagonal. As each element of $\mathcal{M}_n$ is a finite, ordered collection of integers, $\mathcal{M}_n$ is countable. Furthermore, $\mathcal{M}^* = \bigcup_n \mathcal{M}_n$ is countable because it is a countable union of countable sets. There exists a bijection between $\mathcal{M}^*$ and $\mathcal{G}^*$, so $\mathcal{G}^*$ must also be countable. □

Recall that graph statistics are functions $\varphi : \mathcal{G}^* \to \mathbb{R}$. By Theorem 2.4 we can infer that all graph statistics are measurable and therefore are a random variable when their argument is a random graph.

**Example 2.1.** Consider the following graphs.



For concreteness we will choose the probability space $(\Omega, 2^\Omega, \mathbb{P})$, where $\Omega = \{1, 2, 3\}$ and

$$\mathbb{P}(\{\omega\}) = \begin{cases} \frac{1}{2} & \text{if } \omega = 1 \\ \frac{1}{3} & \text{if } \omega = 2 \\ \frac{1}{6} & \text{if } \omega = 3. \end{cases}$$

Then, $G$, defined by $G(\omega) = g_\omega$ is a random graph, with $\mathbb{P}(G(\omega) = g_1) = \frac{1}{2}, \mathbb{P}(G(\omega) = g_2) = \frac{1}{3}$ and $\mathbb{P}(G(\omega) = g_3) = \frac{1}{6}$.

We can now look at the distributions of various graph statistics. For example, the distribution of the graph density is

$$\mathbb{P}(\text{Den}(G(\omega)) = x) = \begin{cases} \frac{2}{3} & \text{if } x = \frac{2}{3} \\ \frac{1}{3} & \text{if } x = \frac{1}{2} \\ 0 & \text{Otherwise.} \end{cases}$$

While there is no concept of an "expected graph", we will often look at the expected value of graph statistics. For example the expected distance between vertices 1 and 4 is $\mathbb{E}(d_G(1, 4)) = \frac{11}{6}$. ♦

The underlying probability space is generally not necessary to define. By Theorem 2.4 the set of multigraphs $\mathcal{G}^*$ is countable so every random graph $G$ can be converted into $g_X$ where $X$ is a positive integer-valued random variable. Finally we can apply Theorem 14.1 in [10] to our distribution for $X$, ensuring that any valid distribution over our multigraphs can be turned into a random graph. To simplify notation, we will write $G$ in place of $G(\omega)$ when the underlying probability space is implicitly defined.

While considering single random graphs is interesting in its own right, we ultimately want to allow the graphs to change over time. Some random graphs are constructed via a sequence of steps, each step containing its own random graph.

**Definition 2.5.** A collection of random graphs $\{G_t\}_{t \in \mathcal{T}}$ is called a *graph process* with *index set* $\mathcal{T}$. ♦

The usual choices for $\mathcal{T}$ will be $\{0, 1, 2, \ldots\}$ for discrete time and $[0, \infty)$ for continuous time processes. As we will be dealing with sequences of multigraphs $(g_t)_{t \geq 0}$ we will often write $\mathcal{C}_t(v)$, $D_t(v)$ and $d_t(v, u)$ in place of $\mathcal{C}_{g_t}(v)$, $D_{g_t}(v)$ and $d_{g_t}(v, u)$ when there are no ambiguities.

As mentioned in Section 2.1.1, many real-world networks have low density. A multigraph which exhibits this behaviour is called *sparse*. Formal definitions of sparsity vary, for example compare the definitions from [47] and [74]. As this thesis focusses on random graphs, we use the following definition.

**Definition 2.6.** A sequence of multigraphs $\{G_n\}_{n\in\{1,2,\dots\}}$, where each $G_n$ has $n$ vertices, is sparse if

$$\limsup_{n\to\infty} \mathbb{E}(\deg_{G_n}(V_i)) < \infty, \tag{2.6}$$

where $V_i$ is a vertex chosen uniformly at random from $G_n$. ♦

This definition is usually applied to random graph models which have the number of vertices $n$ as a parameter. When context is clear, referring to a single random graph $G$ as sparse implies that is generated by a model which satisfies Definition 2.6.

The notion of a *small world* is important to the study of complex networks. Informally, a network is a small world if the typical distance is bounded by the logarithm of the number of nodes in the network. The formal definition requires a sequence of random graphs $\{G_n\}_{n\in\{1,2,\dots\}}$, where each $G_n$ has $n$ vertices.

**Definition 2.7.** We say that a random graph model has the *small-world* property if

$$\lim_{n\to\infty} \mathbb{P}(\mathrm{TD}(G_n) \leq K \log n) = 1 \tag{2.7}$$

where $K$ is some constant and $G_n$ is a random graph from the model of interest with $n$ vertices. ♦

## 2.3 Erdős-Rényi Random Graphs

The first random graph model of interest to this thesis is the most influential and famous: the *Erdős–Rényi* (ER) random graph [24]. It is important to note that this graph was never intended to model real-world phenomena, rather it was developed as a tool to assist with graph theory proofs. The model has few direct applications — its popularity stems from its interesting theoretical properties and the relative ease through which said properties are derived.

There are two related ER models, with both distributed over the set of simple graphs $\mathcal{G}$. The first is denoted $\mathrm{ER}(n, m)$, where $m$ edges are distributed among $n$ vertices by selecting

vertex pairs uniformly without replacement [24]. Figure 2.3 has a few typical realizations for various $m$. The alternative formulation $\mathrm{ER}(n,p)$, first proposed by [28], adds an edge between each pair of vertices with probability $p$. Figure 2.4 contains a visualization for this formulation for various $p$. Both models avoid generating self-loops and parallel edges, so one may work in the space of simple graphs for convenience.

Let $\mathbb{P}_p(E)$ and $\mathbb{P}_m(E)$ be the probabilities of any event $E$ under $\mathrm{ER}(n,p)$ and $\mathrm{ER}(n,m)$ respectively,

$$\mathbb{P}_p(E) = \sum_{m=0}^{\bar{n}} \mathbb{P}_m(E) \binom{\bar{n}}{m} p^m (1-p)^{\bar{n}-m} \tag{2.8}$$

where $\bar{n} = \binom{n}{2}$ [74]. This means that properties of one variant can be translated to the other. From here on, we will use the second formulation $\mathrm{ER}(n,p)$ unless stated otherwise.

We now formally define the ER random graph using the notation from Sections 2.1 and 2.2.

**Definition 2.8.** A random graph $G = (\mathcal{V}_G, \mathcal{E}_G)$ is an Erdős–Rényi random graph $\mathrm{ER}(n,p)$ if $|\mathcal{V}_G| = n$ and the event $\{\{v,u\} \in \mathcal{E}_G\}$ is independent for each distinct pair $v,u \in \mathcal{V}_G$, with $\mathbb{P}(\{v,u\} \in \mathcal{E}_G) = p$. ♦

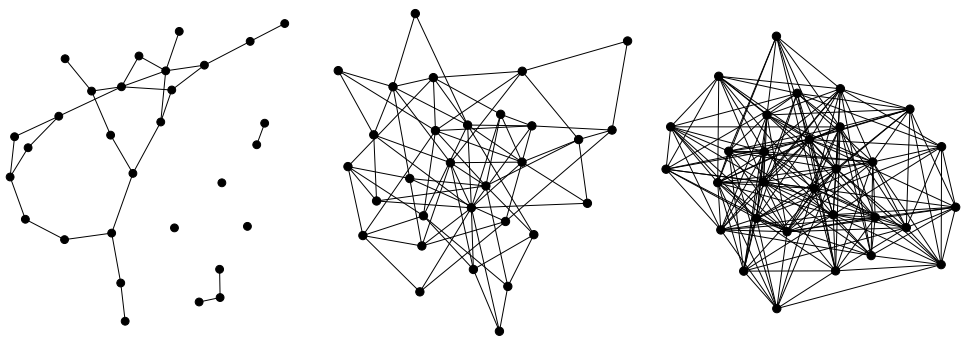


Figure 2.3: Realizations of the $\mathrm{ER}(30,m)$ model for $m = 30, 80$ and 200 respectively.

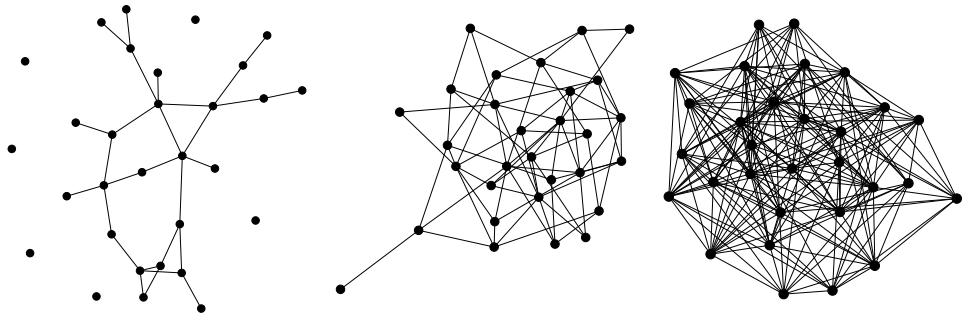


Figure 2.4: Realizations of the ER$(30, p)$ model for $p = 0.07, 0.16$ and $0.46$ respectively. These numbers were chosen to produce similar graphs to Figure 2.3.

As each edge is potentially present, the number of edges follows a Bin $\left(\binom{n}{2}, p\right)$ distribution, where Bin is the binomial distribution. Furthermore, each vertex has degree Bin$(n-1, p)$. Often the probability $p$ will be of the form $p = \lambda/n$ and the behaviour of the graph is studied as $n \to \infty$. As each vertex has a Bin$(n-1, \lambda/n)$ distribution, the probability distribution of the degree of a vertex approaches a Poi$(\lambda)$ distribution as $n \to \infty$. Here Poi is the Poisson distribution. Indeed, throughout this thesis when discussing the "asymptotic" behaviour of a random graph distribution, we will usually be talking about what happens as we increase the number of vertices. As the expected total number of edges grows at a linear rate, this satisfies the definition of sparsity specified in [74], meaning ER$(n, \lambda/n)$ is sparse for any $\lambda > 0$.

One property that has been extensively studied is the size of the largest component in a realization of an Erdős–Rényi random graph $G$. For $\lambda < 1$, $\lambda = 1$ and $\lambda > 1$ we refer to as the *sub-critical*, *critical* and *super-critical* regimes respectively. Here we will briefly discuss the super-critical regime. Refer to [54] or [74] for an in-depth look at all three regimes. Consider a Poisson branching process with parameter $\lambda$. As $\lambda > 1$, the branching process will have extinction probability $\eta_\lambda < 1$. We can couple this branching process to a breadth first search in a realization of ER$(n, \lambda/n)$, so that the total progeny of the branching process stochastically dominates the component size found through the breadth first search of our graph. This leads to the following theorem from [4]:

**Theorem 2.5.** *Fix $\lambda > 1$, set $\eta_\lambda$ to be the extinction probability of the Poisson branching process with parameter $\lambda$. For every $\nu \in (\frac{1}{2}, 1)$ there exists $\delta > 0$ such that*

$$\mathbb{P}\left(\left|\,|\mathcal{C}^*| - (1-\eta_\lambda)n\right| \geq n^\nu\right) = O(n^{-\delta}), \tag{2.9}$$

*where $\mathcal{C}^*$ is the size of the largest component in an* $\mathrm{ER}(n, \lambda/n)$ *model.*

What this means is that the largest component and the total progeny of the branching process are roughly the same size. Furthermore it implies that a randomly chosen vertex is likely to be in this *giant component* $\mathcal{C}^*$.

## 2.4 Preferential Attachment Models

The Erdős–Rényi random graph model is a poor fit for real-world networks as edges in real-world networks do not form independently of each other. For example, in a social network, you are far more likely to meet a friend-of-a-friend than a total stranger.

It was observed in [1] that the world wide web obeys a *power law*. That is, $\mathbb{P}(\deg(v) = k) \propto k^{-\tau}$ for arbitrary $v$ and some constant $\tau > 1$. The desire for graph distributions that can model this behaviour drove the development of the *Barabási–Albert* (BA) random graph. First proposed in [7] and formally defined in [12], the Barabási-Albert random graph uses preferential attachment to simulate the underlying processes found in many real-world networks. Since its inception, there have been numerous variants of the BA model. In this thesis, we focus on a generalization of the BA model called the *preferential attachment* (PA) model.

The PA model has three parameters: the number of vertices $n$, minimum degree $m$ and attachment bias $\alpha$. See Figure 2.6 for a demonstration of varying $m$ and Figure 2.5 for examples of varying $\alpha$. The Shorthand to denote the model with a particular set of parameters is $\mathrm{PA}(n, m, \alpha)$. The BA model is the special case where $\alpha = 0$.

The standard method of constructing a realization of a $\mathrm{PA}(n, m, \alpha)$ model first requires the specific case of $m = 1$ to be described. Realizations of $\mathrm{PA}(n, 1, \alpha)$ are created via a graph process $\{G_t\}_{t\in\mathcal{T}}$, where each $G_t$ is a multigraph and $\mathcal{T} = \{0, 1, \ldots, n\}$. The process starts with $G_0$ empty. To advance the process from any $G_t$, add a vertex labelled $t+1$ followed by an edge $\{t+1, V_{t+1}\}$, where $V_{t+1}$ is a vertex chosen randomly from the vertex set of $G_t$ according to the probability

$$\mathbb{P}(V_{t+1} = v \mid G_t = g_t) = \frac{\deg_{g_t}(v) + \alpha + \delta_{v,t+1}}{t(2+\alpha)+1} \quad v = 1, \ldots, t+1, \tag{2.10}$$

where $\delta_{i,j}$ is the Kronecker delta and $\alpha > -1$. This new graph with additional edge and vertex is called $G_{t+1}$. Once $t = n$, we deliver $G_n$ as a realization of a $\mathrm{PA}(n, 1, \alpha)$ random graph.

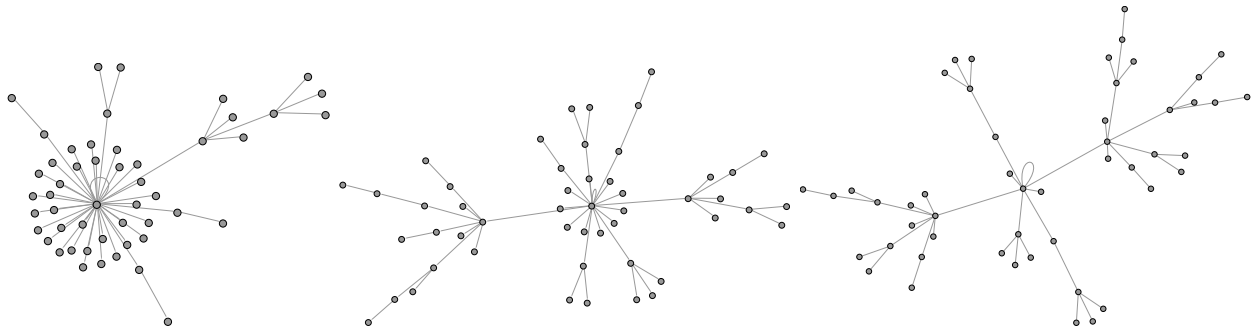


Figure 2.5: Realizations of $\mathrm{PA}(50, 1, \alpha)$ for $\alpha = 0.75, 1$ and 3 respectively.

To extend this to generating $\mathrm{PA}(n, m, \alpha)$ random graphs for $m \geq 2$, a common approach is to construct a $\mathrm{PA}(nm, 1, \alpha/m)$ graph and then merge vertices to reduce the total number of vertices to $n$. We will call this the *indirect construction*. Specifically, say we have generated $G$ with vertices $1, 2, \ldots nm$. Define $\mathcal{V}_i = \{m(i-1)+1, m(i-1)+2, \ldots, mi\}$ for $i \in \mathbb{Z}, 1 \leq i \leq n$. Construct a new graph $H$ with $n$ vertices. For every edge $\{u, v\}$ in $G$ with $u \in V_i$ and $v \in V_j$, add edge $\{i, j\}$ in $H$. Note that both $i = j$ and $u = v$ are permitted, creating self-loops in $H$.

The only parameter restriction for $\mathrm{PA}(n, m, \alpha)$ is $\alpha > -m$, inherited from the intermediary construction of $\mathrm{PA}(nm, 1, \alpha/m)$. The $\alpha = m$ case does not work for our construction, as at $t = 1$, Equation 2.10 evaluates to $\frac{0}{0}$. Some circumvent this issue by starting the graph with a single vertex with a loop. If this is done, the graph generated is trivial — all edges attach to vertex 1.

We will now cover some properties of the Barabási–Albert model (that is, $\mathrm{PA}(n, m, 0)$). Empirical studies indicate that the degree of a randomly chosen vertex follows a power law with $\tau = 3$ when $n$ is sufficiently large [7]. This result was proven to be true asymptotically, with a proof available in [23]. This power-law degree sequence means that we expect each BA graph to have a few nodes with a high concentration of edges.

In the general preferential attachment model, the $\alpha$ parameter alters the power-law exponent. Work in [12] has shown that the power-law exponent is $\tau = 3 + \alpha/m$ for $\mathrm{PA}(n, m, \alpha)$ making it a useful generalization of the Barabási–Albert model. The maximal degree of the PA model has also been investigated [53]. Let $M_n$ be the maximal degree of a PA random graph with $\alpha$ and $m$ fixed. Then there exists a random variable $X$ with $\mathbb{P}(X = 0) = 0$ such that $M_n n^{-1/(2+\alpha/m)} \to X$ as $n \to \infty$ almost surely. Compare this to the maximum of $n$ iid random variables distributed acording to our power-law $\tau$ — we find that $\tau$ This means that $M_n$ has the same order as the maximum of $n$ iid random variables distributed according to a power-law

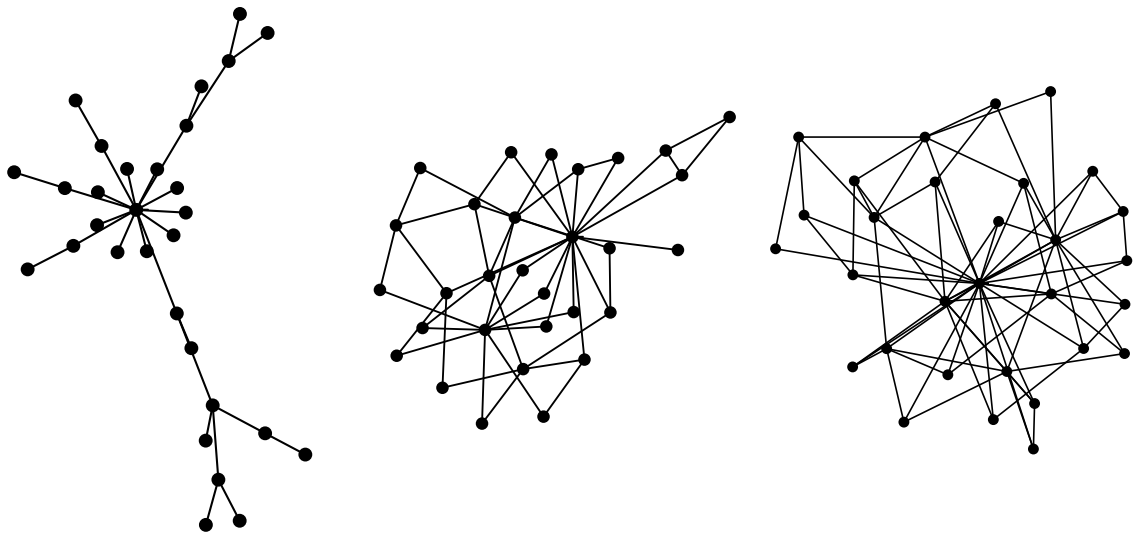


Figure 2.6: Realizations of PA$(30, m, 0)$ for $m = 1, 2$ and 3 respectively.

with exponent $\tau = 3 + \alpha/m$, as the expected maximum of the iid sequence is proportional to $n^{1/(2+\alpha/m)}$.

One key result from the study of PA models is that they exhibit the small-world property (see Definiton 2.7). Work in [11] used results from [57] to show that for PA$(n, 1, 0)$, the asymptotic typical distance bounded on both sides by an $O(\log n)$ function. For more general $m = 1$ models we refer the reader to the accessible proof of Theorem 7.1 of [74], which shows PA$(n, 1, \alpha)$ models satisfy

$$\frac{TD(G_n)}{\log(n)} \to \frac{2(1+\alpha)}{2+\alpha}, \tag{2.11}$$

where the convergence is in probability. Simple rearrangement will show that this satisfies Definition 2.7, thus an $O(\log n)$ function bounds typical distance with high probability. Further work has shown that for $m \geq 2$, the typical distance can be much smaller. For $\alpha < 0$, the typical distance converges in probability to an $O(\log \log n)$ function [21]. Of note is the $\alpha = 0, m \geq 2$ case, which corresponds to a power-law exponent $\tau = 3$. Here it can be proven that the typical distance converges to some $O(\frac{\log n}{\log \log n})$ function [11]. For completeness we note that when $m \geq 2$ and $\alpha \geq 0$, the typical distance is bounded by $O(\log n)$ as with the $m = 1$ case [22].

As the PA random graph exhibits a power-law degree sequence and is a small world, it has the base properties required for simulating and modelling real-world complex networks. For example it has been used to simulate human sexual relationships [41] and protein network evolution [58]. We note that the PA model uses a set of simple and sensible rules for simulating a network, which means it can be used to explain how the real-world networks developed.

As seen in the realizations of Figure 2.5, PA random graphs typically consist of a single large component. In the $m = 1$ case, a disconnectivity must be due to vertex selecting itself and forming a self-loop. As there is no way to bridge between two existing components, the random variable $C_n$ for the number of components in a PA$(n, 1, \alpha)$ is

$$C_n = X_1 + X_2 + \cdots + X_n, \tag{2.12}$$

where $X_i = 1$ with probability $\frac{1+\alpha}{i(2+\alpha)+1}$, and 0 otherwise. For large graphs $C_n/\log n$ converges in probability to $(1+\alpha)/(2+\alpha)$[74], implying that the number of components grows logarithmically in $n$. When $m \geq 2$, components may be joined as the construction of the graph proceeds. Furthermore the probability that a vertex forms the $m$ self-loops necessary to start a new component is more difficult as $m$ increases. This leads to $m \geq 2$ PA graphs to tend towards a single giant component. Proof of this asymptotic result is available in [74].

Creating PA$(n, m, \alpha)$ random graphs using the indirect construction presented earlier are unsuitable for the Monte Carlo methods applied later in Chapter 5. The intermediate graph used when constructing $m \geq 2$ PA graphs will not give correct results if we were to evaluate graph statistics of it. For that reason, we use the *direct construction* defined by Algorithm 2.1. Theorem 2.6 demonstrates that this construction is equivalent to the one presented in the literature.

**Theorem 2.6.** *The direct construction* $\{G_t\}_{t\in\{0,1,\ldots nm\}}$ *is equivalent to the indirect construction* $\{H_t\}_{t\in\{0,1,\ldots nm\}}$ *of* PA$(n, m, \alpha)$ *random graphs.*

*Proof.* Recall that the indirect construction first creates a PA$(nm, 1, \alpha/m)$ graph and then merges the vertex sets $\mathcal{V}_i = \{m(i-1)+1, m(i-1)+2, \ldots, mi\}$ for $i \in \mathbb{Z}, 1 \leq i \leq n$. Let $K_t$ be the $\lceil t/m \rceil$ vertex graph of $H_t$ after this vertex merging process.

