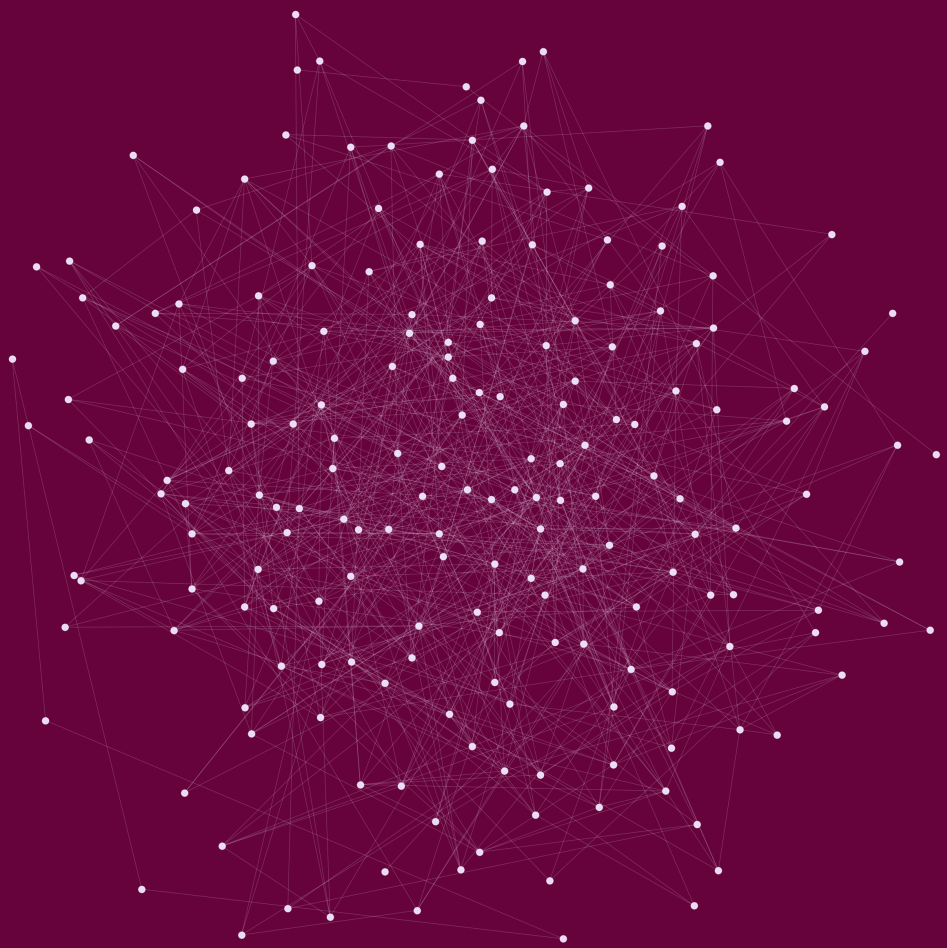
The Coloured Half-Edges Model

Importance Sampling for Large Deviations of the Giant Component in the Configuration Model

Julius Strack van Schijndel



TU Delft

The Coloured Half-Edges Model
Importance Sampling for Large Deviations of the Giant Component in the Configuration Model

Julius Strack van Schijndel
Student number: 5829909

Faculty of Applied Mathematics / EEMCS
Bachelor Thesis

Programme: BSc in Applied Mathematics
University: Delft University of Technology

Thesis committee: Dr. J. Komjáthy, DIAM, Advanced Probability
Dr. Ir. M. Keijzer, DIAM, Mathematical Physics

TU Delft Delft University of Technology

Date: June 2025

Non-expert summary

This thesis is about building a model that creates specific types of networks. A network, in this case, is simply a collection of elements (like people) and the connections between them (like friendships or communication). Mathematically, we call this a graph. There already exists a method called the *configuration model* that randomly builds such graphs while following certain rules—such as how many connections each person should have. When we use this model to make very large graphs, something interesting happens: a large, connected group of 'people' almost always forms. We call this the *giant component*, and it's a bit like the main cluster in a social network where everyone is connected, directly or indirectly. For a given setup, the giant component has a predictable–typical– size. It's extremely rare—-almost impossible–for it to be much smaller or larger than typically. But sometimes, we want to study what happens in those rare cases. This thesis explores how to build a new model that can generate these unusual graphs on purpose; graphs that still look natural, but where the giant component is either larger or smaller than the typical giant. This allows us to study networks in extreme scenarios that we normally can't access.