If the probability of adding edge $\{v, u\}$ at step $t$ is the same for both $G_t$ and $K_t$ then both constructions are equivalent. Set $v = \lceil (t+1)/m \rceil$, the newest vertex to the graph at time $t+1$. The probability of adding edge $\{v, u\}$, with $u \neq v$, to $G_t$ is $\deg_t(u) - m + m(1+\alpha/m) = \deg_{G_t}(u)+\alpha$. The probability of edge $\{v, v\}$ being added to $G_t$ is $\deg_t(u)-y+(y+1)(1+\alpha/m) = \deg_t(u) + 1 + y\alpha/m$, where $y = t \mod m$, which is the number of edges added from $v$ in $G_t$.

Next we cover the probabilities for $K_t$. Adding the edge $\{t, z\}$ to $H_t$, with $z \in \mathcal{V}_u$ translates to the edge $\{v, u\}$ being added to $K_t$. The probability of this is determined by the probability that the vertex chosen lies in $z \in \mathcal{V}_u$. Thus the probability of adding edge $\{v, u\}$ with

$u \neq v$ is $\sum_{z \in \mathcal{V}_u} \left(\alpha/m + \deg_{H_t}(z)\right) = \alpha + \deg_{K_t}(z)$, where the equality is valid as the merging process preserves degree. Similarly, the probability of edge $\{v, v\}$ being added to $K_t$ is $\sum_{z \in \mathcal{V}_v} \left(1 + \deg_{H_t}(z) + \alpha/m\right) = \deg_{K_t}(z) + 1 + y\alpha/m$, where $y = t \mod m$.

Clearly if $K_t = G_t$ then the probability of adding $\{v, u\}$ is the same. We have $K_0 = G_0$, and given $K_t = G_t$ then $K_{t+1} = G_{t+1}$. Inductively this means $K_t = G_t$ for all $t \in \{0, 1, \ldots nm\}$ as required. □

**Algorithm 2.1:** Simulating a PA$(n, m, \alpha)$ random graph.

**Input:** Number of vertices $n$, edge parameter $m$, attachment weight $\alpha > -m$

**Output:** PA$(n, m, \alpha)$ graph $G_{nm}$

Initialize $G_0$ as an empty graph

Initialize attachment weight array $A$

$w \leftarrow 0$

$t \leftarrow 0$

**for** $t < nm$ **do**

- $v \leftarrow \lceil (t+1)/m \rceil$
- **if** $t \mod m \equiv 0$ **then**
  - Add vertex labelled $v$ to graph $G_t$
- $A[v] \leftarrow A[v] + 1 + \frac{\alpha}{m}$, $w \leftarrow w + 1 + \frac{\alpha}{m}$
- $U \leftarrow$ Random integer chosen from $\{1, 2, \ldots, v\}$ with probabilities $A/w$
- $A[U] \leftarrow A[U] + 1$, $w \leftarrow w + 1$
- Add edge $\{v, U\}$ to $G_t$ to form $G_{t+1}$
- $t \leftarrow t + 1$

**return** $G_{nm}$

## 2.5 Exponential Random Graph Models

The *exponential random graph* (ERG) model was briefly discussed Chapter 1; here we will give further details and insights for the model.

**Definition 2.9.** The ERG model encompasses a family of distributions whose probability mass functions $f$ satisfy

$$f(g) = \kappa \exp\left(\sum_{i=1}^{m} \theta_i T_i(g)\right), \tag{2.13}$$

where $g$ is a simple graph on $n$ vertices, $\kappa$ is the normalisation constant, $T_1, \ldots, T_m$ are graph statistics, and $\theta_1, \ldots, \theta_m$ are model parameters. ♦

The ERG model is predominantly used as a tool for detecting interesting behaviour when fit to a data set. For example in [32] it was used to detect transitive behaviour (that is, if triangles are more likely than 2-stars). The model also can be used to detect structure between labelled vertices. In [37] the model was fit to friendship data obtained from primary schools. As the student's race was included in the data, the researchers of [37] created a graph statistic that counts the number of homogenous relationships. Maximum likelihood estimation suggested that the parameter is positive, giving evidence that students prefer to make friends with people of similar ethnicity.

A large number of possible selections of $\boldsymbol{T}$ (including the Markov model, for example) exhibit *near-degenerate* behaviour. Near-degeneracy refers to the random graph being either full or empty with high probability, which renders the model useless. The example provided in [17] is an ERG model where the sufficient statistics are edge and triangle count, with parameters $\beta_1$ and $\beta_2$, respectively. It was found that models such as this are approximately $\mathrm{ER}(n, u^*)$ for $n$ large and $u^* = u^*(\beta_1, \beta_2)$ is a computable constant. Figure 3 of [17] contains a plot of $u^*(-5, \beta_2)$ for $\beta_2 \in (0, 2)$. The extreme values of $u^*$ imply that we will either obtain full or empty graphs.

Careful manipulation of parameters is necessary to make full use of the model and avoid degeneracy. A good example of this is the concept of *alternating k-stars* and *alternating k-triangles* introduced in [69] and further investigated in [31]. The concept is motivated by simple deduction: $k$-stars and $k$-triangles contain $k$ copies of a $(k-1)$-star or $(k-1)$-triangle respectively. Hence by giving 1-triangles and 3-stars positive weight, the higher order structures are weighted too heavily. This in turn causes such higher order structures to appear more often

than the real-world network we are attempting to model. Furthermore, this slows convergence of the chain to stationarity as the chain will become caught in local maxima which have high order triangles or stars. To remedy this, we equate the parameters $\nu_h = \sigma_k$ where $h$ is a $k$-star and set $\sigma_k = -\sigma_{k-1}/\lambda$ for $k = 3, 4, \ldots, n-1$, where $\lambda > 1$ is a constant and $\sigma_2$ is a free parameter. A similar recursive parameter system is also set up for $k-$triangles. Results from [31] indicate that this provides both faster mixing and more appropriate graphs.

# Chapter 3

# Monte Carlo Techniques

In Chapters 4, 5 and 6 we will confront estimation and simulation problems that deterministic numerical methods struggle to solve. As we are already working within the probabilistic framework of random graphs, it is sensible to apply Monte Carlo techniques to these problems in an attempt to find approximate solutions.

The Sections 3.1 and 3.2 of this chapter define and discuss the variance reduction techniques *splitting* and *importance sampling*. Variance reduction is essential for providing accurate estimates of *rare-event* probabilities. We say $A$ is a rare event if $\mathbb{P}(A)$ is very small, for example less than $10^{-5}$. For more detail on why variance reduction is necessary for rare-event estimators, we refer the reader to [45].

Section 3.3 gives a brief description of the *Metropolis–Hastings* algorithm and an important special case known as the *Gibbs sampler*. We explain the wide applications for these techniques and potential problems that may arise. We conclude with Section 3.4, which describes a fairly new method known as the *stratified splitting algorithm*. This technique is designed for estimation, but the temporary results created during the algorithm's run time can also be used for simulation from complicated distributions.

Note that all simulation results presented in this thesis were performed on a machine running Windows 10 64-bit, Intel Core i5 6500  3.20GHz processor with 8.00GB RAM. Multithreading and GPU acceleration were not used.

## 3.1 Splitting

Consider a Markov chain $\{\boldsymbol{X}_t, t = 0, 1, 2, \ldots\}$. We often want to know the probability that it enters some target set $\mathcal{X}_\gamma$ before hitting an undesirable set $\mathcal{X}_0$, with $\mathcal{X}_\gamma \cap \mathcal{X}_0 = \varnothing$. Choose sets $\mathcal{X}_1 \supset \mathcal{X}_2 \supset \cdots \supset \mathcal{X}_{\gamma-1} \supset \mathcal{X}_\gamma$ such that all chosen sets are disjoint from $\mathcal{X}_0$. Let $\tau_i$ be the first time that the process hits either set $\mathcal{X}_0$ or $\mathcal{X}_i$, and define $A_i$ as the event $\{\boldsymbol{X}_{\tau_i} \in \mathcal{X}_i\}$. Then

$$A_1 \subset A_2 \subset \cdots \subset A_\gamma$$

and

$$\ell := \mathbb{P}(A_\gamma) = \mathbb{P}(A_1)\mathbb{P}(A_2 \mid A_1) \cdots \mathbb{P}(A_\gamma \mid A_{\gamma-1}).$$

We can estimate $\mathbb{P}(A_i)$ by running $N$ chains $\{X_t^{(j)}\}$, and stopping when the outcome of $A_i$ is known (at time $\tau_{ij}$ for the $j$th chain). After running these chains we will have $\boldsymbol{X}_{\tau_{ij}}^{(j)}$ for $j = 1, \ldots, N$ and let $\mathcal{Y} = \{\boldsymbol{X}_{\tau_{ij}}^{(j)} \in \mathcal{X}_i\}$. Provided $\mathcal{Y}$ is not empty, we can then resample this set to produce a set $\mathcal{Z}$ which contains $N$ chains conditioned on the success of $A_i$. We can use this resampled set to estimate $\mathbb{P}(A_{i+1} \mid A_i)$ by running the chains in $\mathcal{Z}$ until the outcome of $A_{i+1}$ is known.

The above reasoning leads to the *splitting method*, an unbiased Monte Carlo technique [33]. In particular, the variant where we keep a fixed population of $N$ sample chains is called *fixed effort* splitting. If each level set $\mathcal{X}_i$ is chosen appropriately, such that $\mathbb{P}(A_i \mid A_{i-1})$ is not too small, crude Monte Carlo estimation for each of these individual probabilities will not have high variance. A complete pseudocode algorithm can be seen in Algorithm 3.1.

Splitting estimators can produce poor estimates under certain circumstances. If the set $\mathcal{Z}$ is empty at the $i$th level of splitting, $N_i = 0$ and $\hat{\ell}$ returns the inaccurate result of zero. Choosing $\mathcal{X}_i$ in such a way to avoid generating poor sample pools is a non-trivial, problem-specific task.

The splitting method has been applied to a variety of problems, ranging from applied sciences such as photon and electron radiography [60], [65]; to pure mathematics where splitting can be used to solve the SAT problem [15].

**Algorithm 3.1:** Splitting

**Input:** Fixed effort $N$, functions for determining if $\boldsymbol{X}_\tau \in \mathcal{X}_i$ for $i \in \{0, 1, \ldots, \gamma\}$

**Output:** Returns estimate $\hat{\ell}$ of $\mathbb{P}(A)$

Generate initial points $\boldsymbol{X}_0^{(1)}, \boldsymbol{X}_0^{(2)}, \ldots, \boldsymbol{X}_0^{(N)}$

$\mathcal{Z} \leftarrow \{\boldsymbol{X}_0^{(1)}, \boldsymbol{X}_0^{(2)}, \ldots, \boldsymbol{X}_0^{(N)}\}$

$\mathcal{Y} \leftarrow \emptyset$

**for** $i = 1, 2, \ldots, \gamma$ **do**

  **for** $j = 1, 2, \ldots, N$ **do**

    Run $\boldsymbol{X}_{\tau_{i-1}}^{(j)}$ until time $\tau_i$, when it enters either $\mathcal{X}_0$ or $\mathcal{X}_i$

    **if** $\boldsymbol{X}_{\tau_i}^{(j)} \in \mathcal{X}_i$ **then** $\mathcal{Y} \leftarrow \mathcal{Y} \cup \{\boldsymbol{X}_{\tau_i}^{(j)}\}$

  $\mathcal{Z} \leftarrow \emptyset$

  **for** $j = 1, 2, \ldots, N$ **do**

    Choose an $\boldsymbol{X}_{\tau_i}^{(j)}$ uniformly at random from $\mathcal{Y}$

    $\mathcal{Z} \leftarrow \mathcal{Z} \cup \{\boldsymbol{X}_{\tau_i}^{(j)}\}$

  $N_i \leftarrow |\mathcal{Y}|$

  $\mathcal{Y} \leftarrow \emptyset$

**return** $N^{-\gamma} \prod_{i=1}^{\gamma} N_i$

## 3.2 Importance Sampling

Consider the problem of finding $\ell = \mathbb{E}_f[h(\boldsymbol{X})]$ where $\boldsymbol{X}$ is a random variable with density $f$. For any alternative probability density function $k$ that satisfies $k(x) = 0 \iff h(x)f(x) = 0$, the following holds true:

$$\begin{aligned}
\mathbb{E}_f[h(\boldsymbol{X})] &= \int h(\boldsymbol{x}) f(\boldsymbol{x}) d\boldsymbol{x} \\
&= \int \frac{h(\boldsymbol{x}) f(\boldsymbol{x})}{k(\boldsymbol{x})} k(\boldsymbol{x}) d\boldsymbol{x} \\
&= \mathbb{E}_k \left[ h(\boldsymbol{X}) \frac{f(\boldsymbol{X})}{k(\boldsymbol{X})} \right].
\end{aligned}$$

This result implies that rather than sampling $\boldsymbol{X}$ from $f$, we can sample from $k$ and reweigh the outcome to get the same result. Intuitively, if $k$ assigns more probability mass to regions where $H(\boldsymbol{x})$ varies greatly, we should be able to reduce the variance for estimating $\ell$.

**Definition 3.1.** We say that $\hat{\ell}$ is an *importance sampling* estimator for $\mathbb{E}_f[h(\boldsymbol{X})]$ if

$$\hat{\ell} = \frac{1}{N}\sum_{i=1}^{N} h(\boldsymbol{X}_i)\frac{f(\boldsymbol{X}_i)}{k(\boldsymbol{X}_i)}, \tag{3.1}$$

where $\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_N \sim_{iid} k$ and $N$ is the number of samples taken from $k$. ♦

A mere glance at the Monte Carlo literature shows numerous applications of importance sampling, including estimating sums of heavy-tailed random variables [5], [6]; counting the number of self avoiding walks of a specific length [51] [81]; and efficient construction of bootstrap confidence intervals [40].

The main drawback of importance sampling is choosing an appropriate density $k$. A poor choice will result in an estimator with higher variance than the *crude Monte Carlo* (CMC) estimator, obtained by choosing $k = f$. Furthermore, generating samples from $k$ may be difficult. The additional time required for each sample may be better spent on generating more samples from $f$.

## 3.3 Markov Chain Monte Carlo

Some probability distributions are only known in terms of an unnormalized density or mass function. Using a technique developed by Metropolis [52] and Hastings [34], we are able to take approximate samples from such distributions by simulating a particular Markov chain. This technique is called *Markov chain Monte Carlo* (MCMC). We will assume the state space of our Markov chain, $\mathcal{S}$, is countable and all chains discussed have a transition function $P(\boldsymbol{x}, \boldsymbol{y})$, signifying the probability of transitioning from $x$ to $y$.

Recall that a *stationary distribution* $\pi$ of a Markov chain $\{\boldsymbol{X}_t\}_{t\geq 0}$ satisfies

$$\sum_{\boldsymbol{x}\in\mathcal{S}} \pi(\boldsymbol{x})P(\boldsymbol{x}, \boldsymbol{y}) = \pi(\boldsymbol{y}), \tag{3.2}$$

for all $\boldsymbol{y} \in \mathcal{S}$. Next, the *limiting distribution* $\pi^*$ of $\{\boldsymbol{X}_t\}$ is

$$\pi^*(\boldsymbol{y}) = \lim_{n\to\infty} \sum_{\boldsymbol{x}\in\mathcal{S}} \pi_0(\boldsymbol{x})P^n(\boldsymbol{x}, \boldsymbol{y}), \tag{3.3}$$

for all $\boldsymbol{y} \in \mathcal{S}$, where $\pi_0$ is an initial distribution for the chain. If there exists a time $\tau$ such that for and $t \geq \tau$, $P^t(\boldsymbol{x}, \boldsymbol{y}) > 0$ for all states $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{S}$ then we say $\{\boldsymbol{X}_t\}$ is *ergodic*. Ergodicity

implies that $\{\boldsymbol{X}_t\}$ has only one limiting distribution for all $\pi_0$, only one stationary distribution, and $\pi = \pi^*$. Therefore if it is possible to construct an ergodic Markov chain such that $\pi = f$, a realization $\boldsymbol{X}_t$ for $t$ large will be approximately distributed according to our target distribution $f$.

We first look at the *Metropolis–Hastings* implementation of MCMC, shown below in Algorithm 3.2. We will assume the distribution of interest is discrete — for details on a continuous implementation, see [26] or [45]. Let $\bar{f}$ be an unnormalized probability mass function for the target distribution and $P$ be the selected Markov transition function. We can choose any $P$ provided that the resulting chain is ergodic [48]. It is straightforward to prove that the limiting distribution of the chain specified in Algorithm 3.2 is $f$; see [48] for a detailed proof.

**Algorithm 3.2:** Metropolis–Hastings Algorithm

**Input:** Unnormalized density $\bar{f}$, proposal transition function $P$, iterations $T$

**Output:** $T$ approximate samples from $\bar{f}$

$\boldsymbol{X}_0 \leftarrow x$ for any $\boldsymbol{x}$ such that $f(\boldsymbol{x}) > 0$

**for** $t = 1, 2, \ldots, T-1$ **do**

- $\boldsymbol{Y} \sim P(\boldsymbol{X}_t, \boldsymbol{y})$
- $\alpha \leftarrow \min\left\{\frac{P(\boldsymbol{Y},\boldsymbol{X}_t)\bar{f}(\boldsymbol{Y})}{P(\boldsymbol{X}_t,\boldsymbol{Y})\bar{f}(\boldsymbol{X}_t)}, 1\right\}$
- $U \leftarrow$ Random real number uniformly chosen from $(0, 1)$
- **if** $U \leq \alpha$ **then** $\boldsymbol{X}_{t+1} \leftarrow \boldsymbol{Y}$
- **else** $\boldsymbol{X}_{t+1} \leftarrow \boldsymbol{X}_t$

**return** *Approximate samples* $\boldsymbol{X}_0, \boldsymbol{X}_1, \ldots, \boldsymbol{X}_{T-1}$

Of interest to this thesis is a special case of the Metropolis–Hastings algorithm known as the *random sweep Gibbs sampler*. For simplicity, we will refer to this sampler as *Gibbs* or the *Gibbs sampler*. It follows Algorithm 3.2 but the transition density $P$ is chosen in a specific way. Given that our sample-space is $n$-dimensional, the Gibbs sampler has us choose a random dimension $i$ uniformly followed by "updating" that dimension by sampling from the conditional density. Specifically, let $\boldsymbol{y}_{-j}$ be the vector formed from $\boldsymbol{y}$ by excluding the $j$th element. Then for a Gibbs sampler, the transition kernel $P$ is given by

$$P(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{n} \mathbb{I}\{\boldsymbol{x}_{-i} = \boldsymbol{y}_{-i}\} \frac{1}{n} \frac{\bar{f}(\boldsymbol{y} \mid \boldsymbol{x}_{-i} = \boldsymbol{y}_{-i})}{\sum_{\boldsymbol{w}} \bar{f}(\boldsymbol{w} \mid \boldsymbol{x}_{-i} = \boldsymbol{w}_{-i})}, \tag{3.4}$$

where the sum is taken over all $w$ such that $\boldsymbol{x}_{-i} = \boldsymbol{w}_{-i}$.

A key assumption of Algorithm 3.2 is that the Markov chain $\{\boldsymbol{X}_t\}$ quickly converges to the stationary distribution, ensuring our samples closely follow the target distribution. The *mixing time* $t_{\text{mix}}$ is when the chain is close to stationarity, with

$$t_{\text{mix}} = \min\left\{ t : \sum_{\boldsymbol{y}\in\mathcal{S}} \left| \left( \sum_{\boldsymbol{x}\in\mathcal{S}} \pi_0(\boldsymbol{x}) P^t(\boldsymbol{x},\boldsymbol{y}) \right) - \pi(\boldsymbol{y}) \right| \leq \frac{1}{2} \right\}, \tag{3.5}$$

where $P$ is the transition function for the chain, $\pi_0$ is the initial distribution of the Markov chain, $\pi$ is the stationary distribution. The choice of $\frac{1}{2}$ is mostly convention, see Section 4.5 of [48] for more details. Indeed, Theorem 4.9 of [48] tells us that after some time point the chain will converge at an exponential rate. However it is not difficult to construct an example where, as the problem size (e.g., dimension or states) increases, the mixing time increases exponentially. Attempting to reduce mixing time depends heavily on the target distribution $\bar{f}$, see [30] for examples.

## 3.4 Stratified Splitting Algorithm

The *stratified splitting algorithm* (SSA) combines ideas from splitting, importance sampling and MCMC to estimate $\mathbb{E}[h(\boldsymbol{X})]$ where $\boldsymbol{X}$ comes from a complicated density $f$ with unknown normalization constant [73]. The algorithm can also be repurposed into a method for approximate sampling from $f$. As with the previous section, we will assume that the sample space $\mathcal{X}$ is discrete; however, SSA can also be applied to general probability spaces. To use SSA we need another distribution $\varphi$ which has a known normalization constant and can be sampled from exactly. The details of how to implement SSA are found in Algorithm 3.3. We will now give some intuition on what the algorithm does and why it works.

The first loop of Algorithm 3.3 creates the set $\mathcal{X}_0$ by generating $N$ samples from $\varphi$. A simple choice of $\varphi$ for a discrete system is a uniform distribution across all elements of the sample space. For each iteration of the main loop, $\mathcal{X}_t$ is partitioned into an elite set $\mathcal{Y}_{t+1}$ and leftovers $\mathcal{Z}_t$. To choose the partition, all samples $\boldsymbol{X} \in \mathcal{X}_t$ are first sorted by their value of $\bar{f}(\boldsymbol{X})$, and value $\gamma_{t+1}$ is chosen such that the top $\rho N$ samples satisfy $\bar{f}(\boldsymbol{X}) \geq \gamma_{t+1}$. The adaptive level parameter $\rho \in (0,1)$ determines which proportion of samples should be accepted as elites for the next iteration. The partition is then set to $\mathcal{Z}_t = \left\{\boldsymbol{X} \in \mathcal{X}_t \,:\, \bar{f}(\boldsymbol{X}) < \gamma_{t+1}\right\}, \mathcal{Y}_{t+1} = \left\{\boldsymbol{X} \in \mathcal{X}_t \,:\, \bar{f}(\boldsymbol{X}) \geq \gamma_{t+1}\right\}$.

Once the partition is determined, the samples in $\mathcal{Z}_t$ are weighed and stored for the final

estimate calculation. The samples in $\mathcal{Y}_{t+1}$ need to be replenished so that there are $N$ samples all with weight greater than $\gamma_{t+1}$. To achieve this the following procedure is repeated $N$ times: chosen uniformly at random a sample from $\mathcal{Y}_{t+1}$ and move it using the Resample-Move algorithm from [29]. The $N$ moved samples are added to the set $\mathcal{X}_{t+1}$. The move is made according to the distribution $\varphi_t(\boldsymbol{x}) = h(\boldsymbol{x})\mathbb{I}\{\bar{f}(\boldsymbol{x}) \geq \gamma_t\}$.

If $\varphi$ is a uniform distribution across $\mathcal{X}$ then the local maxima and minima present in $\bar{f}$ will not slow down the mixing of the Markov chain, which overcomes a large drawback of basic Gibbs sampler approaches.

To sample from $\bar{f}$ using SSA we use the weighted samples $\{(\boldsymbol{Z}_m, w_m)\}$ defined on Line 12 of Algorithm 3.3. If a random variable $\boldsymbol{X}$ has its distribution defined by

$$\mathbb{P}(\boldsymbol{X} = \boldsymbol{Z}_m) = \frac{w_m}{\sum_{i=0} w_i}, \text{ for } i = 0, 1, \ldots, \tag{3.6}$$

then as the number of samples in our SSA approaches infinity, the distribution of $X$ converges to our target $f$ [50]. To obtain a realization of $X$, one can build a binary tree similar to the one specified in Algorithm 6.1.

SSA can fail under certain circumstances. If the dimensionality of the problem is too high, it will take many iterations before the samples are obtained from high density regions of $\bar{f}$. When this problem occurs, the algorithm will be too slow for practical use. This issue can be mitigated by carefully choosing an $\varphi$ which assigns higher probability to high density regions of $\bar{f}$.

**Algorithm 3.3:** SSA Sampler

**Input:** Unnormalized density $\bar{f}(\boldsymbol{x})$, adaptive level parameter $\rho$, function of interest $h(\cdot)$, and sample size $N$.