Contents

Chapter 1

Introduction

Networks have been an integral part of human history. From the time of hunters and gatherers, where complex social connections and thus networks were vital to survival [1], to modern times, where people use platforms like Instagram, LinkedIn, and WhatsApp to maintain personal and professional connections [2]. As mankind evolved and society got more complex, so did these social networks. For this reason it is important to understand properties of these (giant) networks [3].
Graphs are an important mathematical tool to model these social networks. Graphs that model real-world social networks, such as LinkedIn, can be useful for studying emergent properties, but if we want to determine whether certain properties emerge naturally from the degree distribution or are artifacts of structural bias, we have to make graphs at random and inspect their properties. A very useful tool to create network graphs with such a distribution in mind is the *configuration model* [4].

The *configuration model* is the setting of this thesis. More concise we will be looking at rare events of the model. When using the *configuration model* to generate large graphs, a very large component emerges: a giant. Molloy and Reed [5] published a paper on the density of the giant component. Their research focused on the emergence of the giant component. They showed that the giant undergoes a phase transition based on a (now) well known condition. For a given degree distribution $\{p_k\}$ a giant component will appear (for large enough n) only if:

$$\sum_{k=0}^{\infty} k(k-2)p_k > 0.$$

Moreover, they gave an explicit way of finding the density of the giant component. By solving a fixed point equation involving the generating function of $\{p_k\}$ the fraction of vertices that belong to the giant component can be found. Hence the typical size of the giant is inherently dictated by the degree distribution and not arbitrary.

But what if the giant does not realise this density? Bhamidi *et al.* have shown in their research that this is a rare event, showing that the probability of the giant deviating, quickly diminishes as the graphs grow larger [6]. This is known as the large deviation principle (LDP), which describes the exponential decay of probabilities for rare events as the system size grows. Much research has been conducted on this area of random graphs; Andreis *et al.* published explicit LDPs for component size distributions, talking about a full spectrum of component sizes [7], and later published similar results for inhomogeneous graphs [8]. Agazzi *et al.* researched the asymptotic behaviour of rare paths in the graph, which relate to the probability of observing atypical component configurations [9]. Jorritsma *et al.* recently (2024) published research about LDP in scale-free graphs [10].

Importance sampling is a method used to efficiently estimate the probabilities of rare events, such as large deviations, by simulating from an alternative distribution under which the rare event becomes typical [11]. In our context, this means deviating from the standard *configuration model* to sample graphs more likely to exhibit atypical giant components. Because of the exponential decay in probability of seeing a deviant giant, as documented by Bhamidi *et al.* standard Monte Carlo sampling is ineffective, making importance sampling a natural alternative. While a lot of research has been done regarding LDPs as aforementioned, the use of importance sampling to practically simulate and estimate these rare-event probabilities remains under-explored. Only a few works, by Guyader *et al.* [12] or by Dembo *et al.* for example [13], touch on general importance sampling frameworks within the large deviation setting, but tailored schemes for random graphs—particularly the *configuration model*—are limited.

The main goal of this thesis will be to realise importance sampling of a deviating giant component. After reviewing the basics regarding random graphs, the *configuration model*, and the research on large deviation principles with respect to the *configuration model*, we will explain in detail how we build a new model: Coloured half-edges model (CHEM).

We begin by detailing the structure and algorithmic construction of the CHEM model. Then we prove that this model realises the LDP stated by Bhamidi *et al.* and show simulation results of creating graphs using this model, along with validation of this model.

Chapter 2

Preliminaries

2.1 Graph Theory

Graph theory studies a mathematical structure, conveniently called a graph. Graphs are "simple" objects use to show networks. Graphs consist of nodes and edges, nodes describe elements while edges describe some connection between them. We show in Figure 2.1 an example of a graph. Graphs come in more forms to convey different and/or more information. For example the edges can be changed to have a direction, useful if one would like to show car routes or likewise. Let us introduce some definitions for graphs, which will be handy later on.

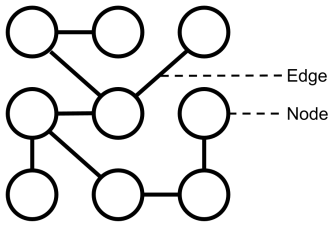


Figure 2.1: Example of a small connected graph.

Definition 2.1 (Graph)**.** A *graph* G is an ordered pair $G = (V, E)$, where:

- V is a finite set of *vertices* (or *nodes*),
- $E \subseteq \{\{u, v\} \mid u, v \in V, u \neq v\}$ is a set of unordered pairs of distinct vertices, called *edges*.

If $\{u, v\} \in E$, we say that u and v are *adjacent*, or that they are *connected by an edge*.

Remark 2.1.1. Definition 2.1 is about *undirected*, *simple* graphs.
Graphs are called *simple* when they contain no loops (edges from itself to itself), and no multi-edges (multiple edges with the same start and end node). Furthermore, graphs are called *directed* when E is an ordered pair.

Definition 2.2 (Components)**.** Let $G = (V, E)$ be an undirected graph. A *connected component* (or simply a *component*) of G is a maximal subset of vertices $C \subseteq V$ such that for every pair of vertices $u, v \in C$, there exists a path in G connecting u and v.