**Output:** An estimate of $\mathbb{E}_f[h(\boldsymbol{X})]$

Set $\widehat{R}_0 \leftarrow 1$, $m = 0$, and $\mathcal{X}_0 \leftarrow \emptyset$

Create empty vectors $\boldsymbol{Z}$ and $w$

**for** $i = 1$ **to** $N$ **do**

  Generate $\boldsymbol{X}$ from density $\varphi$

  Add $\boldsymbol{X}$ to $\mathcal{X}_0$

**for** $t = 0$ **to** $n - 1$ **do**

  Set $\gamma_{t+1} \leftarrow \bar{f}(\boldsymbol{X})_{(\lfloor N\rho \rfloor)}$ (i.e. the $\lfloor N\rho \rfloor$th order statistic )

  Set $\mathcal{Z}_t \leftarrow \left\{\boldsymbol{X} \in \mathcal{X}_t \ : \ \bar{f}(\boldsymbol{X}) < \gamma_{t+1}\right\}$

  Set $\mathcal{Y}_{t+1} \leftarrow \left\{\boldsymbol{X} \in \mathcal{X}_t \ : \ \bar{f}(\boldsymbol{X}) \geq \gamma_{t+1}\right\}$

  $\widehat{R}_{t+1} \leftarrow \frac{|\mathcal{Y}_{t+1}|}{N}$

  $\widehat{P}_t \leftarrow \left(1 - \widehat{R}_{t+1}\right) \prod_{j=0}^{t} \widehat{R}_j$

  **for** $k = 1$ **to** $|\mathcal{Z}_t|$ **do**

    $m \leftarrow m + 1$

    $\boldsymbol{Z}_m \leftarrow \mathcal{Z}_t^{(k)}, w_m \leftarrow \frac{\widehat{P}_t \bar{f}(\boldsymbol{Z}_m)}{|\mathcal{Z}_t|}$

  $\mathcal{X}_{t+1} \leftarrow \emptyset$

  **for** $i = 1$ **to** $N$ **do**

    Select $\boldsymbol{Y}_i$ uniformly from $\mathcal{Y}_{t+1}$

    Use MCMC starting at $\boldsymbol{Y}_i$ to generate $\boldsymbol{X}_i$ from $\varphi(\boldsymbol{x})\mathbb{I}\{\bar{f}(\boldsymbol{x}) \geq \gamma_{t+1}\}$

    Add $\boldsymbol{X}_i$ to $\mathcal{X}_{t+1}$

**return** $\sum_{k=1}^{m} h(\boldsymbol{Z}_k) w_k / \sum_{k=1}^{m} w_k$ as an estimator of $\mathbb{E}_f[h(\boldsymbol{X})]$.

# Chapter 4

# Generalized Dynamic Networks

In the year 2000, the ILOVEYOU virus spread virulently through the world, infecting over fifty-million computers and causing billions of dollars in damage [46]. Once a computer was infected it would send emails with an infectious attachment to everyone in the locally stored contact list. The international connectivity of the internet and poor malware defences at the time allowed the virus to propagate at an alarming rate. In order to determine how to minimize the damage of outbreaks such as this, we need a model to test potential strategies. One way to model this virtual epidemic is with a *contact process*.

A contact process $\{\boldsymbol{X}_t\}_{t\geq 0}$ is a stochastic process which evolves in a manner dependent on the topology of an underlying graph $g$. To model the computer virus example above, $g$ would represent an email network comprised of computers and $\boldsymbol{X}_t = (X_{1,t}, \ldots, X_{n,t})$ would represent the infection status of each computer $v \in \{1, \ldots, n\}$ at time $t$. A computer could contract the infection only if one of its neighbours in $g$ is infected.

Here we go further and model the contact process atop a random graph $G$. Many papers on this topic (for examples see [56] and [16]) model the network by running the contact process on a static realization of a random graph. However this approach ignores the effect that a changing graph can have on the propagation of the contact process. Examples of the changing graph architecture affecting the behaviour of the contact process were explored in [42], [35] and [38].

In [42] it was found that a changing graph decelerates the spread of an epidemic as a result of vertices exhibiting fluctuating periods of high and low connectivity. By comparing real-world

epidemic data to both static and dynamic models, they demonstrated that the dynamic models could better approximate the real-world process. This shows that modelling both the graph process and contact process in tandem is a worthwhile area of study.

This is what motivated the development of the *generalized dynamic network* (GDN) — a Markovian, continuous-time weighted graph process. Unlike other graph models where the graph process is studied as its order becomes very large (see [78], [7] or survey paper [54] for examples), the focus of GDN is the long-run behaviour where the expected number of vertices is bounded. Most models considered will be ergodic to ensure that the limiting distribution is unique (see Section 3.3 for details on ergodicity).

As with most random graph models, the true insight comes from computing various graph statistics of our process (Refer to Section 2.1.1). For example, if we were simulating the spread of a computer virus, a simple statistic would be the number of infected computers at time $t$.

This chapter starts by formally defining the GDN and providing example models. This is followed by discussion of the C++ package designed for rapid development and simulation of GDN models. The package is demonstrated by constructing three original models, all of which take advantage of the flexibility that GDN accommodates.

Section 4.3 explores fitting GDN models to observed data. While standard renewal theory techniques prove useful, they depend upon unrealistic data quality. An alternative scheme involving the splitting method is proposed and demonstrated.

In this chapter we are only concerned with **simple graphs**. No loops or parallel edges are permitted.

## 4.1 Definition and Examples

We first cement our definition of a *weighted graph*.

**Definition 4.1.** A *weighted graph* $g$ is a triple $(\mathcal{V}, \mathcal{E}, W)$ where $(\mathcal{V}, \mathcal{E})$ is a graph and $W \in \mathbb{R}^{|\mathcal{V}| \times m}$ is the *weight* matrix, where $m$ is the number of weights. Entry $W_{ij}$ corresponds to the $j$th weight of vertex $v_i \in \mathcal{V}$. ♦

Note that the vertices of the graph are being weighted, not the edges. Unless stated otherwise, $\mathcal{V} = \{1, 2, \ldots, n\}$, where the number of vertices $n$ will either be stated or be clear from the context. For the remainder of this chapter, the set $\mathcal{G}$ will represent the set of all weighted, simple graphs.

**Definition 4.2.** We say that a continuous time process $\{G_t\}_{t\geq 0}$ is a *Generalized Dynamic Network* (GDN) if the process $\{G_t\}_{t\geq 0}$ Markovian with state space over some set of weighted graphs. ⧫

A GDN is a continuous-time Markov Chain, so to specify how the process behaves it is necessary to identify the transition rates between states. For weighted graphs $g$ and $h$, let $q(g, h)$ be the rate at which $g$ transitions to $h$. While graph transitions of arbitrary complexity are possible, most sensible models will be constrained to simple transitions, such as adding or removing a single edge or vertex.

**Example 4.1.** Consider the following rudimentary epidemic model, where the vertices represent people and an edge $\{u, v\}$ implies that the people represented by $u$ and $v$ share close contact on a regular basis. The model will have a single weight per vertex, which takes values in $\{0, 1\}$. Let $W_{v1} = 1$ if person $v$ is infected by the virus of interest and 0 if $v$ is susceptible.

A susceptible person is added to the system at constant rate $\lambda$ and each person leaves the system at constant rate $\mu$. People form edges at rate $\alpha$ and existing edges are destroyed at constant rate $\beta$. Infected people recover at rate $\gamma$. The most interesting rate is infection, where each infected person transmits the infection to all susceptible neighbours at rate $\nu$. To ensure the process is ergodic, we include a background infection rate of $\eta$. Put succinctly, we have the following rates:

- $q(g, g + v) = \lambda$ for $v = |\mathcal{V}_g| + 1$.
- $q(g, g - v) = \mu$ if $v \in \mathcal{V}_g$.
- $q(g, g + \{u, v\}) = \alpha$ for all vertex pairs $\{u, v\} \notin \mathcal{E}_g$.
- $q(g, g - \{u, v\}) = \beta$ for all $\{u, v\} \in \mathcal{E}_g$.
- $q(g, (\mathcal{V}_g, \mathcal{E}_g, W^{(g)} + E_v)) = \eta + \nu \sum_{u\in\mathcal{N}_g(v)} W^{(g)}_{u1}$ for all susceptible $v \in \mathcal{V}$.
- $q(g, (\mathcal{V}_g, \mathcal{E}_g, W^{(g)} - E_v)) = \gamma$ for all infected $v \in \mathcal{V}$.

The variables $\lambda, \mu, \alpha, \beta, \nu, \eta, \gamma > 0$ are model parameters; $E_v$ is the $n \times 1$ matrix with a one in the $v$th row and zeroes elsewhere; and $\mathcal{N}_g(v)$ is the set of vertices adjacent to $v$. We assume

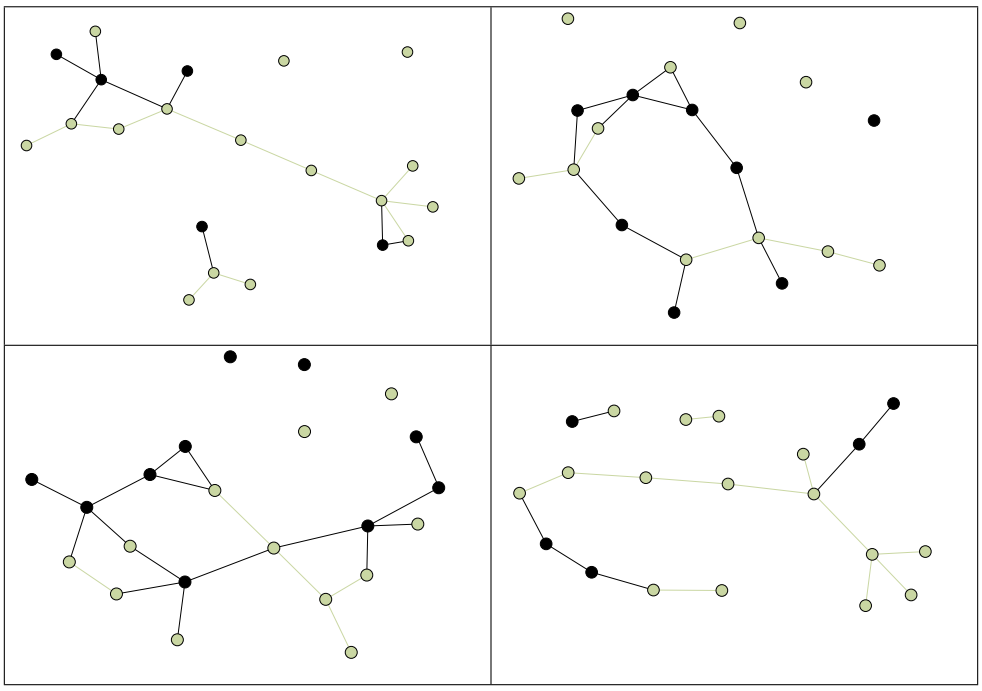


Figure 4.1: Time slices of the GDN defined in Example 4.1, taken at $t = 25, 50, 75$ and 100. The black vertices are infected (i.e. $\boldsymbol{w}_v = 1$), others are susceptible. The parameters used for this process were $(\lambda, \mu, \alpha, \beta, \nu, \eta, \gamma) = (0.02, 0.001, 0.01, 0.1, 0.1, 0.005, 0.05)$, selected to produce clear visualizations demonstrating each aspect of the model.

that vertices enter the system without infection. In Figure 4.1 we see some time-slices with typical behaviour for this process. An animated visualization of this model can be found at the URL https://youtu.be/NlutwnlU-qw. ♦

A major focus of the GDN model is the limiting distribution. To sample from the limiting distribution, the standard approach is to run the chain for a burn-in period and then take intermittent samples [45]. To determine the burn-in time, we can apply the diagnostic given in [27] to various graph statistics (e.g., edge density and number of vertices) and assess if the chain is sufficiently mixed. The samples that are obtained can be used to construct a crude Monte Carlo estimator for various graph statistics, for example number of edges, connectivity, maximum degree, and expected typical distance (refer to Section 2.1.1 for details).

Figure 4.2 demonstrates the convergence to stationarity for the proportion of infected vertices over time using the rates specified in Example 4.1. Here it is important to stress that the stochastic process $\{s(G_t)\}_{t\geq 0}$ induced by a graph statistic $s(\cdot)$ is not necessarily Markovian. This does not interfere with our objective, as $\{s(G_t)\}_{t\geq 0}$ will still converge to a limiting distribution provided that $s$ is well behaved.

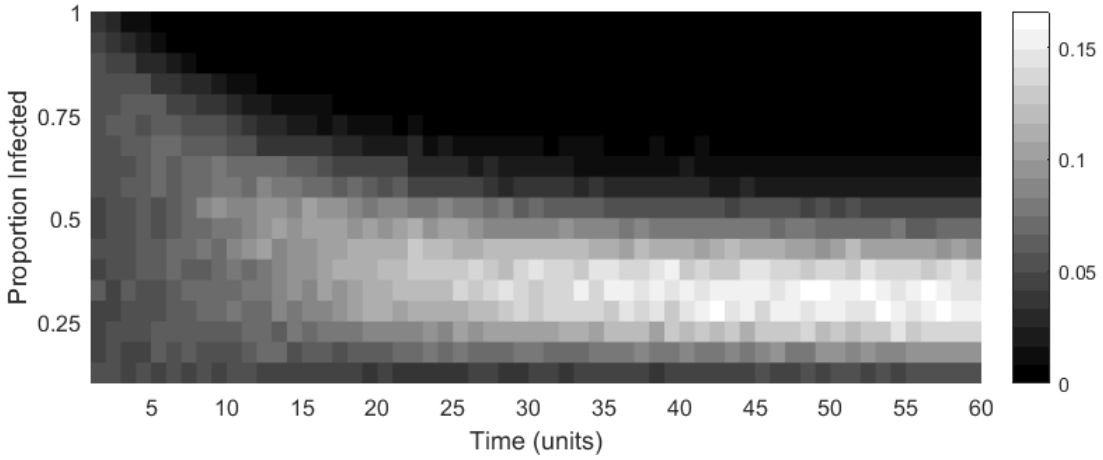



Figure 4.2: Empirical demonstration of the proportion of infected individuals in Example 4.1 converging to its stationary distribution. Lighter boxes have higher probability. Result was obtained by running 5000 chains with the parameters specified in Figure 4.1, finding the infected proportion at each time step and binning the output.

The next example demonstrates how a GDN can have edge transition rates affected by the underlying vertex weights.

**Example 4.2.** In this section we demonstrate the software and flexibility of the GDN network, specifically how it is possible for the weights of a vertex to influence its connections. Here we propose the *opinion network*, which draws inspiration from the voter model (see [49]) to incorporate personal convictions into a social network setting.

Consider a group of $n$ people who socialize, with friends-of-friends forming connections at a higher rate than complete strangers. Each person $v$ has their own set of opinions, beliefs and principles, represented by the $v$th row of weight matrix $W$. If $W_{vi} = W_{ui}$ then persons $v$ and $u$ share the opinion assigned to the $i$th column. The total number of opinions is known and fixed to $m$.

We now describe the network rates. If $u$ and $v$ share a common neighbour, they form an edge at rate $\alpha$, else they form an edge at rate $\gamma$. When there is an edge between $u$ and $v$, it is removed at rate

$$a + b\,\mathrm{Logit}\left(\sum_{i=1}^{m}(W_{vi} - W_{ui}) - \frac{m}{2}\right), \tag{4.1}$$

where $a, b > 0$ are parameters, Logit is the standard logistic function. This means that people with similar opinions lose their connection at rate approximately $a$ and people who are completely opposed to one-another lose their connection at a faster rate of approximately $a + b$. Next, each person changes opinion spontaneously at rate $\eta$ — the opinion changed is chosen

uniformly from the $m$ total opinions. Finally, one of the key parts of the model is that each $v$ will adopt an opinion of a neighbour at rate $\nu$. The precise mechanism is that an opinion $i$ of the $m$ total will be selected uniformly at random, and a neighbour $u$ will be selected uniformly at random as well.

In summary we have:

- $q(g, g + \{u, v\}) = \gamma$ for all vertex pairs $\{u, v\} \notin \mathcal{E}_g$ and $u, v$ share no neighbours.
- $q(g, g + \{u, v\}) = \alpha$ for all vertex pairs $\{u, v\} \notin \mathcal{E}_g$ and $|N_g(v) \cap \mathcal{N}_g(u)| > 0$ .
- $q(g, g - \{u, v\}) = a + b \operatorname{Logit}\left(\sum_{i=1}^{m} (W_{vi} - W_{ui}) - \frac{m}{2}\right)$, for all $\{u, v\} \in \mathcal{E}_g$.
- $q(g, (\mathcal{V}, \mathcal{E}, W^{(g)} \oplus E_{vi})) = \eta/m$ for $i \in \{1, 2, \ldots, m\}$.
- $q(g, (\mathcal{V}, \mathcal{E}, W^*[v, u, i])) = \nu/(m \deg_g(v))$, for all $v$, $u$ and $i$.

where $\oplus$ denote elementwise addition modulo 2; $W^*[v, u, i]$ is the matrix $W^{(g)}$ with the entry at the $v$th row and $i$th column set to $W^{(g)}_{ui}$; and $(E_{ij})$ is the matrix with a one at $i, j$ and zeroes everywhere else.

Our experiments were on a model with two opinions. We motivate this by appealing to the *political compass* [19], which is a two-dimensional representation of a persons political views. Note that we are simplifying the compass greatly by replacing its continuous scale with a discrete 0 or 1 for each compass direction.

As expected, for small groups the diversity in opinions is diminished over time as a majority forms and influences everyone else. Figure 4.3 shows the number of people holding the dominant opinion averaged over many realizations of the process. Interestingly, it appears that we do not get two competing components, just a single majority which absorbs most of the graph over time. The visualization video available at https://youtu.be/dAk4ADPfKew (alternatively see Figure 4.4) provides insight into why this behaviour occurs. A large number of people holding a single opinion form a densely connected group, reinforced by the higher connection rate for friends-of-friends. This cluster then assimilates others into its group — the large group is far more likely to convert a single person than vice versa. ♦

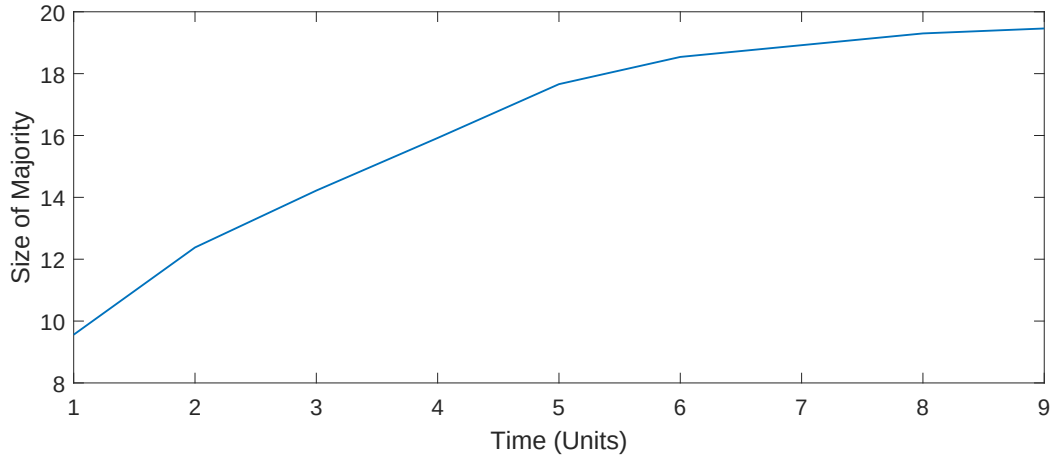



Figure 4.3: Average number of people who hold the most common opinion over 50 independent trials. The parameters are $n = 20$, $\gamma = 0.4$, $\alpha = 1$, $a = 0.4, b = 40$, $\nu = 6, \eta = 0.02$. These parameters were chosen to give short simulation times and a simple visualization, for debug purposes.

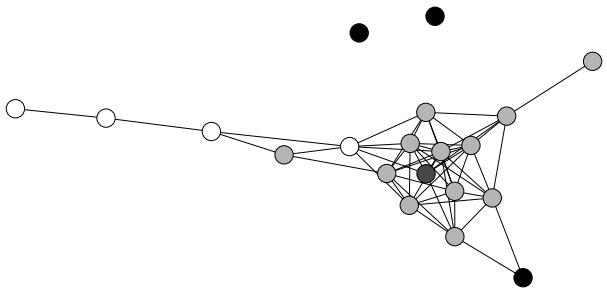


Figure 4.4: Realization of an opinion process after $t = 100$ time units. Different colours denote different opinions— light grey is $(0, 0)$, dark grey is $(1, 1)$, black is $(1, 0)$ and white is $(0, 1)$. The parameters are the same as those in Figure 4.3.

As we intend to use simulation studies to approximate the limiting distribution, we need to consider practical limitations imposed on our models. A major issue is if the expected number of vertices is large at times of interest then the process may require too much time or memory to simulate. There is no technique to mitigate this problem, however some simple calculations can be used to detect possible problems before starting a simulation.

Recall that a birth–death process is a continuous-time Markov chain on state space $\{0, 1, 2, \ldots\}$, where state transitions are only permitted between consecutive states. Consider a birth–death process with birth rate

$$\lambda_i = \max_{g \in \mathcal{G}_i} \left\{ \sum_{h \in \mathcal{G}_{i+1}} q(g, h) \right\}, \tag{4.2}$$

and death rate

$$\mu_i = \min_{g \in \mathcal{G}_i} \left\{ \sum_{h \in \mathcal{G}_{i-1}} q(g, h) \right\}, \tag{4.3}$$

for $i \in \{0, 1, \ldots\}$ and $\mathcal{G}_j$ representing the set of all weighted graphs on $j$ vertices. The following theorem allows us to bound the expected number of vertices in $G_t$, but first recall the definition of *stochastic domination*.

**Definition 4.3.** For random variables $X$ and $Y$, we say that $Y$ has *stochastic dominance* over $X$ if $\mathbb{P}(X \leq x) > \mathbb{P}(Y \leq x)$ for all $x \in \mathbb{R}$. We denote this relationship by $X \preceq Y$. ♦

**Theorem 4.1.** *Let $\{G_t\}_{t \geq 0}$ be an ergodic GDN with rates defined by $q(\cdot, \cdot)$ with $q(g, h) = 0$ if $|\mathcal{V}_g - \mathcal{V}_h| > 1$. If $\{B_t\}_{t \geq 0}$ is the birth–death process with birth rates $\{\lambda_i\}$ and death rates $\{\mu_i\}$ defined above, and $|\mathcal{V}_{G_0}| = B_0$, then*

$$|\mathcal{V}_{G_t}| \preceq B_t \,. \tag{4.4}$$

*Proof.* We will prove Equation 4.4 via a coupling argument. Consider the Markov process $\{(G_t, B_t)\}_{t \geq 0}$ with $G_t \in \mathcal{G}$, $B_t \in \{0, 1, \ldots\}$ and transition rate function $r$.

The goal is to build the transition rate function $r$ so that the marginal process $\{G_t\}_{t \geq 0}$ is a GDN with rate function $q$ and $\{B_t\}$ is a birth–death process with birth rates $\{\lambda_i\}$ and death rates $\{\mu_i\}$. First, for $g, h \in \mathcal{G}$ and $i \in \{0, 1, \ldots\}$ set $r\left((g, i), (h, i)\right) = q(g, h)$ if $|\mathcal{V}_g| = |\mathcal{V}_h|$.