In other words, C is a connected subgraph of G, and no vertex in $V \setminus C$ is connected to any vertex in C.

Remark 2.1.2. The term 'maximal' in definition 2.1 means that you cannot add any other *vertex* to the *component* without expanding the subgraph.

A specific set of graphs we are interested in is random graphs.

2.1.1 Random Graphs

The field of random graphs is considered to be at the intersection of graph theory and probability. The goal is to study properties of models coming from distributions (probability measures) on the set of graphs (simple or not).

Definition 2.3 (Random graph)**.** A *random graph* is defined via a *probability distribution* over a *set* of possible *graphs*.

Remark 2.1.3. More concretely, one can write that a *random graph* is defined by a measure (or sequence of measures) on the set of graphs. Where G_n denotes the realization of this measure (sequence). Here n indicates the number of vertices.

This topic was introduced by Hungarian mathematicians Paul Erdős and Alfred Rényi[14]– and at the same time independently by American mathematician Edgar Gilbert[15]. They propose a graph G which depends on n nodes and a probability p, this graph is generally known as an Erdős–Rényi graph.

Definition 2.4 (Erdős–Rényi Graph)**.** An *Erdős–Rényi graph* $G(n, p)$ is a random undirected graph with $n \in \mathbb{N}$ vertices, where each of the $\binom{n}{2}$ possible edges between distinct pairs of vertices is included independently with probability $p \in [0, 1]$. Formally, the vertex set is $V = \{1, 2, \ldots, n\}$, and for each pair $\{i, j\}$ with $i < j$, the edge $\{i, j\}$ is present in the edge set E with probability p, independently of all other pairs.

Remark 2.1.4. The expected amount of edges in the graph is $\mathbb{E}[E]$, so the average edge density is given by $\frac{1}{n}\mathbb{E}[E]$. By the law of large numbers (and i.i.d. distribution), this converges exactly to p as $n \to \infty$.

Remark 2.1.5. There is a different way to define Erdős–Rényi graphs, where instead of a probability p you input a total number of edges M such that $G(n, M)$ has exactly M edges. This is done by uniformly choosing a subset of M edges, where a subset is chosen with probability $\binom{n(n-1)/2}{M}^{-1}$.

Another important concept regarding random graphs is the degree distribution of the graph.

Definition 2.5 (Empirical Degree distribution)**.** Let $G = (V, E)$ be a graph. The *degree distribution* of G_n is the function $p_k(n)$ that gives the probability that a randomly chosen vertex has degree k, that is,

$$p_k(n) = \frac{|\{v \in V : \deg(v) = k\}|}{|V|}, \quad k \in \mathbb{N}_0.$$

In other words, $p_k(n)$ is the fraction of vertices in G_n that have degree k. The collection $\{p_k\}_{k\in\mathbb{N}_0}$ defines the empirical degree distribution of the graph.

2.1.2 Configuration model

Instead of the aforementioned method to make random graphs, based on a certain probability of connections forming, there is a different way to make random graphs. The configuration model is a way to make random graphs based on a given degree-distribution.

Configuration model (CM) random graphs are commonly defined as the realisation of a degree sequence.

Definition 2.6 (Configuration model random graph)**.** Let $n \in \mathbb{N}$ and let $\vec{d} = (d_1, \dots, d_n)$ be a degree sequence with $\sum_{i=1}^n d_i$ even. The *configuration model* $\mathrm{CM}_n(\vec{d})$ is the random (multi-)graph constructed as follows:

- To each vertex $i \in \{1, \dots, n\}$ attach d_i *half-edges*.
- Take a *uniform random perfect matching* of the $\frac{1}{2}\sum_i d_i$ unordered pairs of half-edges.
- For each matched pair of half-edges, join their incident vertices by an edge (allowing loops and multiple edges).

The resulting random multigraph on n vertices is denoted $CM_n(\vec{d})$.

In this thesis we are mostly interested in the properties of the degree sequence $\vec{d}$, namely the proportion of each degree. Thus we define the (limit) degree distribution.

Definition 2.7 (Degree distribution)**.** Let $n \in \mathbb{N}$ and let $\vec{d} = (d_1, \dots, d_n)$ be a degree sequence with $S = \sum_{i=1}^n d_i$ even. The induced degree distribution $\vec{p}(n) = \{p_k(n)\}_{k=0}^\infty$ has the following relation to $\vec{d}$.

$$\forall k \; p_k(n) = \tfrac{1}{S} \sum_{i=1}^n \mathbb{1}_{d_i=k}.$$

Furthermore, we consider the limiting degree distribution $\vec{p} = \{\bar{p}_k\}_{k=0}^\infty$. Where we say that for any limiting degree distribution $\vec{p}$ there is a degree sequence $\vec{d}$ such that the induced degree distribution $\vec{p}$, converges to $\vec{p}$ as $n \to \infty$.

Throughout this thesis we will consider graphs G_n with (limiting) degree distribution $\vec{p}$, since the specific degree sequence is of little importance. Moreover, we define some regularity conditions for configuration model graphs.

Definition 2.8 (Regularity assumptions)**.** For degree sequences $\vec{d} = \{d_1, \cdots, d_n\}$ we assume it to be a realisation of the random variable D_n with distribution function F_n. We assume the following regularity conditions:

I Weak convergence
There exists a distribution function F such that
$$D_n \xrightarrow{d} D,$$
where D has distribution function F.

II First moment convergence
$$\lim_{n\to\infty} \mathbb{E}[D_n] = \mathbb{E}[D].$$

III Second moment convergence
$$\lim_{n\to\infty} \mathbb{E}[D_n^2] = \mathbb{E}[D^2].$$

Remark 2.1.6. Configuration model random graphs depend on a degree sequence as described in definition 2.6, but instead of given the degree sequence $\vec{d}$ we will often just talk about the degree distribution D_n or D.

2.2 Branching Processes

Branching processes have an inherent relation to graphs. Giant components (which will be discussed in depth later) can be described by a branching process. There is a simple way to relate the two concepts, but let us first get more comfortable with branching processes.

Definition 2.9 (Branching process (Galton–Watson process))**.** Let Z_i be the amount of offspring at time step i, and let X_i be i.i.d.[1] discrete[2] random variables with distribution $\{p_k\}$ such that $\mathbb{P}(X_i = k) = p_k$ and with k the amount of offspring. A branching process starts with an ancestor, write $Z_0 = 1$. Then

$$Z_{i+1} = \sum_{i=0}^{Z_i} X_i.$$

For reasons which will become apparent in the rest of the thesis, we would rather look at a *random tree* created by a branching process, called a *Galton-Watson tree* (See Fig. 2.2).

Definition 2.10 (Galton-Watson tree)**.** Let $\{X_i\}_{i\geq 1}$ be a sequence of i.i.d. random variables such that $\mathbb{P}(X_i = k) = p_k$ for $k = 0, 1, 2, \cdots$. A Galton-Watson rooted tree $\mathscr{T}$ is constructed in the following way:

I Start with the root vertex v_0.

II The root has X_1 *offspring* connected to it by edges. Label these *offspring* arbitrarily.

III Each *offspring* independently has offspring of their own, according to independent copies of X.
Thus, if a node has label v, its number *offspring* is an i.i.d. copy X_v of X, and they are connected to v with an edge.

IV Repeat this process recursively for all newly created *offspring*.

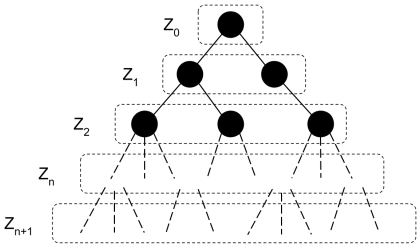


Figure 2.2: Example of a branching tree.

Remark 2.2.1. For Galton-Watson (rooted) trees, the tree has positive probability of being infinite if $\mathbb{E}[X] > 1$, otherwise the tree dies out almost surely.

To relate these Galton-Watson trees to (random) graphs, we take a look at the local weak limit. We first introduce the notion of rooted graph isomorphisms as they are an essential concept in the theorem on the local weak limit.

Definition 2.11 (Rooted graph-isomorphism.)**.** Let T_1 and T_2 be rooted graphs with root ρ_1, ρ_2 respectively. G_1 and G_2 are root isomorphic if there exists an isomorphism $\phi : T_1 \to T_2$ such that $\phi(\rho_1) = \rho_2$.

Remark 2.2.2. An isomorphism maps from $G \to F$ by mapping edges to edges and non-edges to non-edges. Furthermore, we denote *rooted isomorphic* with the symbol $\cong$.

Theorem 2.1 (Local weak limit (Benjamini–Schramm convergence) [16])**.** *Let $B_R(G_n, v_n)$ be a ball of radius R of a uniformly chosen vertex v_n in G_n (w.r.t. graph-distance) and let T_n be a branching tree with root offspring distribution D and with all other vertices having offspring distribution $\tilde{D}$.*
If G_n has degree distribution D with finite second moment, then:

$$\mathbb{P}(B_R(G_n, v_n) \cong H(R, \rho)) \to \mathbb{P}(B_n(T_n, v_n) \cong H(R, \rho)) \text{ as } n \to \infty \text{ for any finite rooted graph } H(R, \rho). \tag{2.1}$$

Where $\tilde{D}$ is such that $\mathbb{P}(\tilde{D} = k - 1) = kp_k/\mathbb{E}[D]$

A nice proof of this theorem is provided by Van der Hofstad [17].