Next, if $|\mathcal{V}_g| = i$ and $|\mathcal{V}_h| = i+1$ then $r((g, i), (h, i+1)) = q(g, h)$. To ensure the marginal process $\{B_t\}_{t \geq 0}$ has the required birth rates $\{\lambda_i\}$, we must set $r((g, i), (g, i+1)) = \lambda_i - \sum_h q(g, h)$, where the sum is taken over all $h$ which satisfies $|\mathcal{V}_h| = i + 1$. The rate is always non-negative as an immediate consequence of Equation 4.2.

Now take $|\mathcal{V}_g| = i$ and $|\mathcal{V}_h| = i-1$. Set $r((g,i),(h,i-1)) = c_{gi}q(g,h)$ and $r((g,i),(h,i)) = (1-c_{gi})q(g,h)$ for some $c_{gi} \in [0,1]$. Clearly this ensures that the rate from $g$ to $h$ in the marginal process $\{G_t\}_{t\geq 0}$ is $q(g,h)$ — we only need to select $c_{gi}$ in such a way that $\{B_t\}$ transitions from $i$ to $i-1$ at rate $\mu_i$. If we set

$$c_{gi} = \frac{\mu_i}{\sum_h q(g,h)},$$

then $\sum_h r((g,i),(h,i-1)) = c_{gi}\sum_h q(g,h) = \mu_i$ as required.

This ensures that if the graph process adds a vertex, the birth–death process does too. Similarly, if the birth-death process loses a vertex, the graph process is guaranteed to lose one. As we assume $B_0 = |V_{\mathcal{G}_0}|$, this implies $\mathbb{P}(|\mathcal{V}_{G_t}| \leq B_t) = 1$, and so $|\mathcal{V}_{G_t}| \preceq B_t$. □

**Corollary 4.2.** *Let $G_t$ be an ergodic GDN with rates defined by $q(\cdot,\cdot)$. We let $\pi$ be the stationary distribution of the vertex counting process $\{|\mathcal{V}_{G_t}|\}_{t\geq 0}$. If $\{B_t\}_{t\geq 0}$ is the birth–death process (with stationary distribution $\varphi$) defined above, $\alpha = \sup_i \lambda_i/\mu_i < 1$ and $q(g,h) = 0$ if $|\mathcal{V}_g - \mathcal{V}_h| > 1$, then*

$$\mathbb{E}_\pi|\mathcal{V}_{G_t}| \leq \mathbb{E}_\varphi B_t, \tag{4.5}$$

*where $\mathbb{E}_f X_t$ means the expectation of $X_t$ given that the process had initial distribution $f$.*

*Proof.* First recall $X \preceq Y$ implies $\mathbb{E}\,X \leq \mathbb{E}\,Y$, a well known result. By Theorem 4.1 we get $\mathbb{E}\left[|\mathcal{V}_{G_t}| \mid |\mathcal{V}_{G_0}| = x\right] \leq \mathbb{E}\left[B_t \mid B_0 = x\right]$ for all $x \in \{0,1,2,\ldots\}$. As all values are non-negative, the following holds:

$$\sum_x \pi(x)\,\mathbb{E}\left[|\mathcal{V}_{G_t}| \mid |\mathcal{V}_{G_0}| = x\right] \leq \sum_x \pi(x)\,\mathbb{E}\left[B_t \mid B_0 = x\right]$$

$$\mathbb{E}_\pi\,\mathbb{E}\left[|\mathcal{V}_{G_t}| \mid |\mathcal{V}_{G_0}| = X\right] \leq \mathbb{E}_\pi\,\mathbb{E}\left[B_t \mid B_0 = X\right]$$

$$\mathbb{E}_\pi|\mathcal{V}_{G_t}| \leq \mathbb{E}_\pi B_t.$$

As the left-hand-side is invariant under time, we can take the limits of both sides to obtain $\mathbb{E}_\pi|\mathcal{V}_{G_t}| \leq \lim_{t\to\infty}\mathbb{E}_\pi B_t$. If the $\{B_t\}$ are uniformly integrable, swapping limit and expectation of the RHS is permitted, giving $\mathbb{E}_\pi|\mathcal{V}_{G_t}| \leq \mathbb{E}_\pi B_\infty = \mathbb{E}_\varphi B_t$ as required.

For completeness, we will show that $\{B_t\}$ is a uniformly integrable sequence of random variables (See [63] for more on uniform integrability). Note that $B_0 \sim \pi$ and $B_\infty \sim \varphi$, thus by Theorem 4.1, $\mathbb{P}(B_t > i) \leq \mathbb{P}(B_\infty > i)$ for all $t \geq 0$. Recall that $B_\infty$ has distribution $\varphi$, which is the stationary distribution of a birth–death process, hence $\varphi_i = \varphi_0\prod_{j=0}^{i-1}\frac{\lambda_i}{\mu_i} \leq C\alpha^i$, where $C$ is

some positive constant. Thus the tail distributions $\mathbb{P}(B_t > i)$ are bounded by some geometric random variable, meaning $\{B_t\}$ is uniformly integrable. □

For Example 4.1 we have $\lambda_i = \lambda$ and $\mu_i = i\mu$, so the expected number of vertices in stationarity is $\lambda/\mu$. Thus if we design a process with $\lambda/\mu$ too high (say, over 500) then our simulation will take too long to execute.

## 4.2 Simulation Software

In this section we give an overview into how to use our C++ implementation of a GDN simulator. Currently in its third revision, the package is designed to handle routines common to all models so that model creation and testing requires less time and effort. The source code can be accessed from https://bitbucket.org/MorganAndMore/graph-work. Note that the code requires the Boost graph library to compile (tested to work with Visual Studio 2015 v14.025431.01 and Boost library version 1.62.0). We compiled the This code was used to generate the data for the various figures and visualizations throughout this chapter.

The generality of GDN models means that creating new models is not as simple as specifying a few new parameters. One must evaluate new transition rates after every transition, evaluate how long to stay in the current state and select the next state. The procedure for creating a new GDN specification requires the creation of a class `CustomProcess` which inherits from `GeneralizedDynamicNetwork`. Then, in `CustomProcess`, override the functions `CreateInitialGraph` and `EvaluateRates` to match the desired GDN model.

Further details can be found in the header files of the code. Examples of overriding the aforementioned functions are found in the source files which implement a contact process and the opinion process (See Examples 4.1 and 4.2 respectively).

## 4.3 Parameter Fitting

Consider a GDN $\{G_t\}_{t\geq 0}$ with transition function $q(g, h)$, which is parametrized by vector $\boldsymbol{\theta}$. We wish to take an observed graph sequence $\{g_t\}_{t\geq 0}$ and estimate $\boldsymbol{\theta}$ such that the resulting process $\{G_t\}_{t\geq 0}$ is the most likely to have generated $\{g_t\}_{t\geq 0}$. We denote the "true" parameter

vector by $\boldsymbol{\theta}^*$ and an estimate of the "true" parameter vector by $\hat{\boldsymbol{\theta}}$.

Before introducing an advanced method for estimating $\boldsymbol{\theta}^* = (\theta_1^*, \theta_2^*, \ldots)$, we describe a rudimentary approach. If there exists a continuous, invertible function $k$, along with graphs $g_i$ and $h_i$ such that $q(g_i, h_i) = k(\theta_i)$, then it is possible to construct a simple asymptotically unbiased estimator for $\theta_i^*$ using the method of moments.

**Theorem 4.3.** *Let $T_1, T_2, \ldots, T_N$ be the holding times of state $g_i$ and $B_1, B_2, \ldots, B_N$ be Bernoulli random variables where successes mean that $g_i$ transitioned to $h_i$. Then*

$$\hat{\theta}_i := k^{-1}\left(\frac{\sum_j^N B_j}{\sum_j^N T_j}\right) \tag{4.6}$$

*is an asymptotically unbiased estimator for $\theta_i^*$.*

*Proof.* Under the assumption that $\{g_t\}$ is a realization of a GDN, $T_j \sim \exp(r)$ and $B_j \sim \text{Ber}(p)$ with $rp = k(\theta_i^*)$. Then

$$\begin{aligned}\lim_{N\to\infty} \hat{\theta}_i &= \lim_{N\to\infty} k^{-1}\left(\frac{\sum_j^N B_j}{\sum_j^N T_j}\right) \\ &= k^{-1}\left(\lim_{N\to\infty} \frac{N^{-1}\sum_j^N B_j}{N^{-1}\sum_j^N T_j}\right) \\ &= k^{-1}(rp),\end{aligned}$$

where the third equality holds due to the independence of holding times and resulting state and applying the law of large numbers. □

This approach is highly flawed, and is infeasible to use on non-trivial examples. The first problem is that the statespaces of interest are typically high-dimensional, and the method described above requires that we focus on a single state which will rarely be observed. This means the estimator will have high variance as $N$ will be quite low unless we spend a substantial amount of time generating the dynamic random graph.

The second problem is the unrealistic quality of data required to apply this method. If we were observing a network through surveys at fixed points, we would only know the graph up to a few statistics at times $t_1, t_2, \ldots, t_\tau$. This motivates the following approximate algorithm.

### 4.3.1 Approximate Maximum Likelihood Estimation

Let $\boldsymbol{s}(g_{t_1}), \ldots, \boldsymbol{s}(g_{t_\tau})$ be the data observed from a real-world dynamic network $\{g_t\}_{t\geq 0}$, where $\boldsymbol{s}$ is the vector of statistics collected at predetermined time points. Then, the likelihood of the data given $\boldsymbol{\theta}$ is

$$\ell(\boldsymbol{\theta}) = \mathbb{P}_\theta(\boldsymbol{s}(G_{t_1}) = \boldsymbol{s}(g_{t_1}), \ldots, \boldsymbol{s}(G_{t_\tau}) = \boldsymbol{s}(g_{t_\tau})), \tag{4.7}$$

where $G_t$ was generated under the selected GDN model with parameter vector $\boldsymbol{\theta}$, as indicated by the use of $\mathbb{P}_\theta$. The $\boldsymbol{\theta}$ which maximises $\ell(\boldsymbol{\theta})$ is a *maximum likelihood estimate* (MLE) of $\boldsymbol{\theta}^*$. Our goal is to create a method which finds the MLE for arbitrary GDN specifications.

Directly evaluating $\ell(\boldsymbol{\theta})$ for a given dataset requires integration over a set of functions. That is,

$$\ell(\boldsymbol{\theta}) = \int_A \mathbb{I}\left\{\boldsymbol{s}(h_{t_1}) = \boldsymbol{s}(g_{t_1}), \ldots, \boldsymbol{s}(h_{t_\tau}) = \boldsymbol{s}(g_{t_\tau})\right\} \mathrm{d}\{h_t\}_{t\geq 0}, \tag{4.8}$$

where the $A$ is some appropriate measurable set of functions and $\{h_t\}_{t\geq 0}$ is the complete trajectory of a continuous-time dynamic graph. Integrating with respect to functions (as opposed to integrating over a real-valued space) requires very advanced techniques and in general is too difficult to compute exactly. This motivates the use of estimation techniques to approximate the likelihood function. Our method draws inspiration from sequential filtering, see [14] for an overview. Furthermore, to reduce computation time we use a relaxation similar to a common technique in approximate Bayesian computation [70].

Crude Monte Carlo can be used to estimate $\ell(\boldsymbol{\theta})$, however the probability that a sample path $\{\boldsymbol{s}(G_t)\}_{t\geq 0}$ triggers the event $\{\boldsymbol{s}(G_{t_1}) = \boldsymbol{s}(g_{t_1}), \ldots, \boldsymbol{s}(G_{t_\tau}) = \boldsymbol{s}(g_{t_\tau})\}$ will be very low in most models. As this is a rare-event problem involving Markov chains, the splitting method (See Section 3.1) offers a potential solution.

Let $\mathcal{X}_1 = \{\{h_t\}_{t\geq 0} : \boldsymbol{s}(g_{t_1}) = \boldsymbol{s}(h_{t_1})\}$, where $\{h_u\}_{u\geq 0}$ is any dynamic graph. Furthermore, let $\mathcal{X}_i = \{\{h_t\}_{t\geq 0} : \boldsymbol{s}(g_{t_i}) = \boldsymbol{s}(h_{t_i})\} \cap \mathcal{X}_{i-1}$. Clearly this gives $\mathcal{X}_1 \supset \mathcal{X}_2 \supset \cdots \supset \mathcal{X}_\tau$. Finally let $\mathcal{X}_0 = \{\{h_t\}_{t\geq 0} : \exists i \text{ such that } \boldsymbol{s}(g_{t_i}) \neq \boldsymbol{s}(h_{t_i})\}$, so the sets $\mathcal{X}_i$, $i \in \{0, \ldots, \tau\}$ can be directly used with splitting as defined in Section 3.1. The continuous-time nature of the underlying GDN process does not cause any problems with Algorithm 3.1.

Meaningful results can still be obtained if the sets $\{\mathcal{X}_i\}$ are relaxed to permit more values. Specifically, take $\mathcal{X}_i = \{\{h_t\}_{t\geq 0} : \boldsymbol{s}(g_{t_i})(1-\epsilon) \leq \boldsymbol{s}(h_{t_i}) \leq s(g_{t_i})(1+\epsilon)\} \cap \mathcal{X}_{i-1}$, where $\epsilon > 0$ is the relaxation parameter. Larger $\epsilon$ means $\mathbb{P}(\{G_t\}_{t\geq 0} \in \mathcal{X}_i)$ is larger for each $i$ and thus

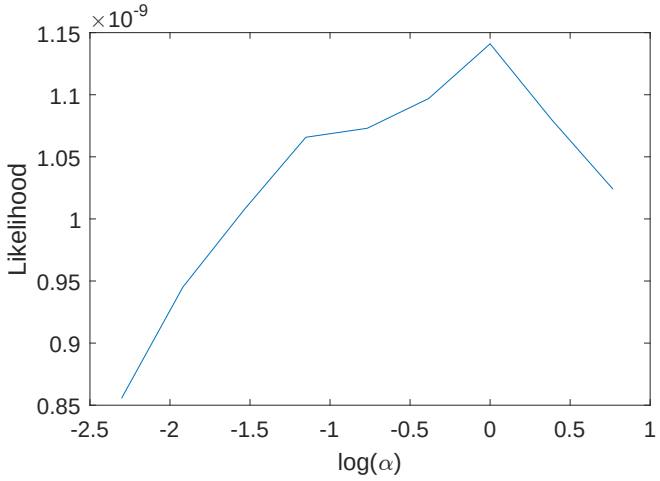



Figure 4.5: Results from running the approximate likelihood scheme for varying values of $\alpha$ and $\beta$. The plot is a cross section of the the likelihood surface where $\beta = 9\alpha$ — all values $(\alpha, \beta)$ not shown in this cross section returned a likelihood of zero. Note that $\log(\alpha) = 0$ is the true value.

Algorithm 3.1 requires fewer samples to produce a low variance estimator. Increasing $\epsilon$ also reduces the accuracy of the MLE, meaning it must be chosen carefully.

Now that estimations of $\ell(\theta)$ can be made, we can attempt to fit a GDN model to observed data. There is no guarantee that $\ell(\cdot)$ will be convex, thus it potentially has a set of optimal values rather than a single point. This makes maximum likelihood estimation challenging, as we will see in the next two examples.

**Example 4.3.** Consider finding the MLE for a GDN that fits the framework of Example 4.1. Specifically the parameters $\alpha$ and $\beta$ are of interest — to reduce the computation time of this example, all other parameters are assumed to be known. The data will be generated by observing a realization of the same GDN in Example 4.1 with the parameters $\boldsymbol{\theta}^* = (\lambda, \mu, \alpha, \beta, \nu, \eta, \gamma) = (0, 0, 0.1, 0.9, 0.1, 0.005, 0.05)$.

If the observation points $t_1, t_2, \ldots, t_\tau$ are taken too far apart, the graph process will reach stationarity between them. For estimating the rates of edge formation and destruction, the process at stationarity will only give us information about $\alpha/(\alpha + \beta)$, which is not enough for uniquely estimating both $\alpha$ and $\beta$.

To keep this simple, the only statistic of interest will be the edge-count. All parameters other than $\alpha$ and $\beta$ are considered known and zero. There will be a fixed count of twenty

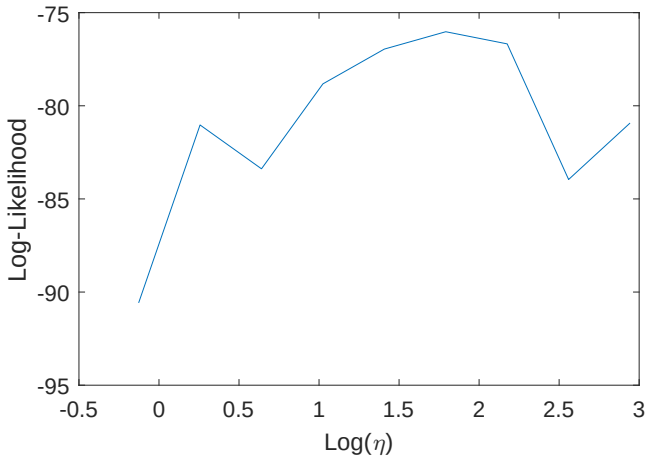



Figure 4.6: Results from running the approximate likelihood scheme for varying values of $\eta$ where $\log(\eta) = 1.79$ is the true value.

vertices in the model. We write $\hat{\ell}(\alpha, \beta)$ for the estimated likelihood of the data given the model with parameter vector $\boldsymbol{\theta} = (0, 0, \alpha, \beta, 0.1, 0.005, 0.05)$.

Figure 4.5 shows the behaviour of the maximum likelihood in the domain $[0.09, 9] \times [0.01, 1]$. The maximal value is attained at $\hat{\ell}(0.1, 0.9) = 3.144 \times 10^{-6}$, which matches the true values of $\alpha$ and $\beta$ in $\boldsymbol{\theta}^*$. ♦

**Example 4.4.** For our final example, we will use our likelihood estimator on the opinion network defined in Section 4.2. The goal is to find the maximum likelihood estimate of the rate that a person's beliefs are converted by one of their peers, represented by $\eta$.

The statistic vector $s$ selected returns the order statistics for the number of vertices holding a particular opinion. For example, let $g$ be the graph presented in Figure 4.4; then $s(g) = (12, 4, 3, 1)$.

We assume knowledge of all parameters except for $\eta$. The true process $\{G_t\}_{t \geq 0}$ was generated under parameters $(\gamma, \alpha, a, b, \eta, \nu) = (0.4, 1.2, 0.4, 40, 6, 0.02)$, with $s(G_t)$ recorded at times $0.05, 0.1, \ldots, 1$. Results of running the splitting algorithm likelihood estimation algorithm on various points is found in Figure 4.6.

To avoid the lower values of $\eta$ from attaining $\hat{\ell}(\eta)$ higher than the true value of $\eta = 6$, it was necessary to run the algorithm with the relaxation parameter $\epsilon$ set to zero. We see

that the estimate is relatively close, and importantly the very rapid rates are ruled out as possibilities. ♦

## 4.4 Discussion

The GDN has a great amount of flexibility, allowing for processes running atop the random graph, such as contact processes, to affect its structure. As seen with Example 4.2, the GDN opens new avenues for simulation studies. In the case of a SIS simulation study, this allows for quarantine effects to be applied to infected individuals. Many graph models in the literature, such as the one presented in [43], do not account for the infectives behaving differently to susceptibles. A preliminary study in [44] demonstrates this application of GDN by modelling a hospital quarantine.

Continuous-time Markov Chains simplify many aspects of model construction. For most models, the transition rates will only be positive for actions which only affect a single aspect of the graph. That is, a single vertex is born or dies, a single edge is born or dies, etc. This is seen in Example 4.1 where only six simple rates are needed to define a complete SIS model. In discrete-time models, this would need to be an assumption — that for any time step, multiple changes to the graph will not be seen. Furthermore, continuous-time models allow for the efficient handling of graphs that have periods of "downtime" when few transitions occur. In such cases, a discrete model would require multiple transitions from the current state to itself but a continuous-time model skips directly to the time when the state changes.

A significant problem for GDN models is that they are computationally intensive. Simulating a GDN process requires the re-evaluation of every transition rate after each transition. If any of these rates are of the order $O(n^2)$ or worse, it would take tremendous computation power to simulate graphs with expected vertex counts in the tens-of-thousands. This issue of scale is compounded by the infinite resolution of continuous-time. The more "activity" present in a process, the shorter each transition advances the simulation clock. For example, a trivial graph with one vertex and a single binary label that flips at rate 1 would be expected to advance time by 1 unit for each transition. A similar graph with $n = 10^6$ vertices (and no edges) flipping their labels independently would be expected to advance time by $10^{-6}$ units per transition.

This problem of scale could be significantly mitigated if instead of re-evaluating every rate

after a transition, only the rates which change are updated. For example, in Example 4.1 the rates associated with each vertex (including the incident edge birth and death rates) only change when the vertex itself or one of its neighbours are altered from a state transition. This property of rates entirely dependent on local structure is common in many graph models. Implementing this as an optional setting in the GDN simulation software package would require a major rewrite of the back-end systems, and as such was left off the current version.

As discussed at the start of Section 4.3, attempting to fit models to observed data using the classical approach requires precise knowledge of the transition times. Such information would not be available for most real-world dynamic networks. Using a Maximum Likelihood Estimator is a viable alternative, however it requires the evaluation of a highly complicated integral. We saw no way to create a general software system which could evaluate these integrals analytically, so we turned to Monte Carlo techniques.

The approach given in Section 4.3.1 was capable of providing estimates of the likelihood function $\hat{\ell}(\theta)$. Though it was demonstrated in Example 4.3 and 4.4 that this scheme can work as expected, there are issues which prevent this method from being used as an effective tool for fitting a model to real-world data. First, the time to compute an estimate $\hat{\ell}(\theta)$ is highly dependent on $\theta$. Specifically if a particular choice of $\theta$ causes high rates for the graph process, far more transitions would be required to advance the process to the required time. In computing the results for Example 4.3, this issue was observed through the debug output, where the choice of $\theta = (1, 9)$ took roughly ten times longer than $\theta = (0.1, 0.9)$. The lopsided nature of the estimation time makes pilot trials difficult — some results will be returned within seconds, others will take days. If a number of potential models are being considered, the MLE calculations for each should be run in parallel to avoid weeks of delay.

The next issue is that of the sensitivity of results to the $\epsilon$ hyperparameter. Setting $\epsilon$ too high produces overestimates $\hat{\ell}(\theta)$ for a $\theta$ that corresponds to graph processes with low transition rates. This is because if $\epsilon$ is high, the "window" to hit is so wide that not changing the process at all is preferred. Indeed, Example 4.4 required $\epsilon = 0$ to get reasonable results, as otherwise there were large overestimates for low $\eta$. Alternatively, if $\epsilon$ is too low, none of the simulated graph processes will hit the specified targets, resulting in an estimate of $\hat{\ell}(\theta) = 0$. This occurred in Example 4.3, with a handful of pilot trials necessary to find an appropriate value for $\epsilon$. To ensure

A key open question remains: whether using an approximate Bayesian computation approach would provide higher quality models over the MLE scheme proposed in this section. Approximate Bayesian computation has a wealth of literature discussing the quality of its results, see for example [39] and [80]. To the author's knowledge, similar results for the approach we used do not exist. A study investigating the two approaches to identify the pros and cons of each is an avenue of future research.

# Chapter 5

# Expected Typical Distance Estimation of Preferential Attachment Models

Six-degrees of separation is a commonly claimed property of our highly connected, international society. It asserts that every person can be linked to any other via a chain of six or fewer acquaintances on a first name basis. The notion that society is closely connected through friends-of-friends is referred to as the *small-world* phenomenon.

Studies of this phenomenon were popularized by Stanely Milgram's experiment in 1967 [72]. The experiment involved sending packages to random recipients in the United States and requesting that they be sent back to a member of the research team. The catch was that the package-holder could only send the package to someone they knew on a first name basis. By recording the number of exchanges needed for the package to reach its destination, the researchers estimated the typical distance between any two people in the country to be about 5.5. There have been numerous, more rigorous, studies of the small-world phenomenon since Milgram's famous experiment, see [81] and [43] for example.