[1] Independent, Identically, Distributed, thus X_i do not influence each other and they follow the same distribution.
[2] X will only take integer values.

2.3 Giant Component

When creating random graphs, Erdős–Rényi graphs, configuration model graphs, and other random-graphs, large connected components emergence as the number of vertices grows larger. Specifically *giant components*, which are very large parts of the total graph. Let us give a proper definition of these *giant components*.

Definition 2.12 (Giant component)**.** Let G_n be a graph sequence, and let $\mathcal{C}_n^{(1)}, \mathcal{C}_n^{(2)}, \mathcal{C}_n^{(3)}$ be the largest/second largest/third largest component. We call them a giant component if, $|\mathcal{C}^{(i)}(n)|/n$ does not converge to zero in probability as $n \to \infty$.

Remark 2.3.1. Notation
$\mathcal{C}_n^{(k)}$ denotes the k$^{\text{th}}$ giant component of a graph according to size, where $\mathcal{C}_n^{(1)}$ is the largest. Furthermore, we denote by $|\mathcal{C}|$ the number of vertices in a component $\mathcal{C}$.

Figure 2.3 shows a sketch of a giant component. The smaller components are coloured grey, and we can see that the giant contains 11 of the 16 nodes. When these giant components do or do not appear is a very interesting question, one that Erdős also asked [14].

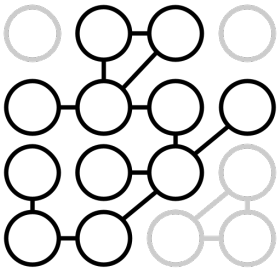


Figure 2.3: Example of a small connected graph.

Theorem 2.2. *Let $G(n,p)$ be an* Erdős–Rényi graph *with $n \in \mathbb{N}$. If $p > \frac{1}{n}$, then there is a single giant component with high probability. If $p < \frac{1}{n}$, there is no giant component, with high probability.*

Erdős proves this theorem in his original paper.

Definition 2.13 (With high probability)**.** Let $\{\gamma_n\}_{n\in\mathbb{N}}$ be a sequence of events. We say that γ_n holds with high probability if,

$$\lim_{n\to\infty} \mathbb{P}(\gamma_n) = 1.$$

For the configuration model the question on the appearance of the giant component was answered by Molloy [5], who found a specific criterion. Before we give this criterion, we define the degree proportion of a graph, which makes the criterion much easier to read.

Definition 2.14 (Degree distribution)**.** Let G_n be a graph and let $\rho_k(n)$ denote the proportion of degree k vertices in G. Call D_G the distribution of degrees in G_n, and define it as the random variable (r.v.) representing the degree of a uniformly randomly chosen node in G such that $\mathbb{P}(D_G = k) = \rho_k(n)$

Theorem 2.3 (Molloy Criterion)**.** *[5]*
Let $G_n = \mathrm{CM}_n(\vec{d})$ be an instance of the configuration model with degree distribution D_G that satisfies the regularity assumptions given in Definition 2.8, and let Q be as follows:

$$Q(D_G) := \sum_{i=1}^{\infty} k(k-2)p_k = \mathbb{E}[D_G(D_G - 2)]. \tag{2.2}$$

If $Q(D_G) > 0$ a unique giant component emerges. While if $Q(D_G) < 0$ there will not be any giant component.[3]

Remark 2.3.2. The proof of Theorem 2.3 is given by Molloy and Reed [5]. The uniqueness stated comes from the fact that, with high probability, we have

$$\frac{|\mathcal{C}_n^{(1)}|}{n} \nrightarrow 0 \text{ in probability} \quad \frac{|\mathcal{C}_n^{(2)}|}{n} \to 0 \text{ in probability.} \tag{2.3}$$

2.3.1 Giants Proportion

Further important properties of giant components is their relative size in a graph. Let us introduce a few more definitions to get familiar with this.

Definition 2.15 (Giant component density)**.** Let G_n be a graph[4] with n vertices and let $\mathcal{C}_n^{(1)}$ be the largest giant component of G_n. Then, if the first and second moment of the related degree distribution are finite, we have:

$$\frac{|\mathcal{C}_n^{(1)}|}{n} \xrightarrow{p} \theta.$$

For some constant $\theta \in \mathbb{R}$.