The rise of digital communication and extensive data collection has allowed far greater accuracy regarding network observations. As of February 2016, the popular social networking site Facebook had over 1.6 billion users and complete data regarding the friendship network. Facebook released a statement that their calculations suggest the average distance between two randomly selected users is 4.57 [8].

One explanation for this *small-world* behaviour is that humans organize themselves through

a form of preferential attachment. This motivates us to investigate the expected typical distance (ETD) of the preferential attachment random graph model. To glean some further understanding of this phenomenon, see Definition 2.3.

We begin our investigation in Section 5.1 by proving some theorems related to expected typical distance. This is followed by an empirical study in Section 5.2 where we produce kernel density estimates for the typical distribution of various PA models. Section 5.3 and Section 5.4 discuss two approaches to estimating the tail probabilities: splitting and sequential importance sampling. Both sections contain algorithms used to implement each scheme and the related results.

In this chapter all graphs are taken to be **multigraphs**. Note that as we use the convention that self-loops are counted twice with regards to degree, every edge contributes two towards the total degree of the graph. Thus, the total degree sum of any graph is twice the number of edges.

## 5.1 Theoretical Results

This section is concerned with expected typical distance results for deterministic multigraphs. The results will directly translate into useful tools for the algorithms in the following sections. Recall that the typical distance from $u$ is defined as $D_g(u) = \frac{1}{|\mathcal{C}_g(u)|}\sum_{v\in\mathcal{C}_g(u)} d(v,u)$.

**Theorem 5.1.** *Consider graphs $g$ and $h$, where $h$ is formed by adding an edge $\{z,w\}$ between two disconnected components $\mathcal{C}_g(z)$ and $\mathcal{C}_g(w)$. Then if $\mathcal{C}_g(z)$ and $\mathcal{C}_g(w)$ have $m$ and $n$ vertices respectively,*

$$D_h(v) = \frac{mD_g(v) + n(D_g(w) + d_g(v,z) + 1)}{(m+n)}, \quad v \in \mathcal{C}_g(z). \tag{5.1}$$

*Proof.* Let $U$ be a uniformly chosen vertex from $h$. Then

$$\begin{aligned} D_h(v) &= \frac{1}{m+n}\sum_{u\in\mathcal{V}_h} d_h(v,u) \\ &= \frac{1}{m+n}\left(\sum_{u\in\mathcal{C}_g(z)} d_g(u,v) + \sum_{u\in\mathcal{C}_g(w)} d_g(v,z) + 1 + d_g(w,u)\right) \\ &= \frac{1}{m+n}\left(mD_g(v) + n(d_g(u,z)+1) + \sum_{u\in\mathcal{C}_g(w)} d_g(w,u)\right) \\ &= \frac{mD_g(v) + n(d_g(v,z) + D_g(w) + 1)}{m+n}. \end{aligned}$$

□

**Lemma 5.2.** *Consider the graph $h$ which is formed by adding an edge $\{z,w\}$ to $g$ such that $\mathcal{C}_g(z) = \mathcal{C}_g(w)$. Then $D(h) \le D(g)$.*

*Proof.* Recall

$$D(h) = \frac{1}{|\mathcal{M}_h|}\sum_{\{u,v\}\in\mathcal{M}_h} d_h(u,v), \tag{5.2}$$

where $\mathcal{M}_h$ is the set of all vertex pairs $u, v$ with $\mathcal{C}_h(v) = \mathcal{C}_h(u)$. Each shortest path in $h$ either traverses $\{z,w\}$ or it does not. As $h$ has all edges of $g$, the shortest paths of $g$ are still present in $h$. Without loss of generality, this gives $d_h(v,u) = \min\{d_g(v,z) + 1 + d_g(w,u), d_g(v,u)\}$. Thus $d_h(v,u) \le d_g(v,u)$ and $D(h) \le D(g)$ as $g$ and $h$ have the same components. □

A *bridge* is an edge whose removal would increase the number of components in the graph. The following corollary is concerned with edges that are not bridges.

**Corollary 5.3.** *If $g$ is constructed from $h$ by removing a non-bridge edge, then $D(g) \ge D(h)$.*

*Proof.* $h$ can be thought of as $g$ with an extra edge $\{z,w\}$ such that $\mathcal{C}_g(z) = \mathcal{C}_g(w)$. By Lemma 5.2 we have $D(g) \ge D(h)$. □

**Lemma 5.4.** *For a tree $g$ and path $p$ with the same number of vertices, $D(g) \le D(p)$.*

*Proof.* If $g$ is isomorphic to $p$ then $D(g) = D(p)$. So assume $g$ is not isomorphic to $p$, and thus $g$ must have a vertex $v_1$ with degree greater than 2.

Consider subgraph partition $h_1, h_2, h_3$ such that $v_1 \in \mathcal{V}_{h_3}$ and disconnected when $v_1$ is cut from $g$. Without loss of generality assume $|\mathcal{V}_{h_1}| \leq |\mathcal{V}_{h_2}| \leq |\mathcal{V}_{h_3}|$. There exists a path $v_1, v_2, \ldots, v_t$ where $v_t$ is of degree one and $v_t \in h_2$.

We will now generate a sequence of graphs starting at $g_1 = g$ and advancing from $g_i$ to $g_{i+1}$ via the following procedure. From the above definitions, $g_i$ has an edge from $v_i$ to some vertex $u \in h_1$. To create $g_{i+1}$, disconnect this edge and adding an edge from $v_{i+1}$ to $u$. This causes

$$d_{i+1}(z,w) \geq \begin{cases} d_i(z,w) + 1 & z \in h_1, w \in h_3 \\ d_i(z,w) & z \in h_2, w \in h_3 \\ d_i(z,w) - 1 & z \in h_1, w \in h_2, \end{cases} \tag{5.3}$$

where we have in equality the first two cases. Thus $D(g_{i+1}) \geq D(g_i) + |\mathcal{V}_{h_3}| - |\mathcal{V}_{h_2}| \geq D(g_i)$ by our assumption on the sizes of the subgraphs.

So by repeated application of the above procedure, we find $D(g) = D(g_1) \leq D(g_2) \cdots \leq D(g_t)$. Effectively we have moved the bridge — connecting subgraph $h_1$ to the rest of the graph — down the path $v_1, v_2, \ldots, v_t$. Furthermore, $g_t$ has one fewer vertex of degree 3. We can take $g_i$, find another vertex of degree 3 (or greater) and repeat the above procedure until all vertices are degree 2 or fewer. At this point we must have a path, and as every modification to the graph never reduced the expected typical distance, we have $D(g) \leq D(p)$. □

**Theorem 5.5.** *The graph with largest expected typical distance on $n$ vertices is an $n$ vertex path.*

*Proof.* Let $g \in \mathcal{G}$ be any $n$ vertex graph. From $g$ construct $h$ by sequentially removing non-bridge edges until we are left with a a collection of trees. Let $l$ be the graph constructed by converting each tree component into a path by the method described in Lemma 5.4. Finally connect each of the paths together to form a single $n$ vertex path $p$. From Corollary 5.3 and Lemma 5.4, $D(g) \leq D(h) \leq D(l) \leq D(p)$. □

## 5.2 Numerical Results

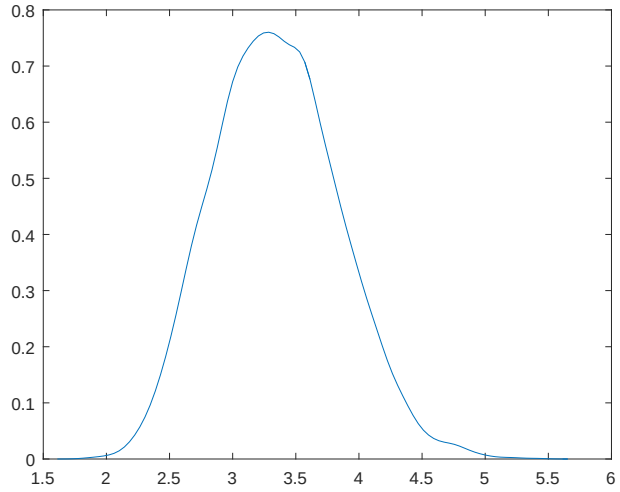
The aim of this section is to analyse the ETD of preferential attachment random graphs through crude Monte Carlo simulations. We begin by generating simple kernel density es-

timates (KDEs). The plots in Figure 5.1 were constructed by simulating many points using Algorithm 2.1, followed by applying the KDE algorithm specified in [45].

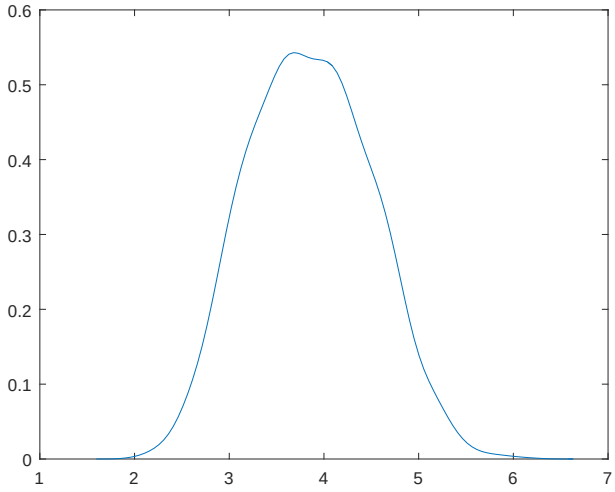
Recall from Section 2.4 that the typical distance of $\mathrm{PA}(n, m, \alpha)$ models are bounded by $O(\log \log n)$ and $O(\log n / \log \log n)$ functions in the $m \geq 2$, $\alpha < 0$ and $\alpha = 0$ cases respectively. We expect this to result in very low variance in these models, especially for large $n$. As this chapter is more concerned with observing graphs which have high ETD despite the small-world nature of the model, we focus on the $m = 1$ case and $\alpha \geq 0$.

By viewing the plots in Figure 5.1, it is possible to observe properties of the ETD distribution for varying parameters of the preferential attachment random graph. All of the plots are unimodal with most of the density concentrated around this mode. Increasing $\alpha$ shifts the mode to the right, increasing $m$ shifts the mode to the left. Figure 5.1(a) and 5.1(b) show a slight right skew on both distributions. Note that only $m = 1, 2$ are shown here as for $m \geq 2$ the typical distance distributions behave similarly to the $m = 2$ case, see [22].

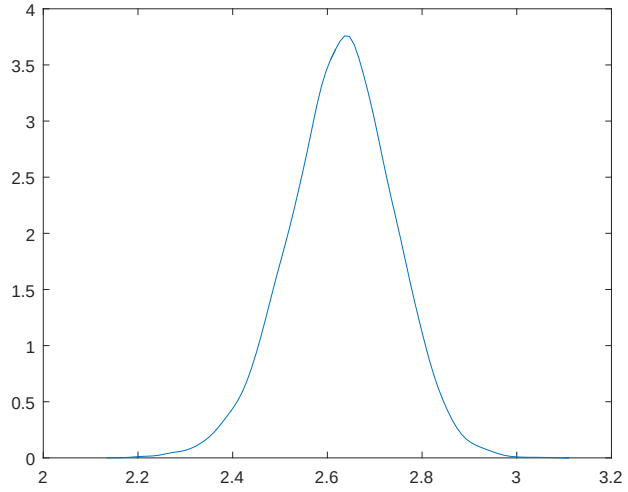
Of particular interest is $\mathbb{P}(D(G_t) > \gamma)$ when $\gamma$ is large. Such a probability would indicate that a self-forming network is inefficient at spreading information. Before attempting to estimate this probability, we first observe the behaviour of sample paths $\{G_t\}_{t \geq 0}$ which satisfy $\{D(G_{nm}) > \gamma\}$ for $\gamma$ relatively large; such sample paths are referred to as *elite*. To observe the behaviour of sample paths, many are generated as per a modified version of Algorithm 2.1 which records $D(G_t)$ at each time step. In Figure 5.2(a) it is demonstrated that for the $m = 1$ case the elite sample paths are easily distinguishable early in the process. Compare this with the $m = 2$ case shown in Figure 5.2(b), where the elite sample paths are only distinguishable after $t = 40$. The difference in behaviour of sample paths when we alter $m$ is likely attributable to $m = 2$ allowing the ETD to drop, whereas it is non-decreasing for $m = 1$, see Theorems 5.1 and Lemma 5.2.



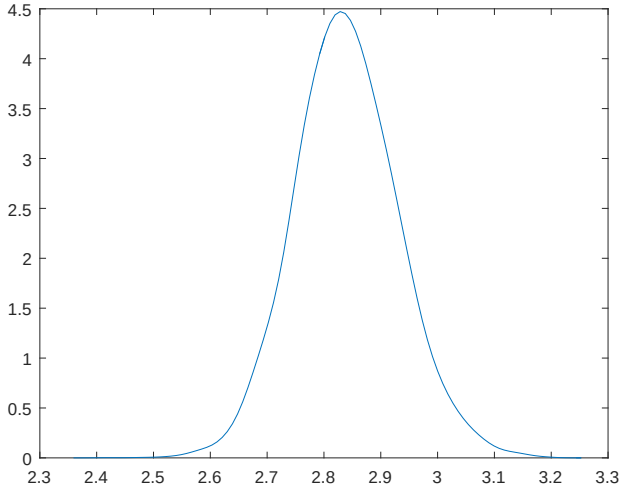

(a) KDE for $n = 50$, $\alpha = 0$, $m = 1$.




(b) KDE for $n = 50$, $\alpha = 3$, $m = 1$.

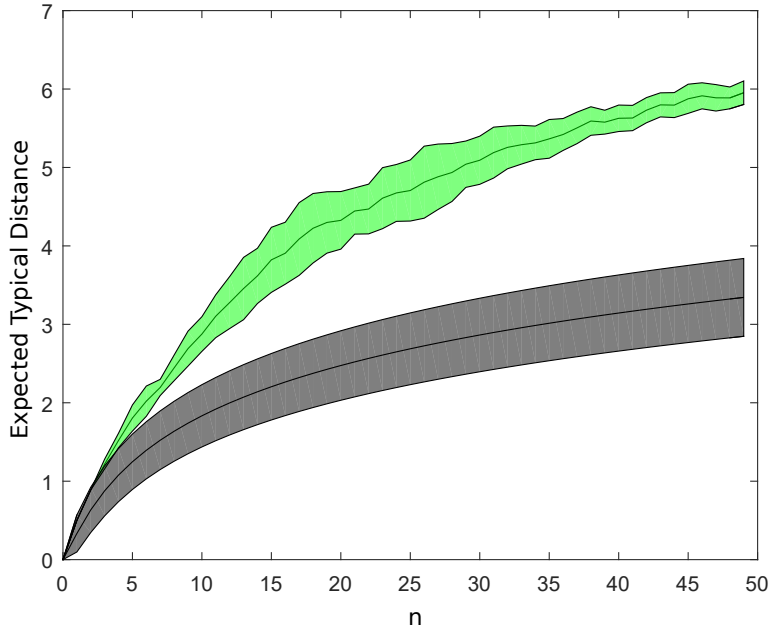



(c) KDE for $n = 50$, $\alpha = 0$, $m = 2$.

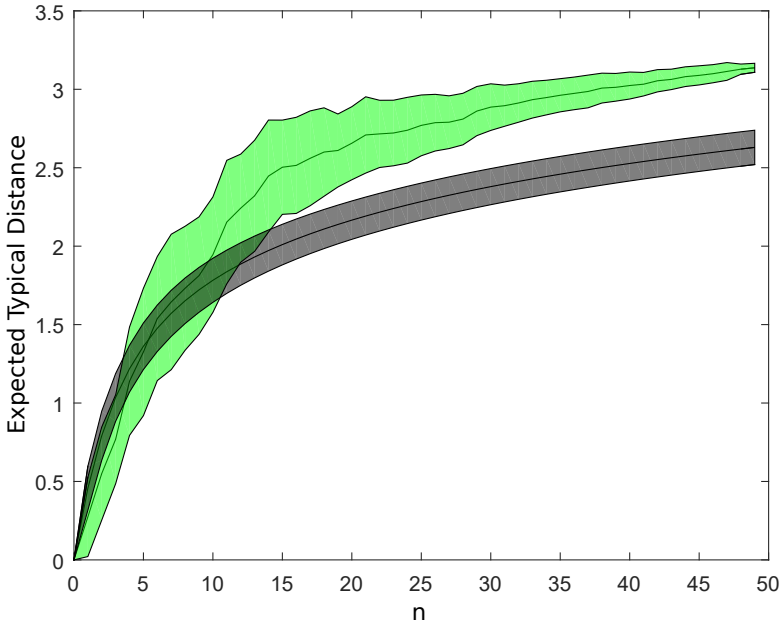



(d) KDE for $n = 50$, $\alpha = 3$, $m = 2$.

Figure 5.1: Kernel density estimates for $\alpha = 0, 3$ preferential attachment models. Each plot was generated using $10^5$ samples of the specified preferential attachment random graph.




(a) Mean paths for $n = 50$, $\alpha = 0$, $m = 1$. Here $\gamma = 5.8$.




(b) Mean paths for $n = 50$, $\alpha = 0$, $m = 2$. Here $\gamma = 3.1$

Figure 5.2: The coloured regions show an empirical estimate for the interval $[\mu_t - \sigma_t, \mu_t + \sigma_t]$ for $t = 0, 1, \ldots nm$, where $\mu_t = \mathbb{E}\, D(G_t)$, $\sigma_t = \sqrt{\operatorname{Var} D(G_t)}$ for the black region; and $\mu_t = \mathbb{E}(D(G_t) \,|\, D(G_{nm}) \geq \gamma)$, $\sigma_t = \sqrt{\operatorname{Var}(D(G_t) \,|\, D(G_{nm}) \geq \gamma)}$. The threshold $\gamma$ is specified under each subfigure, and was chosen so that the probability that $D(G_{nm})$ exceeds gamma is roughly $10^{-5}$.

## 5.3 Splitting

For large $\gamma$, $\{D(G_{nm}) > \gamma\}$ is a rare-event. To obtain accurate results in a reasonable timeframe, it is necessary to develop an advanced Monte Carlo estimator. In this section we apply the splitting method to create an estimator for $\mathbb{P}(D(G_{nm}) > \gamma)$, denote this estimator by $\hat{\ell}_{\text{split}}$

The graph process used to construct a preferential attachment model is a discrete-time Markov chain. In addition, Figure 5.2(a) and Figure 5.2(b) imply that we can identify elite samples with respect to ETD before the graph is completely constructed. The key difficulty in any splitting implementation is designing the sets $\mathcal{X}_0, \mathcal{X}_1, \ldots, \mathcal{X}_k$ so that $\hat{\ell}_{\text{split}}$ has the lowest variance possible. Set

$$\mathcal{X}_i = \left\{ g_t : \sum_{j=1}^{k} \mathbb{I}\{D(g_t) \geq \gamma_j, t \geq \tau_j\} \geq i \right\}, \tag{5.4}$$

where $\tau_1, \tau_2, \ldots, \tau_k$ are thresholds in time and $\gamma_1, \gamma_2, \ldots, \gamma_k$ are thresholds in expected typical distance. Having thresholds in time is necessary as $\mathbb{P}(D(G_{nm}) \geq \gamma \,|\, D(G_t) \geq \gamma)$ increases with $t$ for $m \geq 2$. That is, graphs with high ETD early in the generation process are at far greater risk of loosing the structure which causes their high ETD when more vertices and edges are added.

To perform splitting naively on the preferential attachment model, we can invoke Algorithm 3.1 with initial points $G_0^{(1)}, G_0^{(2)}, \ldots, G_0^{(N)}$ initialized to empty graphs; advancing $G_t^{(j)}$ through time is performed via the update technique described in Section 2.4 and; the sets $\mathcal{X}_i$ are as described in Equation 5.4.

It is important to note that for a graph $g$ on $n$ vertices, $D(g)$ is an $O(n^2)$ operation. As $D(g)$ must be carried out at each time step when advancing our graph processes $\{G_t^{(j)}\}$. This means that naive splitting has time complexity $O(n^3)$, compared to the far more manageable $O(n^2)$ of crude Monte Carlo. To improve the performance of the algorithm, we use a bounding scheme as described in the next section.

### 5.3.1 Bounding Scheme

Determining if a sample $G_t$ has entered one of the target sets $\mathcal{X}_i$ is a prohibitively long calculation on all but the most trivial of models. A solution is suggested from the observation

that if we have a sequence of sets $\mathcal{Y}_1, \mathcal{Y}_2, \ldots, \mathcal{Y}_k$ such that for each $i \in \{1, 2, \ldots, k\}$, $\mathcal{X}_i \subseteq \mathcal{Y}_i$, then $G_t \notin \mathcal{Y}_i$ implies $G_t \notin \mathcal{X}_i$. If determining $G_t \in \mathcal{Y}_i$ is significantly faster than determining $G_t \in \mathcal{X}_i$, the execution time of the algorithm can be greatly reduced. All that remains is to find an appropriate sequence of sets $\mathcal{Y}_1, \mathcal{Y}_2, \ldots, \mathcal{Y}_k$.

Recall that under our choice of vertex set $\mathcal{V}$, each vertex is labelled by a unique integer. So for vertices $v$ and $u$, $v \leq u$ means the the label of $v$ is smaller than the label of $u$.

Say that at time $t$ of our preferential attachment graph process construction we have two vectors of length $n$, $\boldsymbol{c}(t)$ and $\boldsymbol{b}(t)$. The vector $\boldsymbol{c}(t)$ satisfies, for all vertices $u$ and $v$, $\boldsymbol{c}_v(t) \leq v$ and $c_v(t) = c_u(t)$ if $\mathcal{C}_{G_t}(v) = \mathcal{C}_{G_t}(u)$. Thus $\boldsymbol{c}(t)$ contains information regarding the component of each vertex. The second vector $\boldsymbol{b}(t)$ has the property $b_v(t) \geq D_{G_t}(v)$ for every vertex $v$. Thus the following holds:

$$D^*(\boldsymbol{b}) := \frac{1}{|\mathcal{M}_{G_t}|} \sum_{v \in \mathcal{V}} \mathcal{C}_{G_t}(v) \boldsymbol{b}_v(t) \geq \frac{1}{|\mathcal{M}_{G_t}|} \sum_{v \in \mathcal{V}} \mathcal{C}_{G_t}(v) D_{G_t}(v) = D(G_t). \tag{5.5}$$

Note that we can use $\boldsymbol{c}(t)$ to find $\mathcal{C}_g(\cdot)$ and $\mathcal{M}_g$ terms in the above equation, where $\mathcal{M}_g$ is the set of all vertex pairs $\{u, v\}$ for which $\mathcal{C}_g(v) = \mathcal{C}_g(u)$.

Define the sets $\mathcal{Y}_1, \mathcal{Y}_2, \ldots, \mathcal{Y}_k$ by:

$$\mathcal{Y}_i = \left\{ g_t : \sum_{j=1}^{k} \mathbb{I}\{D^*(\boldsymbol{b}) \geq \gamma_j, t \geq \tau_j\} \geq i \right\}, \tag{5.6}$$

with $\tau_1, \tau_2, \ldots, \tau_k$ and $\gamma_1, \gamma_2, \ldots, \gamma_k$ matching the variables used for the corresponding $\mathcal{X}_i$ set (see Equation 5.4). As $D_{G_t}(v) \leq \boldsymbol{b}_v(t)$, it follows that $\mathcal{X}_i \subseteq \mathcal{Y}_i$.

The problem that remains is to ensure $\boldsymbol{c}(t)$ and $\boldsymbol{b}(t)$ correctly satisfy the properties assumed above. First set $\boldsymbol{c}_v(0) = v$ and $\boldsymbol{b}_v(0) = 0$. As $G_0$ of all preferential attachment random graphs is empty, the two vectors satisfy their respective requirements. Recall that the PA construction process always adds one edge to advance $G_t$ to time $t + 1$. Either the edge lies within a single component, or it bridges two disconnected components. In the former case, Lemma 5.1 ensures setting $\boldsymbol{b}(t + 1) = \boldsymbol{b}(t)$ and $\boldsymbol{c}(t + 1) = \boldsymbol{c}(t)$ maintains the properties we require of $\boldsymbol{b}$ and $\boldsymbol{c}$. In the latter case, say $\{z, w\}$ was the edge added, then Theorem 5.2 can be used to update each each $\boldsymbol{b}_v(t)$ with a single breadth-first search. Next set $\boldsymbol{c}_v(t + 1) = \min\{\boldsymbol{c}_z(t), \boldsymbol{c}_w(t)\}$ for $v \in \mathcal{C}_{G_t}(z) \cup \mathcal{C}_{G_t}(w)$ and for $v \notin \mathcal{C}_{G_t}(z) \cup \mathcal{C}_{G_t}(w)$, set $\boldsymbol{c}_v(t + 1) = \boldsymbol{c}_v(t)$. As finding $\boldsymbol{b}(t + 1)$ and $\boldsymbol{c}(t + 1)$ at-worst require $O(n)$ operations, checking $G_t \in \mathcal{Y}_i$ is faster than checking $G_t \in \mathcal{X}_i$, as required. Algorithm 5.1 gives complete details on how to perform this update scheme when the most recent edge added is between two disconnected components.

**Algorithm 5.1:** UpdateVector

**Input:** Component vector $\boldsymbol{c}$, ETD bound vector $\boldsymbol{b}$, graph $\mathcal{G}$, latest edge $\{z, w\}$.

**Output:** Updated vectors $\boldsymbol{c}$ and $\boldsymbol{b}$.

$m \leftarrow$ number of elements of $\boldsymbol{c}$ with $\boldsymbol{c}_v = \boldsymbol{c}_z$

$n \leftarrow$ number of elements of $\boldsymbol{c}$ with $\boldsymbol{c}_v = \boldsymbol{c}_w$

$\boldsymbol{b}^* \leftarrow \mathbf{0}$ of length $|\mathcal{V}_\mathcal{G}|$

$\Delta \leftarrow$ result from breadth first search starting at $z$ (so $\Delta_v = d_t(z, v)$)

**for** $v \in \{1, 2, \ldots, |\mathcal{V}_\mathcal{G}|\}$ **do**

  **if** $\boldsymbol{c}_j = \boldsymbol{c}_z$ **then**

    $\boldsymbol{b}^*_v \leftarrow \frac{m\boldsymbol{b}_v + n(\boldsymbol{b}_w + \Delta_v + 1)}{m+n}$

    $\boldsymbol{c}_v \leftarrow \min\{\boldsymbol{c}_z, \boldsymbol{c}_w\}$

  **else if** $\boldsymbol{c}_j = \boldsymbol{c}_w$ **then**

    $\boldsymbol{b}^*_v \leftarrow \frac{n\boldsymbol{b}_v + m(\boldsymbol{b}_z + \Delta_v)}{m+n}$

    $\boldsymbol{c}_v \leftarrow \min\{\boldsymbol{c}_z, \boldsymbol{c}_w\}$

  **else**

    $\boldsymbol{b}^*_v \leftarrow \boldsymbol{b}_v$

$\boldsymbol{b} \leftarrow \boldsymbol{b}^*$

**return** $\boldsymbol{c}, \boldsymbol{b}$

When it is found that $G_t \in \mathcal{Y}_i$, we must compute $D_{G_t}(v)$ for each vertex so that we can determine if $G_t \in \mathcal{X}_i$ is true. After computing $D_{G_t}(v)$, the evaluation of $\boldsymbol{b}_v(t+1)$ should use $D_{G_t}(v)$ in place of $\boldsymbol{b}_v(t)$ to ensure the best possible bounds.

Note that we start the process with exact bounds (i.e., $\boldsymbol{b}_v(0) = D_{G_0}(v)$) and Theorem 5.2 shows that if an edge is added between two components, $\boldsymbol{b}_v(t) = D_{G_t}(v)$ implies $\boldsymbol{b}_v(t+1) = D_{G_{t+1}}(v)$. This means that preferential attachment models with $m = 1$, $\mathcal{Y}_i = \mathcal{X}_i$, reducing the algorithm to an $O(n^2)$ operation. Empirical tests have shown that there is a significant speed boost when using the bounding scheme in the $m = 2$ case — up to fifteen times faster. This allowed us to perform experiments on graphs with $n = 50$.

**Algorithm 5.2:** ETD Splitting

**Input:** Fixed effort $N$, thresholds in time $\tau_1, \tau_2, \ldots, \tau_k$, thresholds in ETD $\gamma_1, \gamma_2, \ldots, \gamma_k$, preferential attachment parameters $n, m, \alpha$.

**Output:** Returns estimate $\hat{\ell}$ of $\mathbb{P}(A)$

**for** $j = 1, 2, \ldots, N$ **do**
- Generate empty graph $G^{(j)}$
- $\boldsymbol{b}^{(j)} \leftarrow \mathbf{0}$ (zero vector of length $n$)
- $\boldsymbol{c}^{(j)} \leftarrow (1, 2, \ldots, n)$

$\mathcal{Z} \leftarrow \{(G^{(1)}, \boldsymbol{b}^{(1)}, \boldsymbol{c}^{(1)}), \ldots, (G^{(N)}, \boldsymbol{b}^{(N)}, \boldsymbol{c}^{(N)})\}$

$\mathcal{Y} \leftarrow \emptyset$

**for** $i = 1, 2, \ldots, k$ **do**
- **for** $j = 1, 2, \ldots, N$ **do**
  - continue $\leftarrow$ true
  - Obtain $(G^{(j)}, \boldsymbol{b}^{(j)}, \boldsymbol{c}^{(j)})$ by reference from $\mathcal{Z}$
  - **while** *continue* **do**
    - $t \leftarrow |\mathcal{E}_{G^{(j)}}|$
    - **if** $t < nm$ **then**
      - Advance $G^{(j)}$ one time step under the rules imposed by $\mathrm{PA}(n, m, \alpha)$
      - UpdateVector$(\boldsymbol{c}, \boldsymbol{b}, G^{(j)})$
    - **if** $\sum_{l=1}^{k} \mathbb{I}\{D^*(\boldsymbol{b}^{(j)}) \geq \gamma_l, t \geq \tau_l\} \geq i$ **then**
      - $\boldsymbol{b}_v \leftarrow D_{G^{(j)}}(v)$ for all $v \in \mathcal{V}_{G^{(j)}}$
      - **if** $\sum_{l=1}^{k} \mathbb{I}\{D^*(\boldsymbol{b}^{(j)}) \geq \gamma_l, t \geq \tau_l\} \geq i$ **then**
        - $\mathcal{Y} \leftarrow \mathcal{Y} \cup \{(G^{(j)}, \boldsymbol{b}^{(j)}, \boldsymbol{c}^{(j)})\}$
    - **if** $t = nm$ **then**
      - continue $\leftarrow$ false
- $\mathcal{Z} \leftarrow \emptyset$
- **for** $j = 1, 2, \ldots, N$ **do**
  - Choose an $X_{\tau_i}^{(j)}$ uniformly at random from $\mathcal{Y}$
  - $\mathcal{Z} \leftarrow \mathcal{Z} \cup \{X_{\tau_i}^{(j)}\}$
- $N_i \leftarrow |\mathcal{Y}|$
- $\mathcal{Y} \leftarrow \emptyset$

**return** $N^{-\gamma} \prod_{i=1}^{\gamma} N_i$

The results in Table 5.1 demonstrate the poor performance of this splitting implementation, even with the major speed improvements provided by the bounding scheme. In order to produce times comparable to crude Monte Carlo, very few samples are allowed per splitting level. This issue is likely a result from using an inefficient approach for copying graphs.

Another issue is the splitting algorithm trends towards providing many underestimates and a small number of significant overestimates. This greatly increases the variance of the estimator and is particularly evident in the $\alpha = 3$ cases.

The time required to copy graphs is quite expensive as they are high-dimensional objects which are frequently moved during the execution of the algorithm. This motivates us to investigate an alternative estimator which does not require any copy operations.

## 5.4 Sequential Importance Sampling

Recall that our goal is to evaluate $\ell = \mathbb{P}(D(G) > \gamma)$ for $\gamma$ large, where $G$ is a $\mathrm{PA}(n, m, \alpha)$ random graph. For the rest of this chapter we restrict ourselves to $m = 1$; this allows us to easily construct a *sequential importance sampling* (SIS) estimator which outperforms the splitting estimator from the previous section. Section 5.5 contains a detailed explanation of why $m \geq 2$ is not considered for the SIS estimator.

We can uniquely determine $G_t$ using the vector $\boldsymbol{V}_t = (V_1, \ldots, V_t)$ of vertices selected as an end-point for each edge. Hence there exists a function $h$ such that $G_t = h(\boldsymbol{V}_t)$. Furthermore denote

$$f_{t+1}(v_{t+1} \mid \boldsymbol{v}_t) = \mathbb{P}(V_{t+1} = v \mid G_t = h(\boldsymbol{v}_t))$$

for $t = 1, 2, \ldots, n$; for convenience set $S(\boldsymbol{V}_t) := D(h(V_t))$.

We can sample from an importance sampling density (See Section 3.2) for each additional edge. Construct such a density by taking $k(\boldsymbol{v}_n) = k_1(v_1)k_2(v_2 \mid \boldsymbol{v}_1) \cdots k_n(v_n \mid \boldsymbol{v}_{n-1})$ where $k$ satisfies $k(\boldsymbol{v}_n) = 0 \implies \mathbb{I}\{S(\boldsymbol{V}_n) > \gamma\} f(\boldsymbol{v}_n) = 0$.

By taking the expectation with respect to $k$, we can represent $\ell$ as

$$\ell = \mathbb{E}_g \left[ \mathbb{I}\{S(\boldsymbol{V}_n) > \gamma\} \prod_{t=1}^{n} \frac{f(V_{t+1} \mid \boldsymbol{V}_t)}{k(V_{t+1} \mid \boldsymbol{V}_t)} \right]. \tag{5.7}$$

Assume it is easy to simulate from $k_1(v_1)$ to find $V_1$, and then simulate from $k_2(v_2 \mid v_1)$ to find

$V_2$ and so on, until we have $\boldsymbol{V}_n$ from $k(\boldsymbol{v}_n)$. As the nominal pdf $f$ has conditionals $f_t(v_t \mid \boldsymbol{v}_{t-1})$ that are easy to evaluate, we arrive at the following unbiased SIS estimator:

$$\hat{\ell} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}\{S(\boldsymbol{V}_n^{(i)}) > \gamma\} \prod_{t=1}^{n} \frac{f(V_{t+1}^{(i)} \mid \boldsymbol{V}_t^{(i)})}{k(V_{t+1}^{(i)} \mid \boldsymbol{V}_t^{(i)})}, \tag{5.8}$$

where $\boldsymbol{V}_n^{(1)}, \boldsymbol{V}_n^{(2)}, \ldots, \boldsymbol{V}_n^{(N)}$ are iid samples from $g$.

The remaining problem is to select $k_1, k_2, \ldots, k_n$ such that the variance of the estimator is minimized. We propose the following family of importance sampling densities:

$$k_{t+1}(v_{t+1} \mid \boldsymbol{v}_t) \propto \left(\deg_H(v_{t+1}) + \alpha + \delta_{v_{t+1},t+1}\right) \times \max\{1, D_H(v_{t+1})^p\}, \tag{5.9}$$

where $p \geq 0$ is a parameter of the density and $H = h(\boldsymbol{v}_t)$. Essentially this density means that for each new edge added to the graph, it is more likely that the edge will be incident with a vertex that has high distance from other vertices. Equation 5.1 shows that this will produce graphs with higher ETD, therefore sampling from $g$ means the event $D(G) > \gamma$ is more likely to occur. For $p = 0$ the estimator is identical to CMC, which will be useful in analysing the performance of the estimator.

### 5.4.1 Efficient Expected Typical Distance Evaluation

A major issue to be addressed is the expensive ETD calculation required to find $D_t(v)$ for every vertex $v$ at each time $t$. A naive implementation of the estimator would find the ETD using a *breadth first search* (BFS) from each vertex. Here each BFS requires $O(n)$ time and there are at most $n$ vertices, so evaluating the ETD takes $O(n^2)$ time. The ETD would need to be found before evaluating $k_{t+1}(v_{t+1} \mid \boldsymbol{v}_t)$ at each stage, hence it must be computed $n$ times. Thus a naive implementation would take $O(n^3)$ time to execute.

Compare this against CMC, which only requires a single ETD calculation after all edges are added. So CMC requires $O(n^2)$ time, which shows that a naive approach to implementing the SIS estimator is inadequate. Recall from Section 5.3.1 that it is possible to maintain a vector $\boldsymbol{b}$ such that $\boldsymbol{b}_v \geq D_{G_t}(v)$, with equality attained when $m = 1$ (i.e. when each step of the preferential attachment process adds one edge attached to one vertex). As we are only considering $\mathrm{PA}(n, 1, \alpha)$ random graphs, this bounding scheme can be used to find $D_{G_t}(v)$ at each time step with far fewer calculations The full SIS algorithm that incorporates this update scheme is given in Algorithm 5.3.

**Algorithm 5.3:** SIS estimator for $\mathbb{P}(D(G) > \gamma)$

**Input:** Number of vertices $n$, parameter $\alpha$, parameter $p$, threshold $\gamma$

**Output:** Estimate of $\mathbb{P}(G > \gamma)$ for $G$ in $\mathrm{PA}_n(\alpha)$

$W \leftarrow 0$

**for** $i = 1$ *to* $N$ **do**

  Construct $G_0$ as a graph with $n$ vertices

  $w = 1$

  $\boldsymbol{b} \leftarrow \mathbf{0}$

  **for** $j = 1$ *to* $n$ **do** $c_j = j$

  **for** $t = 0$ *to* $n - 1$ **do**

    $V_t \leftarrow$ vertex chosen randomly with density $k_{t+1}(v \mid \boldsymbol{V}_t)$

    $w \leftarrow w \times \frac{f_{t+1}(V_{t+1} \mid \boldsymbol{V}_t)}{k_{t+1}(V_{t+1} \mid \boldsymbol{V}_t)}$

    Construct $G_{t+1}$ by adding edge $\{t+1, V\}$ to $G_t$ to make $G_{t+1}$

    UpdateVector$(\boldsymbol{c}, \boldsymbol{b}, G_{t+1}, V, t+1)$

  **if** $(\sum_{j=1}^{n} b_j)/n > \gamma$ **then** $W \leftarrow W + w$

**return** $W/N$

The asymptotic time complexity of the SIS estimator with this update scheme is on par with CMC.

**Proposition 5.6.** *Algorithm 5.3 has asymptotic time complexity $O(n^2)$, where $n$ is the number of vertices in the random graph.*

*Proof.* There are $n$ iterations for a $\mathrm{PA}_n(\alpha)$ random graph. At each iteration the vertex $V_t$ must be selected and vectors $\boldsymbol{b}$ and $\boldsymbol{c}$ must be updated. The subroutine for choosing $V_t$ in Algorithm 5.3 loops over vertices $1, \ldots, t$. As $t \leq n$, this subroutine is $O(n)$. Within the same iteration we perform single breadth first search which takes $O(n)$ time. Counting the size of the components from $\boldsymbol{c}$ and then updating every element of $\boldsymbol{b}$ and $\boldsymbol{c}$ requires two separate loops over both vectors, and hence requires $O(n)$ time. As all other operations are not dependent on $n$, the full algorithm requires $O(n(n + n + n)) = O(n^2)$ time to run. □

Tables 5.2 through 5.3 demonstrate the performance of the estimator. Here $N$ is the number of graphs generated per estimate. The *relative error* RE of each estimator is itself an estimate,

calculated as

$$\mathrm{RE}(\hat{Z}) = \frac{S}{\bar{X}\sqrt{N}}, \tag{5.10}$$

where $N$ is the number of independent runs of the estimator, and $\bar{X}$ and $S$ are the sample mean and standard deviation of the estimator $\hat{Z}$. To compare trials with differing times, we define the *numerical efficiency* NE of estimator $\hat{Z}$ as

$$\mathrm{NE}(\hat{Z}) = \mathrm{Time}_{\hat{Z}}\mathrm{RE}(\hat{Z})^2 \tag{5.11}$$

where $\mathrm{Time}_{\hat{Z}}$ is the execution time of the estimation algorithm. Under the assumption that each of the $N$ runs of the algorithm is independent, $\mathrm{Time}_{\hat{Z}} \approx cN$ for some constant $c$. Thus $\mathrm{NE}(\hat{Z})$ should be invariant under changes in $N$. Estimators with low $\mathrm{NE}(\hat{Z})$ are more efficient, and only estimators of the same problem (with same parameters) may be fairly compared.

It is clear from Table 5.2 that the estimator outperforms CMC for appropriate parameter selection. Setting the parameter $p$ too high causes the estimator to increase in variance — note that the sample variance will be low because the large deviations anticipated become rare. For the problem described in Table 5.2 the optimal value for $p$ appears to be between 0.8 and 1.2.

Finding the optimal value can be done through a pilot run with fewer samples. Empirical tests show that the estimator is stable, so increasing the number of runs by a factor of $K$ reduces the relative error by a factor of $\sqrt{K}$.

Finally, we see in Figure 5.3 that the use of an update scheme is essential for ensuring the SIS estimator is viable for large $n$.

Table 5.1: Performance comparison of estimators for $\mathbb{P}(D(G_{nm}) > \gamma)$. The parameter $p$ for SIS was set to 1 in all cases. SIS and CMC used $N = 10^7$ samples each, and splitting used $N = 10000$ samples per level, with six levels.

| Parameters | Estimator | Estimate | Relative Error | Time (seconds) | Efficiency |
|---|---|---|---|---|---|
| $\gamma = 6.0, n = 50,$ $m = 1, \alpha = 0$ | CMC | 1.10e-6 | 0.3020 | 14692 | 1331 |
| | SIS | 8.25e-7 | 0.0260 | 11732 | 8 |
| | Splitting | 1.21e-6 | 0.1624 | 5885 | 155 |
| $\gamma = 6.8, n = 50,$ $m = 1, \alpha = 3$ | CMC | 2.6e-6 | 0.1961 | 11583 | 445 |
| | SIS | 2.41e-6 | 0.0418 | 14828 | 26 |
| | Splitting | 1.91e-6 | 0.1169 | 9986 | 136 |
| $\gamma = 3.2, n = 50,$ $m = 2, \alpha = 0$ | CMC | 1.30e-6 | 0.2773 | 20545 | 1580 |
| | Splitting | 1.67e-6 | 0.3677 | 12304 | 1663 |
| $\gamma = 3.3, n = 50,$ $m = 2, \alpha = 3$ | CMC | 8.57e-6 | 0.1291 | 13188 | 220 |
| | Splitting | 1.08e-7 | 0.3773 | 12665 | 1802 |

Table 5.2: SIS estimator results for varying parameter $p$, with $n = 50, \alpha = 0, N = 10^7, \gamma = 6.0$. Recall that larger values of $p$ force the algorithm to generate random graph realizations with larger expected typical distance.

| Parameter $p$ | 0 | 0.2 | 0.4 | 0.6 | 0.8 | 1 | 1.2 |
|---|---|---|---|---|---|---|---|
| Mean | 1.10e-6 | 7.26e-7 | 7.96e-7 | 8.05e-7 | 8.29e-7 | 8.25e-7 | 7.90e-7 |
| Relative Error | 0.302 | 0.185 | 0.096 | 0.061 | 0.037 | 0.026 | 0.020 |
| Time (seconds) | 14692 | 11852 | 12848 | 11204 | 12052 | 11732 | 13308 |
| Efficiency | 1340 | 406 | 118 | 42 | 16 | 8 | 5 |

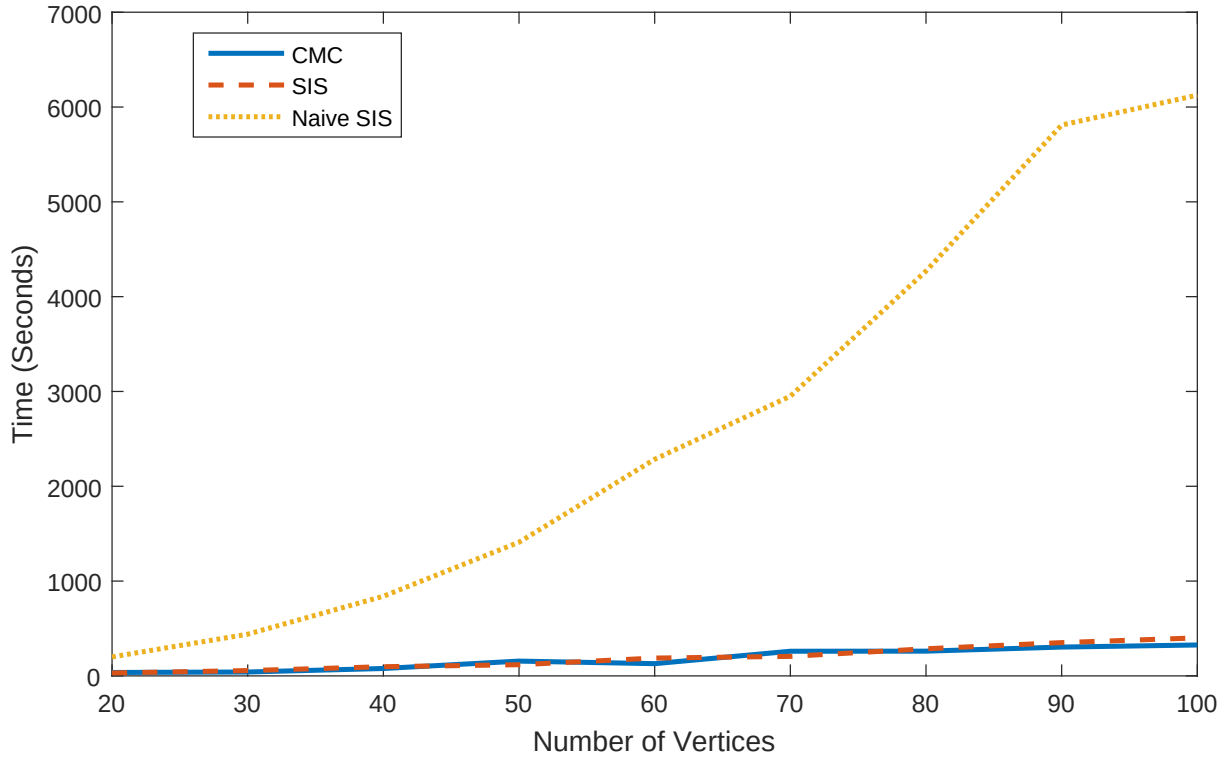



Figure 5.3: Time required for each algorithm to complete. Here $\alpha = 0$, $N = 10000$. Naive SIS is the implementation of the SIS algorithm described at the start of Section 5.4.1

## 5.5 Discussion

The performance of the splitting estimator was hampered by its high computation times. Even with the bounding scheme implemented, the algorithm took too long to execute, forcing fewer samples per splitting level. These long execution times are likely the result of two contributing factors. The first is that splitting, in contrast to the other estimators described in this chapter, requires graphs to be copied in memory. If graphs are stored using an adjacency matrix, this is an $O(n^2)$ operation. The second, more important issue is that there are more full expected typical distance calculations in the splitting implementation, when compared to the other algorithms. For $m \geq 2$, it is necessary to evaluate the expected typical distance at least once per sample per splitting level. To keep the run time reasonable, this results in a necessary

Table 5.3: Comparison of CMC and SIS estimation results for varying $\alpha$ and $\gamma$, with fixed $N = 10^7, n = 50$. For SIS, $p = 1$. Time is in seconds.

| Parameters | Method | Estimate | Relative Error | Time (seconds) | Efficiency |
|---|---|---|---|---|---|
| $\alpha = 0, \gamma = 4.5$ | CMC | 4.53e-5 | 0.0470 | 4202 | 9.28 |
| | SIS | 3.60e-5 | 0.0098 | 3812 | 0.36 |
| $\alpha = 0, \gamma = 4.75$ | CMC | 6.60e-6 | 0.1231 | 3135 | 47.50 |
| | SIS | 4.90e-6 | 0.0211 | 3326 | 1.48 |
| $\alpha = 0, \gamma = 5.0$ | CMC | 5.00e-7 | 0.4472 | 3110 | 621.96 |
| | SIS | 6.18e-7 | 0.0486 | 3685 | 8.70 |
| $\alpha = -0.75, \gamma = 3.75$ | CMC | 4.77e-5 | 0.0458 | 4493 | 9.42 |
| | SIS | 3.49e-5 | 0.0126 | 3922 | 0.62 |
| $\alpha = -0.75, \gamma = 4.0$ | CMC | 4.40e-6 | 0.1508 | 4117 | 93.62 |
| | SIS | 3.52e-6 | 0.0313 | 3596 | 3.52 |
| $\alpha = -0.75, \gamma = 4.25$ | CMC | 5.00e-7 | 0.4082 | 3140 | 523.21 |
| | SIS | 3.11e-7 | 0.0892 | 3972 | 31.60 |
| $\alpha = 3, \gamma = 5$ | CMC | 5.15e-5 | 0.0441 | 3172 | 6.16 |
| | SIS | 3.35e-5 | 0.0090 | 3406 | 0.28 |
| $\alpha = 3, \gamma = 5.25$ | CMC | 9.50e-6 | 0.1026 | 2559 | 26.94 |
| | SIS | 5.49e-6 | 0.0180 | 3118 | 1.01 |
| $\alpha = 3, \gamma = 5.5$ | CMC | 1.30e-6 | 0.2773 | 2527 | 194.31 |
| | SIS | 7.22e-7 | 0.0406 | 3157 | 5.20 |

reduction in the total number of samples by an order of magnitude.