[3] The case where Q(D) = 0 is the critical case, which is beyond the scope of this thesis.
[4] Can also be an instance of the configuration model

Now we use the nice approximation that locally the giant is like a Galton–Watson branching process (Thm. 2.1), to further look at these proportions. First, we introduce the size-biased distribution. If we have a graph G with degree distribution D_G as in Definition 2.5 we have a corresponding size-biased distribution: $\tilde{D}_G$. The probability is given by

$$\mathbb{P}(\tilde{D}_G = k) = \frac{k\mathbb{P}(D_G = k)}{\mathbb{E}[D_G]}$$

We connect this to the branching process by imagining a graph G and picking a random edge in G and choosing one of its end-nodes v. Then the distribution of the degree of v is a size-biased version of the degree distribution of G. Let $Z_D = \tilde{D} - 1$ so,

$$\mathbb{P}(Z_{D_G} = k) = \mathbb{P}(\tilde{D}_G = k+1) = \frac{(k+1)\mathbb{P}(D_G = k+1)}{\mathbb{E}[D_G]}. \tag{2.4}$$

This Z_{D_G} corresponds to the amount of edges you will get to when you follow an edge to its end-node [4].
Before we continue to connect this to the giant component, we have to introduce the *extinction rate* of a branching process.

Theorem 2.4 (Extinction rate [18])**.** *Consider a branching process with offspring distribution $\{p_k\}_{k\geq 0}$ and probability generating function $g(s) = \sum_{k=0}^{\infty} p_k s^k$. The extinction probability x is then the smallest solution (on $[0,1]$) to the fixed point equation:*

$$x = \sum_{k=0}^{\infty} p_k x^k.$$

Remco van der Hofstad has a very detailed proof about this theorem [19].

For the giant component, we will now imagine we are exploring a graph G. Pick a random node in G, then explore the neighbours of this node, the amount of 'new' nodes we discover is $\tilde{D} - 1$[5] $= Z_{D_G}$ distributed. We keep repeating this process until we run out of nodes. If we do not run out of nodes, we find ourselves in the giant component. The probability that this process never dies out is exactly $1 - x$, where x is the ultimate extinction rate (Def 2.4). Now, for a random node, we can determine the probability that all its edges lead to finite components as follows: a random node has k neighbours, the probability that they all lead to dead ends is x^k, so we take a weighted average over all degrees in G. Hence the probability of a random node being in a finite component is $g(x) = \sum_{k\geq 0} p_k x^k$ [20][19]. Thus we conclude that the giant's proportion is $1 - g(x)$.

This is not the full story, while the conclusion is true, we actually need a condition for it to hold. To this end we present here the main result of Molloy and Reed [5][21].

Theorem 2.5. *Let $\mathrm{CM}_n(\vec{d})$ be an instance of the configuration model with degree distribution $D_n \to D$ and size-biased distribution $\tilde{D}_n \to \tilde{D}$. Furthermore, let G_D be the generating function of D. If $\mathbb{E}[D(D-2)] > 0$ then there exists a unique solution $x \in (0,1)$ to the fixed point equation $s = G_{Z_D}(s)$ and,*

$$\frac{|\mathcal{C}_n^{(1)}|}{n} \xrightarrow{p} 1 - G_D(x) =: \theta > 0 \text{ as } n \to \infty.$$

The important takeaway here is that the giant has a typical density within the graph: this θ. But what if the giant is bigger than θ in the limit? For Erdős–Rényi graphs it is clear when the giant gets really big, so big it is the entire graph in fact. Erdős proved the following theorem in his original paper [14].

Theorem 2.6. *Let G be an Erdős–Rényi graph and let $\mathcal{C}_n^{(1)}$ be the largest giant component.*

$$\text{If } p > \frac{\ln(n)}{n} \text{ then } \lim_{n\to\infty} \frac{|\mathcal{C}_n|}{n} = 1 \text{ a.s.}$$

For the configuration model, no such theorem exists, unfortunately. There is however research done on *large deviations* of the configuration model.

2.3.2 Large deviations

The probability that the giant component of an instance of the configuration model is larger than θ is exponentially decreasing. [19] Thus a giant greater than θ is a rare event, also known as a large deviation. Essential to large deviation theory is Cramér's theorem, which determines the rate function for a series of i.i.d events.

[5] This -1 comes from the fact that we 'entered' the node via one edge, thus one is already discovered.

Theorem 2.7 (Cramér's theorem)**.** *[22]*
Consider a series of i.i.d. variables X_i. Assume that there is an open interval containing 0 where the moment-generating function $M(t) = \mathbb{E}[e^{tX_1}]$ is finite. Define $\Lambda(t)$ as follows

$$\Lambda(t) := \ln(M(t)), \quad \Lambda^*(t) := \sup_{t\in\mathbb{R}}\{tx - \Lambda(t)\}, \tag{2.5}$$

Then the sequence X_i satisfies the large deviation principle on $\mathbb{R}$ with speed n and good [6] rate function Λ^*. Equivalently

For every closed subset F of $\mathbb{R}$

$$\limsup_{n\to\infty} \frac{1}{n} \ln \mathbb{P}(\bar{X}_n \in F) \leq - \inf_{x\in F} \Lambda^*(x). \tag{2.6}$$

For every open subset G of $\mathbb{R}$

$$\limsup_{n\to\infty} \frac{1}{n} \ln \mathbb{P}(\bar{X}_n \in G) \geq - \inf_{x\in G} \Lambda^*(x). \tag{2.7}$$

In particular, for any $a > \mathbb{E}[X_1]$

$$\mathbb{P}(\bar{X}_n \geq a) \approx \exp(-n\Lambda^*(a)) \;\; n \to \infty. \tag{2.8}$$

A very nice proof of this theorem is given by Dembo *et al.* [23].