Using the estimators described in this chapter on larger graphs has multiple drawbacks. The most evident is that a full expected typical distance calculation has $O(n^2)$ time complexity. If the problem of time was somehow mitigated, large graphs are still high dimensional objects and Sequential Monte Carlo methods struggle to accurately solve high dimensional problems [59]. Furthermore, as $n$ gets large, the typical distance converges to $O(\log n)$ (or smaller) for the models considered. This means that graphs with large expected typical distance would become less and less common as $n$ increases. Thus studying the behaviour of larger graphs would be best left to a theoretical methods rather than simulation. Specifically, after around $n = 100$ vertices, the estimator would either take too long to compute or be too inaccurate to use.

An application of these estimators is the ability to generate high ETD samples via resampling. For example, if one was to obtain the graphs $\{G^{(1)}, G^{(2)}, \ldots, G^{(N)}\}$ and the corresponding importance weights $\{w_1, w_2, \ldots, w_N\}$ from the SIS estimator, they could resample the graphs according to the weights. The resulting samples would be approximate samples from $\mathrm{PA}(n, m, \alpha)$ restricted to only $D(G) \geq \gamma$ graphs. A tool such as this could provide insights for similar statistics by allowing a researcher to directly observe extreme graphs.

Before arriving at the density in Equation 5.9, others were considered and tested. We devised an importance density $k$ that assigned additional probability to the vertex which was most recently added. The idea was that this would encourage long chains which increase the typical distance and cause the target event $\{D(G) \geq \gamma\}$ to occur more frequently. Experimentally we found that the variance of this estimator was higher than for CMC. While sampling under $k$ does trigger the target event, it does not reproduce the conditions which have the highest chance to trigger the event under $f$, thus increasing the SIS estimator's variance.

For the $m \geq 2$ case we can use the $k$ from Equation 5.9 but only for times $t = 1, m+1, 2m+1, \ldots$. That is, the first edge added between the new vertex and the rest of the graph. The efficient ETD update no longer works exactly — it will instead form a bound as in Section 5.3.1. Experiments showed that both using the bound and performing full calculations resulted in minimal variance reduction but a large increase in computation time. The minimal reduction is very likely caused by a simple observation: attaching a new vertex $v$ to a $u$ with high ETD will increase the overall ETD for that time step, but the next step will add an edge from $v$ to the rest of the graph as per usual. As a result, any increase to the ETD from the importance sampled

step is undone. We tried to remedy this by using another density for $t = 2, m + 2, 2m + 2, \ldots$ which prioritised connecting $v$ to vertices close to $u$ but this had little effect on the estimator variance.

It is possible to naively apply the SIS estimator to the tail probability of typical distance (that is, estimating $\mathbb{P}(\mathrm{TD}(G) \geq \gamma)$). This is done by running the estimator as in Algorithm 5.3 normally but replacing line 14 with $W \leftarrow W + w \times \mathbb{P}(\mathrm{TD}(G_{nm}) \geq \gamma)$ However our tests of this produced poor results compared to the CMC estimator. It is likely that the graphs with large typical distance exhibit different structural properties to graphs with large ETD. A specialized estimator would need to be developed to obtain accurate results.

# Chapter 6

# Exponential Random Graph Model Simulation

The National Longitudinal Study of Adolescent to Adult Health (Add Health) gathered relationship data between students from various schools in the United States. The data contained information regarding which pairs of students were friends, along with the age, sex and race of each student. A random graph can be used to model the collected data, with vertices and edges representing students and friendships respectively. In [37] and [32], an exponential random graph (ERG) model was used to model the Add Health dataset (see Section 2.3 for an introduction to ERG models).

To model data using an ERG, one must select appropriate graph statistics. In [32], the chosen graph statistics included basic subgraph counts such as edge and triangle count, as well as graph statistics that incorporate the dataset's supplementary information, such as the number of same-sex friendships. When an ERG model is fit to the collected data, the model parameters provide insight into the real-world network. For example, if a parameter weighting same-sex friendships is larger than the corresponding parameter weighting opposite-sex friendships, it indicates a preference among students for homogeneous partnerships.

Fitting an ERG model to data usually requires a large number of simulations [68]. Simulating from an ERG model is very difficult in non-trivial cases — the normalization constant is intractable and MCMC has poor convergence to stationarity [17]. We are motivated to apply the stratified splitting algorithm (see Section 3.3) in an attempt to use the ERG model effectively on a wider range of models.

We begin this chapter by discussing the basic approaches for estimating and simulating from ERG models in Section 6.1. The stratified splitting algorithm (SSA) is applied to the problem in Section 6.2. We conclude the chapter with Section 6.3, which contains results from applying the various methods stated to three models, each with their own challenges.

In this chapter we are only concerned with **simple graphs**. No loops or parallel edges are permitted.

## 6.1 Basic Methods

For completeness we will first describe how to apply basic techniques to estimating and simulating from an arbitrary ERG. We will use these techniques later for comparison with our advanced SSA approach.

Note that throughout this chapter, $\bar{f}$ is taken to be the unnormalized probability density function of the ERG. As $\bar{f}$ often takes on extreme values, it is necessary to use the log-sum-exp function to prevent floating-point underflow.

Underflow and overflow are serious problems when dealing with ERGs and other high-dimensional probability systems. The log-sum-exp (LSE) function allows us to work in the log domain for addition operations, which mitigates the limitations of finite precision. Specifically we define a function $\text{LSE}(w_1, w_2, \ldots, w_n)$ which takes log-domain variables $w_i$ and computes their linear domain sum.

The precise formulation is

$$\text{LSE}(w_1, \ldots, w_m) = w^* + \log(\exp(w_1 - w^*) + \cdots + \exp(w_m - w^*)), \tag{6.1}$$

where $w^* = \max\{w_1, \ldots, w_m\}$. It is trivial to show that this is mathematically equivalent to naively converting all log-domain variables to linear, taking their sum and then converting back to the log-domain. However, by bringing the largest weight $w^*$ and noting that $\exp(w_1 - w^*) + \cdots + \exp(w_m - w^*) \geq 1$, we can see that $\text{LSE}(w_1, \ldots, w_m) \geq w^*$ and thus underflow will be avoided.

As techniques such as Gibbs sampling and SSA operate on vectors, not graphs, we introduce a technique for converting simple graphs to binary vectors. Consider a graph $g$ with $\mathcal{V}_g =$

$\{1, 2, \ldots, n\}$. For vertices $v, u \in \mathcal{V}_g$ with $v < u$, let $i = (u-2)(u-1)/2 + v$. Next construct a vector $\boldsymbol{x}$ of length $\binom{n}{2}$ with $x_i = 1$ if edge $\{v, u\}$ is present in $g$, and zero otherwise. From this description of $\boldsymbol{x}$ it is clear that every graph $g$ on $n$ vertices corresponds to a unique binary vector $\boldsymbol{x}$ of length $\binom{n}{2}$. For simplicity, define $\bar{f}(\boldsymbol{x}) := \bar{f}(g)$, where $g$ is the graph constructed from binary vector $\boldsymbol{x}$.

### 6.1.1 Exact Simulation

Simulating from distributions with finite support is usually a trivial matter — one can apply the inverse-transform method [45] and iterate through a list of possible outcomes. Applying this naively to an ERG would be extremely inefficient, requiring a $O(t(d)2^d)$ operation for every sample, where $t(d)$ is the worst-case time for evaluating the necessary graph statistics and $d = \binom{n}{2}$. Some rudimentary tests on an ERG with statistics that count the number of 2-stars, 3-stars and triangles show that a single sample from an $n = 7$ model requires 10.1 seconds on average. For this model $t(\binom{n}{2}) = n^3$, so we can extrapolate an approximation for the time required to generate one million samples for the $n = 8$ case — it would take $1.93 \times 10^9$ seconds, or 61 years.

An alternative approach can be used where only a single $O(t(d)2^d)$ operation is needed, and subsequent samples can be generated at a cost of $O(d)$ per sample. While this is an improvement, in terms of time, it explodes the space complexity to $O(2^d)$ which is too large for $n \geq 9$.

By Theorem 2.4 we know that the set of all graphs with $n$ vertices can be enumerated $g_1, g_2, \ldots$. Let $U$ be a uniform random variable on the interval $[0, 1]$. To generate a random graph $G$ by the inverse transform method, $g_j$ is assigned a value $a_j$ in the interval $[0, 1]$ so that $\mathbb{P}(a_{j-1} \leq U \leq a_j) = \mathbb{P}(G = g_j)$ (take $a_0 = 0$) [62]. Then upon generating a realization of $U$, to generate $G$ simply find the interval $a_{j-1} \leq U \leq a_j$, then take $G = g_j$. As the number of graphs on $n$ vertices grows exponentially with $n$, it would be ideal to construct a system which allows for rapidly finding the interval $a_{j-1} \leq U \leq a_j$.

Recall that all graphs $g$ can be expressed as a binary vector $\boldsymbol{x}$. The binary nature of $\boldsymbol{x}$ means all possible binary vectors of length $n$, thus all possible graphs on $n$ vertices, can be stored implicitly in a binary tree. That is, every path taken from the root node to a leaf node can be viewed as a sequence of $d$ binary choices, 0 if the path moves through the left

child, 1 if the path moves through the right. Order all binary vectors $\boldsymbol{x}$ lexicographically ($\boldsymbol{x}_1 = (\ldots, 0, 0), \boldsymbol{x}_2 = (\ldots, 0, 1), \boldsymbol{x}_3 = (\ldots, 1, 0)$ and so on). Set each internal node of the binary tree to contain the value $a_{j^*}$, where $j^*$ is the largest index in the left subtree from the node. With such a binary tree, finding the interval $(a_{j-1}, a_j]$ which contains $U$ can be done in $d = \binom{n}{2}$ steps, a significant improvement over $2^d$.

Algorithm 6.1 will construct the binary tree when called with the parameters $0, [\,], 0, d$. To use this tree for simulation, we employ Algorithm 6.2.

**Proposition 6.1.** *Simulating $M$ samples from an $n$ vertex ERG under the binary tree approach above requires $O(t(d)2^d + Md)$ time and $O(2^d)$ space, where $t(d)$ is the number of computations required to evaluate $\bar{f}(\boldsymbol{x})$.*

*Proof.* To simulate $M$ samples we must build the tree by Algorithm 6.1 and then run Algorithm 6.2 $M$ times. All steps of Algorithm 6.1 are constant except for evaluating the statistics and calling the recursion. Thus constructing all non-leaf nodes requires $O(2^d)$ time and constructing all leaf nodes requires $O(t(d)2^d)$, producing a total running time of $O(t(d)2^d)$.

Next, we note that to generate samples from the tree, we must traverse from the root to the appropriate leaf, determined by a uniformly generated number $U$. Evaluating such a $U$ is constant time with most algorithms, and the tree has height $d$. Therefore generating $M$ samples from the ERGM tree requires $O(Md)$ time. This gives a total time of $O(t(d)2^d + Md)$ as required.

In terms of space, the binary tree has $2^d$ leaf nodes and $2^d - 1$ non-leaf nodes. As every node in the tree requires a constant amount of space, the space complexity is $O(2 \times 2^d + 1) = O(2^d)$. □

**Algorithm 6.1:** BuildERGMTree

**Input:** curent depth level of tree $l$, binary vector $\boldsymbol{x} = (x_1, \ldots, x_{l-1})$, reference of cumulative weight $w$, total number of edges $d := \binom{n}{2}$.

**Output:** Returns the node created.

node $\leftarrow$ empty node

**if** $l = d$ **then**
  $w \leftarrow w + \bar{f}(\boldsymbol{x})$

**else**
  node.leftChild $\leftarrow$ BuildERGMTree($l + 1$, $[\boldsymbol{x}\ 0]$, $w$, $d$)
  node.leftChild.parent = node
  node.value $\leftarrow$ w
  node.rightChild $\leftarrow$ BuildERGMTree($l + 1$, $[\boldsymbol{x}\ 1]$, $w$, $d$)
  node.rightChild.parent = node

**return** *node*

**Algorithm 6.2:** Full ERGM Simulation

**Input:** Binary Tree $T$ constructed using the above algorithm, the total weight sum $w$.

**Output:** One sample from the respective ERGM.

$i \leftarrow 0$, $\boldsymbol{x} \leftarrow [\,]$

$U \sim \mathsf{Uniform}(0, 1)$

node $\leftarrow$ root of tree $T$

**while** $i < d$ **do**
  **if** $wU <$ *node.value* **then**
    node $\leftarrow$ node.leftChild
    $\boldsymbol{x} \leftarrow [\boldsymbol{x}\ 0]$
  **else**
    node $\leftarrow$ node.rightChild
    $\boldsymbol{x} \leftarrow [\boldsymbol{x}\ 1]$

Construct graph $G$ from $\boldsymbol{x}$

**return** $G$

**Algorithm 6.3:** Gibbs Sampling Algorithm

**Input:** Unnormalized log-domain density $\log \bar{f}$, initial guess $\boldsymbol{X}_0$, iterations $T$

**Output:** Approximate samples $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_T$

**for** $t = 1, 2, \ldots T-1$ **do**

- $M \sim \mathsf{DU}(1, d)$
- $w_1 \leftarrow \log \bar{f}(\boldsymbol{X}_t)$
- $w_2 \leftarrow \log \bar{f}(\boldsymbol{X}_t \bigoplus \boldsymbol{e}_M)$
- $\alpha \leftarrow \exp(w_1 - \mathrm{LSE}(w_1, w_2))$
- $U \sim \mathsf{U}(0, 1)$
- **if** $u \leq \alpha$ **then** $\boldsymbol{X}_{t+1} \leftarrow \boldsymbol{X}_t$
- **else** $\boldsymbol{X}_{t+1} \leftarrow \boldsymbol{X}_t \bigoplus \boldsymbol{e}_M$

**return** $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_T$

### 6.1.2 Markov Chain Monte Carlo

The most common method for sampling from ERGs is to use a Gibbs sampler. For completeness, we now state the Gibbs Sampler algorithm for exponential random graph models. Note that $\boldsymbol{e}_i$ is taken as the vector with a 1 at position $i$ and zeroes everywhere else, and $\bigoplus$ denotes addition modulo 2.

The density of an exponential random graph often has multiple sharp peaks which means the probability of a Gibbs sampler leaving local maxima is incredibly rare. This problem can be difficult to detect in larger models, so results obtained with MCMC may be incorrect despite appearing reasonable.

## 6.2 Stratified Splitting Algorithm

We propose the stratified splitting algorithm (SSA) as an alternative approach to ERG sampling and estimation. Refer to Section 3.4 for an introduction to this method. As it utilizes a series of particles, and the particles transition across the space uniformly, the problem of poor mixing induced by extreme maxima is mitigated. The algorithm we use is identical to Algorithm 3.3 but all weights $w_1, w_2, \ldots$ are handled in the log-domain, using LSE where necessary.

When deploying SSA, a density $\varphi$ must be specified. The default choice of $\varphi$ is a density

with some sense of uniformity over space, for example the density associated with $\mathrm{ER}(n, \frac{1}{2})$ random graphs. However most graphs on $n$ vertices have around $\frac{d}{2}$ edges, which is far more than the expected number of edges for useful ERG models. This slows down SSA by forcing the algorithm to focus its searching effort in the wrong region of the state space.

A possible alternative choice of $\varphi$ is

$$\varphi(g) = \binom{d}{|\mathcal{V}_g|}^{-1} \tag{6.2}$$

which weights each possible graph so that the probability of getting a graph with exactly $i$ edges is uniformly distributed.

## 6.3 Models and Results

To compare the performance of SSA and Gibbs, we will use the estimators to empirically estimate the expected value for a variety of graph statistics, under different ERG models. The three models in this chapter were chosen to cover three cases: a common model from the literature (alternating triangles), a custom model to demonstrate the misleading results produced by Gibbs sampling (strictly business) and a trivial model which can be analytically solved for $n \geq 10$ (edge valley). The following subsections define each of these models and the parameters we used for our simulations.

Plots showing the individual estimates over many independent runs of each estimator are given in Figure 6.1 to 6.4. The differences between the estimators are demonstrated by the empirical mean squared error (MSE) shown in Table 6.1.

The MSE is defined as follows:

$$\mathrm{MSE}(\hat{\ell}) = \mathbb{E}[(\ell - \hat{\ell})^2], \tag{6.3}$$

where $\ell$ is the value we wish to estimate and $\hat{\ell}$ is the random variable corresponding to the output of the estimator of interest. It is easy to show that $\mathrm{MSE}(\hat{\ell}) = \mathrm{Var}(\hat{\ell}) + \mathrm{Bias}(\ell, \hat{\ell})^2$, where $\mathrm{Bias}(\ell, \hat{\ell}) = \mathbb{E}[\hat{\ell}] - \ell$. We will estimate the MSE through our simulations.

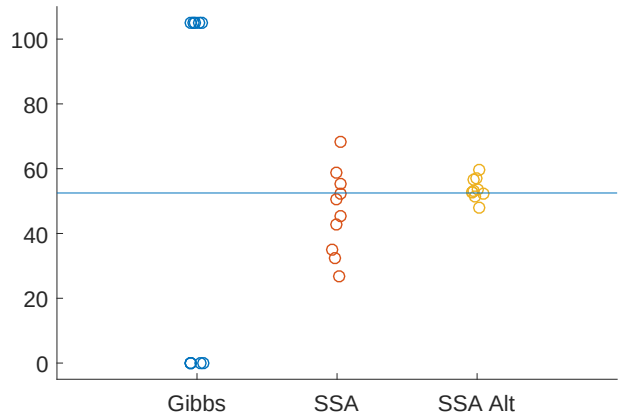



Figure 6.1: Estimates of the expected number of edges for the Edge Valley model with $n = 15, \theta = 0.1$. The horizontal line represents the true value. The middle column is for SSA with density $\varphi(g) = \frac{1}{2^d}$, the right column is for SSA with the density specified in Equation 6.2

### 6.3.1 Edge Valley

*Edge valley* is an ERG model where the only statistic is $T(g) = (|\mathcal{E}_g| - \frac{d}{2})^2$, where $d = \binom{n}{2}$ is the total number of edges possible in the model. This model creates an extreme bimodal distribution with an analytically tractable normalizing constant for large $n$. We choose $n = 15$ and $\theta = 0.1$ for testing, where $\theta$ is the parameter weighting the statistic $T(g)$. These parameters were found to best demonstrate the bimodal effect which traps a Gibbs sampler — increasing $n$ further started to cause issues which will be discussed shortly.

Figure 6.1 shows that both SSA and the MCMC estimator struggle to find the correct answer. SSA demonstrates that more samples per estimation are needed for it to perform well, whereas increasing the number of samples will not help MCMC get any closer to the correct result. This is the only case we found where using SSA with the alternate density specified in Equation 6.2 performs better than the default, uniform $h$.

### 6.3.2 Alternating Stars and Triangles

To produce non-degenerate models with an ERG model it is necessary to have statistics that do not cause a positive feedback loop. For example in the Markov model, promoting the appearance of triangle subgraphs also promotes the appearance of complete graphs of higher order. This issue led to the concept of alternating $k$-stars and $k$-triangles [69], an approach for

extending star and triangle graph statistics to avoid the aforementioned feedback loop.

The alternating $k$-triangle statistic is defined as follows

$$T_1(g) := \sum_{i<j} \mathbb{I}\{\{i,j\} \in \mathcal{E}_g\} \sum_{k=1}^{n-2} \binom{L_{ij}(g)}{k} \left(\frac{-1}{\lambda}\right)^{k-1}, \tag{6.4}$$

where $L_{ij}$ is the number of vertices in $g$ for which $i$ and $j$ are mutually adjacent. Next, the alternating $k$-star statistic is

$$T_2(g) := \sum_{k=2}^{n-1} S_k(g) \left(\frac{-1}{\lambda}\right)^{k-2}, \tag{6.5}$$

where $S_k$ is the number of $k$-stars in $g$. The final statistic for this model is simply $T_3(g) = |\mathcal{E}_g|$.

Using parameter vector $\theta = (2, -1, -1)$, we obtained the results shown in Figure 6.2 and Table 6.1. Gibbs performing well is expected, as the alternating stars/triangles model was developed to avoid degenerate behaviour. Here the key result is that SSA does not perform worse, allowing it to be used as a general tool for small vertex exponential random graph models.

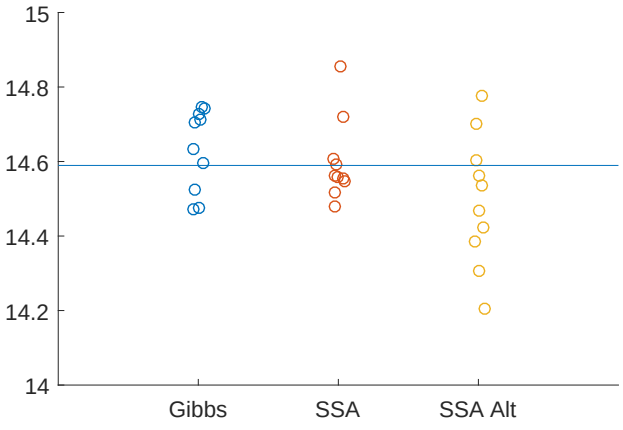



Figure 6.2: Estimates of the expected number of 3-stars in the alternating stars/triangles model. The horizontal line represents the true value. The middle column is for SSA with density $\varphi(g) = \frac{1}{2^d}$, the right column is for SSA with the density specified in Equation 6.2.

### 6.3.3 Strictly Business

The third model, called *strictly business*, uses a multilevel model [76] to simulate an organization where individuals must be related to exactly one department. The model sorts vertices into two

categories, type-A and type-B. Here the type-A vertices are treated as business departments and the type-B vertices represent individual workers.

We take $T(g) = (a_1, a_2, a_3, a_4)$, where

- $a_1$ is the number of edges between type-A vertices
- $a_2$ is the number of edges between type-B vertices
- $a_3$ is the number of type-B vertices which do not connect to exactly one type-A vertex
- $a_4$ is the *efficiency* of all type-B to type-B paths.