Remark 2.3.3. An intuitive way to interpret the equations in Theorem 2.7 is as follows. In any case, (2.6) tells us that the function cannot decay slower than the worst rate in F. While (2.7) says the converse: the rate function cannot decay faster than the infimum. Lastly, (2.8) tells us that the probability of overshooting (or undershooting) the expected value goes to 0 exponentially fast.

In our case, the sequence of i.i.d. variables, are the graphs. Each CM-graph is independent of each other, and they are created equally. Bhamidi *et al.* [6] published results which are closely related to the rate function of the configuration model.
Let us give some context and relevant notation to better understand those results. Starting with the special event that we are interested in. In a typical CM random graph, the giant component is unique and of a certain proportion as stated before (Thm. 2.5). Let us define an event, where the giant component is larger than it typically is.

Definition 2.16 (Large giant)**.** Consider a graph G_n with n vertices. Let G_n have degree distribution which converges to $\vec{p} = \{p_k\}_{k\in\mathbb{N}}$. Then for every vector $\vec{q} = \{q_k\}_{k\in\mathbb{N}}$ such that $0 \leq q_k \leq p_k$ for all k, we consider the event

$$E^{n,\varepsilon}(\vec{q}) = \{\text{There exist a component in } G_n \text{ with } m_k \text{ degree } k \text{ vertices where } m_k \in [n(q_k - \varepsilon, n(q_k + \varepsilon)], k \in \mathbb{N}\} \tag{2.9}$$

The probability of (2.9) happening will be denoted by $\phi_q^{n,\varepsilon}$. Bhamidi *et al.* provided a theorem on the lower and upper bound of $\phi_q^{n,\varepsilon}$.

Theorem 2.8 (Bhamidi lower bound)**.** *[6]*
Let G_n be a configuration model random graph with degree distribution that converges to $\vec{p}$ and let $0 \leq \vec{q} \leq \vec{p}$ then

$$\liminf_{\varepsilon\to 0} \liminf_{n\to\infty} \ln(\phi_q^{n,\varepsilon}) \geq -nI(\vec{p}, \vec{q}) \tag{2.10}$$

The proof of this theorem is given as the main result in the paper by Bhamidi *et al.*[6].

Remark 2.3.4. The upper-bound of $\mathbb{P}(\phi_q^{n,\varepsilon})$ is given by the $\limsup_{\varepsilon\to 0} \limsup_{n\to\infty}$ and is less or equal to $-nI(\vec{p}, \vec{q})$
The function $I(\vec{p}, \vec{q})$ is the rate (or entropy) function: see Cramér's theorem 2.7.

2.3.3 Rate function

In this section we will closely examine this rate function, which appears in (2.10). The function has the following form.

$$I(\vec{p}, \vec{q}) = H(\vec{q}) + H(\vec{p} - \vec{q}) - H(\vec{p}) + K(\vec{q}) \tag{2.11}$$

The H function is closely related to the graph exploration we discussed in Section 2.3.1. Imagine doing multiple exploration walks in the graph: you choose a random node to start and explore until you have uncovered the entire component, then pick a new node which you have not explored yet and repeat. Eventually, you will have discovered the entire graph.

[6] *Good* refers to the fact that the level-sets of the rate function are compact.

For each component you explored, you encountered a certain configuration of offspring degrees. Since we are essentially forcing the graph to display the event $E^{n,\varepsilon}(\vec{q})$ we take away from the freedom. The function H describes the 'cost' of limiting this freedom. The first part of H is the cost of seeing nq_k degree k degrees in those components, which is the sum of $q_k \ln q_k$ over all k. When distributing the degrees across all nodes, we are essentially assigning half-edges. Thus, after handing out the half-edges, one still has to decide how to connect all these half-edges into actual edges. The average degree of a node is $\frac{1}{2}\sum_{k\geq 1} kq_k$, and then there are $\ln(\frac{1}{2}\sum_{k\geq 1} kq_k)$ many ways of connecting the initial half-edges. These many ways of connecting can be seen as a "bonus" since we again allow for some freedom, thus this part will decrease the cost a bit.