The efficiency is defined as $\sum_{v,u \in U} d(u,v)^{-1}$, where $u, v$ are distinct vertices in some subset $U$ and $d(\cdot,\cdot)$ is the usual graph distance function.

As the goal of the model is to generate a network where each type-B has one adjacent type-A vertex, $\theta_3$ is taken to be a significant negative number. This causes the Gibbs sampler, as defined in Algorithm 6.3 to break. For a Gibbs sampler to traverse between various configurations (e.g. two type-A nodes with degree 3 to one of degree 2 and the other of degree 4) it will need to propose and accept a transition into a graph with $a_3 > 0$ and thus it will have very low weight. If we were to simply have the Gibbs sampler update $k$ edges in a block it introduces a new problem: we now have $\binom{d}{k}$ possible transitions from each state, with only a handful leading to worthwhile states. This means that we would require a specialized transition kernel for Gibbs, which violates our goal of developing a simple "plug-and-play" algorithm.

For our simulations, we set $\theta = (-0.02, -5, -100, 0.005)$, $n = 7$ and set two vertices to be type-A, the rest are type-B. The large negative weighting on $\theta_3$ ensures that the random graph distribution prioritises graphs where each type-B vertex has exactly one adjacent type-A vertex. The other parameters were chosen arbitrarily.

As Gibbs will get stuck on specific organization allotments, the results in Figure 6.4 are expected. Note how it is not immediately obvious that Gibbs has failed if only the edge count statistic is considered, as in Figure 6.3. This is why it is important to use a variety of statistics when evaluating the performance of an estimator. The full table of MSE results for each statistic is shown in Table 6.1.

Note that the failure of SSA-Alt is an unusual result. It appears that the algorithm terminates far sooner than the default SSA scheme, causing poor exploration of the graph space.

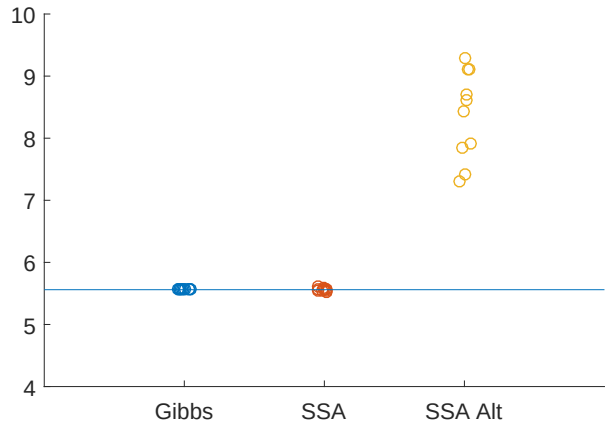



Figure 6.3: Estimates of the expected number of edges for the strictly business model. The horizontal line represents the true value. The middle column is for SSA with density $\varphi(g) = \frac{1}{2^d}$, the right column is for SSA with the density specified in Equation 6.2.

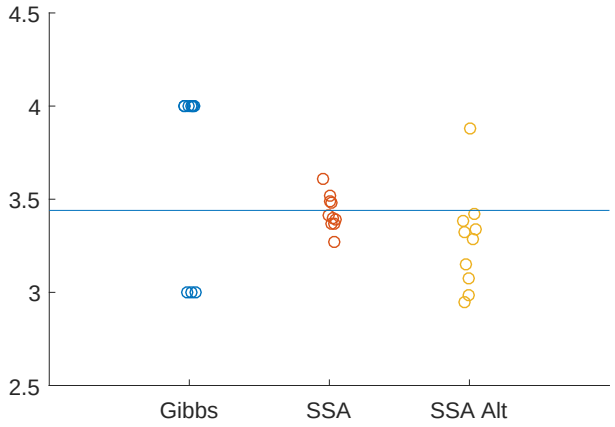



Figure 6.4: Estimates of the expected number of employees in the highest degree organization (Type-A vertex) for the strictly business model. The horizontal line represents the true value. The middle column is for SSA with density $\varphi(g) = \frac{1}{2^d}$, the right column is for SSA with the density specified in Equation 6.2.

| Model (Statistic) | Gibbs | SSA | SSA Alt. |
|---|---|---|---|
| Edge Valley (Edges) | 9.0250e+03 | 1.1813e+03 | 7.8141e+01 |
| Strictly Business (Edges) | 3.4769e-05 | 1.8432e-04 | 1.0908e+01 |
| Strictly Business (3-Stars) | 1.3655e+01 | 3.6989e-02 | 1.5029e+01 |
| Strictly Business (Triangles) | 1.5871e-04 | 1.3668e-06 | 5.0026e+00 |
| Strictly Business (See Figure 6.4) | 4.1712e-01 | 1.1482e-03 | 1.1689e-01 |
| Alternating (Edges) | 6.6297e-3 | 3.4660e-03 | 1.6968e-02 |
| Alternating (3-Stars) | 3.4451e-01 | 2.3831e-01 | 1.0291e-00 |
| Alternating (Triangles) | 5.2700e-02 | 1.1090e-03 | 3.6423e-02 |

Table 6.1: Table comparing the mean squared error for various statistics of the models of interest. Each estimator was run ten times on the model with parameters defined in each model's respective section.

## 6.4 Discussion

The results above give credence to SSA serving as general algorithm for small graphs. However many practitioners desire models with over one thousand vertices, and our tests with SSA on graphs larger than 20 vertices on the edge valley and strictly business performs on par with MCMC. That is, both algorithms fail to completely explore the state space, and thus their accuracy is poor. In general, large vertex counts cause the low probability states to far outnumber the high probability states, effectively making them harder to find. As a result, the algorithm requires more iterations. Not only does this increase the time required to execute the algorithm, it also decreases its accuracy. This is because the product estimator $\widehat{P}_t$ in Algorithm 3.3 increases in variance as more terms are added to the product.

An important result is shown in Figure 6.3. Both the Gibbs sampling estimator and SSA estimate the correct number of edges, despite Figure 6.4 demonstrating that the Gibbs sampler is failing to properly explore the state space. The deceptive result from the Gibbs sampler is why one should use a wide battery of graph statistics to verify the performance of any sampler.

It should be noted that the statistics in Table 6.1 are not exhaustive — it grants evidence that SSA succeeds where the Gibbs sampler fails, but does not guarantee this will be the case

for all graph models on the same number of vertices. The development of standard graph statistics for testing simulation methods would be a good avenue for future research.

One may wonder why it is not standard practice to simply run multiple, independent chains of the Gibbs sampler and take an average of the results. This averaging estimator is only valid if the chains have mixed properly, but if this was true it would not be necessary to run multiple chains at all. It is not difficult to construct an ERG which breaks this averaged estimator — for example a variant of the edge valley model where the peak in probability at no edges is half that of the peak at maximum edges. In such a model, a Gibbs sampler would have even chance of getting stuck at either the empty or complete graph, yet it should spend a third of the time at the empty graph state.

Table 6.1 shows that the SSA estimator consistently outperforms crude Monte Carlo via Gibbs sampling for small graphs. From this we infer that the approximate samples generated by SSA are closer to the true distribution than Gibbs samples. Further work investigating this could be to perform a Kolmogorov–Smirnov test on the various graph statistics to measure the distance between the approximate distributions and the true distribution.

An open problem for future research is to find a $\varphi$ density for SSA which performs better for large graphs. The two choices presented in this chapter (See Section 6.2) are general but do not weight sparse graphs higher than their dense counterparts. Finding a more suitable $\varphi$ density would greatly improve the applicability of the SSA estimator, enabling it to be used on larger graphs.

# Chapter 7

# Conclusion

The theme of this thesis was to investigate the application of sequential Monte Carlo methods to random graph models. To that end, we began with defining random graphs in Chapter 2. We gave random graphs the full measure theoretic treatment, and provided a proof that the set of all multigraphs is countable. This was followed by the introduction of three popular models from the literature: the Erdős–Rényi, preferential attachment and exponential random graph models.

Chapter 3 introduced the simulation tools required for the subsequent chapters. Algorithms and an abridged background for the splitting method, importance sampling and the stratified splitting algorithm were provided. As these tools operate on general, multidimensional objects, we are able to apply them to random graphs.

Chapter 4 defined the generalized dynamic network, a random graph model that evolves through continuous time. Two aspects of this model that make it novel. The first is that has weights which can change through time. This means that additional processes, such as contact processes, can be simulated on the graph as it evolves. The second benefit of the model is that these weights are able to affect the underlying graph topology. This trait means that effects such as quarantining infected individuals in a SIS model can be simulated. The network model was demonstrated by two examples, a simple susceptible-infected-susceptible model, and a social network model inspired by the well-known voter model. Links to a code repository are provided, if the reader wishes to experiment with the model themself. The chapter also contains work on fitting the graphs to data. The continuous time and general nature of the network makes it difficult to specify a 'one-size-fits-all' method. As the standard approaches would

not work, we turn to a sequential Monte Carlo alternative: approximate maximum likelihood estimation via splitting. On low dimensional models it works, however it takes a significant amount of computation time to achieve accurate results.

The fifth chapter focussed on the expected typical distance (ETD) of preferential attachment models. The chapter starts with some small theoretical results regarding how the ETD changes as the underlying graph is modified. For example, adding an edge between any two vertices in the same component cannot increase the ETD. The theoretical study is followed by numerical simulations. These simulations demonstrate that, during the construction of a preferential attachment graph realization, graphs which result in a high ETD can be identified. This lead to studying the use of sequential Monte Carlo methods to find the probability that the ETD is large. The splitting algorithm was not as effective as hoped — each sample required too much computation time to process. This lead to the application of sequential importance sampling, which proved effective in $m = 1$ models, but not $m \geq 2$.

The sixth chapter was concerned with simulating from exponential random graph (ERG) models and estimating quantities of interest. The chapter begins with a description on how to use perform a full simulation from an ERG model. By consuming a significant amount of memory, it is possible to save time — however the exponential time complexity is still far too steep for non-trivial graphs. This leads to the application of the stratified splitting algorithm (SSA) to the estimation problem, with a byproduct of samples approximately distributed according to the underlying model. To test the algorithm, a Gibbs sampler and SSA were applied to three models and the results compared. Gibbs demonstrated misleading results by producing accurate estimates for simple statistics but clearly erroneous estimates for sophisticated graph statistic tests. SSA did not suffer from such a problem, highlighting its potential use on smaller graph models.

As we have seen, sequential Monte Carlo methods can provide solutions to modern random graph simulation problems. The key drawback is graph size — sequential Monte Carlo struggles to perform efficiently in higher dimensions, limiting the scope of the problems it can solve. However, as demonstrated throughout this thesis, sequential Monte Carlo methods do work well on smaller problems. Further research could find new tricks and techniques to expand the types of problems sequential Monte Carlo can be applied to, as well as improving efficiency on the problems investigated in this thesis.

# Chapter 8

# Bibliography


[1] Lada A Adamic and Bernardo A Huberman. Power-law distribution of the world wide web. *Science*, 287(5461):2115–2115, 2000.

[2] Réka Albert, Hawoong Jeong, and Albert-László Barabási. Internet: Diameter of the world-wide web. *Nature*, 401(6749):130–131, 1999.

[3] Noga Alon, Michael Krivelevich, and Benny Sudakov. Finding a large hidden clique in a random graph. *Random Structures and Algorithms*, 13(3-4):457–466, 1998.

[4] Noga Alon and Joel H Spencer. *The Probabilistic Method*. New York: Wiley-Interscience., 2nd edition, 2000.

[5] Søren Asmussen, Klemens Binswanger, Bjarne Højgaard, et al. Rare events simulation for heavy-tailed distributions. *Bernoulli*, 6(2):303–322, 2000.

[6] Søren Asmussen and Dirk P Kroese. Improved algorithms for rare event simulation with heavy tails. *Advances in Applied Probability*, 38(2):545–558, 2006.

[7] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[8] Smriti Bhagat, Moira Burke, Carlos Diuk, Ismail Onur Filiz, and Sergey Edunov. Three and a half degrees of separation. *Facebook Research*, 2016.

[9] Shankar Bhamidi, Guy Bresler, and Allan Sly. Mixing time of exponential random graphs. In *IEEE 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 803–812. IEEE, 2008.

[10] Patrick Billingsley. *Probability And Measure*. John Wiley & Sons, third edition, 1995.

[11] Béla Bollobás and Oliver Riordan. The diameter of a scale-free random graph. *Combinatorica*, 24(1):5–34, 2004.

[12] Béla Bollobás, Oliver Riordan, Joel Spencer, and Gábor Tusnády. The degree sequence of a scale-free random graph process. *Random Structures & Algorithms*, 18(3):279–290, 2001.

[13] Tom Britton, Mathias Lindholm, and Tatyana Turova. A dynamic network in a dynamic population: asymptotic properties. *Applied Probability Trust*, 48:1163 – 1178, 2011.

[14] Olivier Cappé, Simon J Godsill, and Eric Moulines. An overview of existing methods and recent advances in sequential monte carlo. *Proceedings of the IEEE*, 95(5):899–924, 2007.

[15] Frédéric Cérou, Arnaud Guyader, Reuven Rubinstein, and Radislav Vaisman. On the use of smoothing to improve the performance of the splitting method. *Stochastic Models*, 27(4):629–650, 2011.

[16] Shirshendu Chatterjee and Rick Durrett. Contact processes on random graphs with power law degree distributions have critical value 0. *The Annals of Probability*, 37(6):2332–2356, 2009.

[17] Sourav Chatterjee and Persi Diaconis. Estimating and understanding exponential random graph models. *The Annals of Statistics*, 41(5):2428–2461, 2013.

[18] Nicholas A Christakis and James H Fowler. Social network sensors for early detection of contagious outbreaks. *PloS one*, 5(9), 2010.

[19] Political Compass. The political compass, 2001. Available at https://www.politicalcompass.org.

[20] Arman Danesh, Ljiljana Trajkovic, Stuart H Rubin, and Michael H Smith. Mapping the internet. In *IFSA World Congress and 20th NAFIPS International Conference, 2001. Joint 9th*, volume 2, pages 687–692. IEEE, 2001.

[21] Steffen Dereich, Christian Mönch, and Peter Mörters. Typical distances in ultrasmall random networks. *Advances in Applied Probability*, 44(2):583–601, 2012.

[22] Sander Dommers, Remco Van Der Hofstad, and Gerard Hooghiemstra. Diameters in preferential attachment models. *Journal of Statistical Physics*, 139(1):72–107, 2010.

[23] Rick Durrett. *Random Graph Dynamics*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2006.

[24] Paul Erdős and Alfréd Rényi. On random graphs. *Publicationes Mathematicae Debrecen*, 6:290–297, 1959.

[25] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power-law relationships of the internet topology. In *ACM SIGCOMM Computer Communication Review*, volume 29, pages 251–262. ACM, 1999.

[26] Andrew Gelman, John B Carlin, Hal S Stern, David B Dunson, Aki Vehtari, and Donald B Rubin. *Bayesian data analysis*, volume 2. CRC press Boca Raton, FL, 2014.

[27] Andrew Gelman and Donald B Rubin. Inference from iterative simulation using multiple sequences. *Statistical Science*, pages 457–472, 1992.

[28] Edgar N Gilbert. Random graphs. *The Annals of Mathematical Statistics*, 30(4):1141–1144, 1959.

[29] Walter R Gilks and Carlo Berzuini. Following a moving target—Monte Carlo inference for dynamic Bayesian models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(1):127–146, 2001.

[30] Walter R Gilks and Gareth O Roberts. Strategies for improving mcmc. *Markov chain Monte Carlo in practice*, 6:89–114, 1996.

[31] Steven M Goodreau. Advances in exponential random graph (p*) models applied to a large social network. *Social Networks*, 29(2):231–248, 2007.

[32] Steven M Goodreau, James A Kitts, and Martina Morris. Birds of a feather, or friend of a friend? using exponential random graph models to investigate adolescent social networks. *Demography*, 46(1):103–125, 2009.

[33] John Hammersley and D. C. Handscomb. *Monte Carlo methods*. John Wiley & Sons, 1964.

[34] W Keith Hastings. Monte Carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[35] Till Hoffmann, Mason A Porter, and Renaud Lambiotte. Random walks on stochastic temporal networks. In *Temporal Networks*, pages 295–313. Springer, 2013.

[36] T Hotta. Mean-field approximation. In *Nanoscale Phase Separation and Colossal Magnetoresistance*, pages 157–167. Springer, 2003.

[37] David R Hunter, Steven M Goodreau, and Mark S Handcock. Goodness of fit of social network models. *Journal of the American Statistical Association*, 103(481):248–258, 2008.

[38] Emmanuel Jacob and Peter Mörters. The spread of infections on evolving scale-free networks. *arXiv preprint arXiv:1512.00832*, 2015.

[39] Ajay Jasra, Sumeetpal S Singh, James S Martin, and Emma McCoy. Filtering via approximate Bayesian computation. *Statistics and Computing*, pages 1–15, 2012.

[40] M Vernon Johns. Importance sampling for bootstrap confidence intervals. *Journal of the American Statistical Association*, 83(403):709–714, 1988.

[41] James Holland Jones and Mark S Handcock. An assessment of preferential attachment as a mechanism for human sexual network formation. *Proceedings of the Royal Society of London B: Biological Sciences*, 270(1520):1123–1128, 2003.

[42] Márton Karsai, Mikko Kivelä, Raj Kumar Pan, Kimmo Kaski, János Kertész, A-L Barabási, and Jari Saramäki. Small but slow world: How network topology and burstiness slow down spreading. *Physical Review E*, 83(2):025102, 2011.

[43] Judith S Kleinfeld. The small world problem. *Society*, 39(2):61–66, 2002.

[44] L.M.C Kockelkoren. Modelling epidemics in dynamically evolving networks. 2017. Internal Report.

[45] Dirk P Kroese, Thomas Taimre, and Zdravko I Botev. *Handbook of Monte Carlo Methods*. John Wiley & Sons, New York, 2011.

[46] Mark Landler. A filipino linked to 'love bug' talks about his license to hack. *New York Times*, October 2000.

[47] Audrey Lee and Ileana Streinu. Pebble game algorithms and sparse graphs. *Discrete Mathematics*, 308(8):1425–1437, 2008.

[48] David A Levin and Yuval Peres. *Markov chains and mixing times*, volume 107. American Mathematical Soc., 2017.

[49] Thomas Milton Liggett. *Interacting particle systems*, volume 276. Springer Science & Business Media, 2012. Page 162.

[50] Jun S. Liu. *Monte Carlo Strategies in Scientific Computing*. Springer, 2001.

[51] Hagai Meirovitch and Shelly Livne. Computer simulation of long polymers adsorbed on a surface. ii. critical behavior of a single self-avoiding walk. *The Journal of chemical physics*, 88(7):4507–4515, 1988.

[52] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.

[53] T Móri. On random trees. *Studia Scientiarum Mathematicarum Hungarica*, 39(1-2):143–155, 2002.

[54] Mark EJ Newman. The structure and function of complex networks. *SIAM review*, 45(2):167–256, 2003.

[55] Christopher R Palmer and J. Greg Steffan. Generating network topologies that obey power laws. In *IEEE Global Telecommunications Conference*, volume 1, pages 434–438, 2000.

[56] Nathalie Peyrard and Alain Franc. Cluster variation approximations for a contact process living on a graph. *Physica A: Statistical Mechanics and its Applications*, 358(2):575–592, 2005.

[57] Boris Pittel. Note on the heights of random recursive trees and random m-ary search trees. *Random Structures & Algorithms*, 5(2):337–347, 1994.

[58] Oliver Ratmann, Christophe Andrieu, Carsten Wiuf, and Sylvia Richardson. Model criticism based on likelihood-free inference, with an application to protein network evolution. *Proceedings of the National Academy of Sciences*, 106(26):10576–10581, 2009.

[59] Patrick Rebeschini, Ramon Van Handel, et al. Can local particle filters beat the curse of dimensionality? *The Annals of Applied Probability*, 25(5):2809–2866, 2015.

[60] Nick Reynaert, SC Van der Marck, DR Schaart, W Van der Zee, C Van Vliet-Vroegindeweij, M Tomsej, J Jansen, B Heijmen, MARC Coghe, and Carlos De Wagter. Monte carlo treatment planning for photon and electron beams. *Radiation Physics and Chemistry*, 76(4):643–686, 2007.

[61] Garry Robins, Pip Pattison, Yuval Kalish, and Dean Lusher. An introduction to exponential random graph (p*) models for social networks. *Social Networks*, 29(2):173–191, 2007.

[62] Reuven Y Rubinstein and Dirk P Kroese. *Simulation and the Monte Carlo Method*. John Wiley & Sons, New York, third edition, 2017.

[63] Walter Rudin. *Real and complex analysis*. Tata McGraw-Hill Education, 1987.

[64] Sebastian Schnettler. A structured overview of 50 years of small-world research. *Social Networks*, 31(3):165–178, 2009.

[65] Josep Sempau, Andreu Badal, and Lorenzo Brualla. A penelope-based system for the automated monte carlo simulation of clinacs and voxelized geometries—application to far-from-axis fields. *Medical physics*, 38(11):5887–5895, 2011.

[66] Simon J Sheather and Michael C Jones. A reliable data-based bandwidth selection method for kernel density estimation. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 683–690, 1991.

[67] Michael E Smoot, Keiichiro Ono, Johannes Ruscheinski, Peng-Liang Wang, and Trey Ideker. Cytoscape 2.8: new features for data integration and network visualization. *Bioinformatics*, 27(3):431–432, 2010.

[68] Tom AB Snijders. Markov Chain Monte Carlo estimation of exponential random graph models. *Journal of Social Structure*, 3(2):1–40, 2002.

[69] Tom AB Snijders, Philippa E Pattison, Garry L Robins, and Mark S Handcock. New specifications for exponential random graph models. *Sociological methodology*, 36(1):99–153, 2006.

[70] Mikael Sunnåker, Alberto Giovanni Busetto, Elina Numminen, Jukka Corander, Matthieu Foll, and Christophe Dessimoz. Approximate Bayesian Computation. *PLOS Computational Biology*, 9(1):1–10, 01 2013.

[71] Thomas Taimre, Daniel Gibbons, and Brendan Patch. Markovian random graph processes. Unpublished manuscript, July 2015.

[72] Jeffrey Travers and Stanley Milgram. An experimental study of the small world problem. *Sociometry*, pages 425–443, 1969.

[73] R. Vaisman, R. Salomone, and D. P. Kroese. Stratified Splitting for Efficient Monte Carlo Integration. *ArXiv e-prints*, January 2017.

[74] Remco Van Der Hofstad. *Random graphs and complex networks*, volume 2. 2018. February 13, 2018 version.

[75] H.H.C.M. van Wesel. Simulation of a random network contact process. Summer Research Report, Eindhoven University of Technology, Department of Mechanical Engineering, February 2016.

[76] Peng Wang, Garry Robins, Philipa Pattison, and Emmanuel Lazega. Exponential random graph models for multilevel networks. *Social Networks*, 2013.

[77] Stanley Wasserman and Philippa Pattison. Logit models and logistic regressions for social networks: I. An introduction to Markov graphs and p*. *Psychometrika*, 61(3):401–425, 1996.

[78] Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world' networks. *nature*, 393(6684):440–442, 1998.

[79] Bernard M Waxman. Routing of multipoint connections. *IEEE Journal on Selected Areas in Communications*, 6(9):1617–1622, 1988.

[80] Richard David Wilkinson. Approximate bayesian computation (abc) gives exact results under the assumption of model error. *Statistical Applications in Genetics and Molecular Biology*, 12(2):129–141, 2013.

[81] Jinfeng Zhang, Rong Chen, Chao Tang, and Jie Liang. Origin of scaling behavior of protein packing density: A sequential Monte Carlo study of compact long chain polymers. *The journal of chemical physics*, 118(13):6102–6109, 2003.