So we obtain the following expression for H:

$$H(\vec{v}) = \sum_{k=1}^{\infty} kv_k \ln(kv_k) - \frac{1}{2}\sum_{k=1}^{\infty} kv_k \ln\left(\frac{1}{2}\sum_{k=1}^{\infty} kv_k\right) \tag{2.12}$$

When substituting in $\vec{q}$ into H we get the cost of forcing the graph to have q_k degree k vertices. While plugging in $\vec{p}$ into H, obtains the cost of just getting a configuration model random graph with degree distribution $\vec{p}$. There is a middle ground between these two distributions, which comes from the fact that we could have made a configuration model random graph with limit degree distribution $\vec{p} - \vec{q}$.

The H functions in the rate function I in the following way. As can be gathered from Cramér's theorem 2.7 the rate function comes from taking the log of the probability that the event $E^{n,\varepsilon}(\vec{q})$ and dividing by n. That probability is given by the following equation:

$$\mathbb{P}(E^{n,\varepsilon}(\vec{q})) = \prod_{k=1}^{\infty} \frac{np_k!}{(nq_k)!(n(p_k-q_k))!} \cdot \frac{(n\sum_k kq_k)!!(n\sum_k k(p_k-q_k))!!}{(n\sum kp_k)!!} \cdot \mathbb{P}(\text{The 'inside' vertices form a single component}). \tag{2.13}$$

The first part of (2.13) counts the ways of prescribing which degree k vertices are or are not inside the giant component. The second part is the essentially probability that the vertices which were assigned 'inside' and the vertices which where assigned 'outside' form *separate* matchings. Thus they do not connect together somewhere. When taking the log of this probability and dividing by n, the three H functions appear.

Before we continue to the K function, we first talk about typicality. We are trying to obtain the special event $E^{n,\varepsilon}(\vec{q})$, where this $\vec{q}$ is a desired degree distribution. When this degree distribution is implemented, we essentially have to *tilt* the configuration model, in such a way that seeing degree distribution $\vec{q}$ is typical. If we tilt each q_k to $q_k\beta^k$ for a specific parameter β our degree proportion becomes typical in the new model. Finding this tilt β can be done by forcing the tilted profile to be critical. Solving (2.14) yields the tilt parameter β.

$$\sum_{k=1}^{\infty} kq_k = (1-\beta^2)\sum_{k=1}^{\infty} \frac{kq_k}{1-\beta^k}. \tag{2.14}$$

Lemma 2.1. *The parameter β that solves (2.14) uniquely exists if $\mathbb{E}[D] > 2$.*

Proof. Let us start by defining $f(\beta)$:

$$f(\beta) := \sum_k kq_k - (1-\beta^2)\sum_{k=1}^{\infty} \frac{kq_k}{1-\beta^k} = \beta\left(\sum_{k=3}^{\infty} \frac{\beta-\beta^{k-1}}{1-\beta^k} kq_k - q_1\right).$$

We are looking for a $\beta \in (0,1)$ for which $f(\beta) = 0$. The term $(\beta - \beta^{k-1}/(1-\beta^k)$ we define as $f_k(\beta)$. From the definition of $f_k(\beta)$ it is clear that it is strictly increasing on $(0,1)$.
Furthermore, we see $\lim_{\beta\to 0^+} f_k(\beta) = 0$ and $\lim_{\beta\to 1^-} f(\beta) \overset{l'Hôpital}{=} \lim_{\beta\to 1^-} \frac{1-(k-1)\beta^{k-2}}{-k\beta^{k-1}} = \frac{k-2}{k}$. Thus we see

$$-q_1 = f(0^+) < f(\beta) < f(1^-) = \sum_{k=3}^{\infty}(k-2)q_k - q_1. \tag{2.15}$$

Since f is continuous on $(1,0)$ we use the intermediate value theorem [24] to obtain that there exists a unique value β such that $f(\beta) = 0$, which concludes the proof. □

Remark 2.3.5. The right hand side in (2.15) is exactly $\sum_{k\geq 1}(k-2)q_k$, which must be positive since $\sum_{k\geq 1}(k-2)q_k = \mathbb{E}[D-2] > 0 \iff \mathbb{E}[D] > 2$ as stated in the lemma.

The function K hence comes from the last part of (2.13). This function can be related to a "tilt" for which the branching process becomes typical. To express the function K we star by tilting some of the costs of H. First the term $\frac{1}{2}\sum_{k\geq 1} kq_k$ which is the amount of half-edges to match up, as seen before. Now the weight of each half-edges is altered under the critical tilt, and we take the natural log to account for the amount of ways we can combine these half-edges under this tilt. So the term becomes $(\frac{1}{2}\sum_{k\geq 1} kq_k)\ln(1-\beta^2)$. Furthermore, K also has a cost build in for forcing some degrees to be outside the giant. Each vertex with degree k lives outside the giant if all of its half-edges lead to extinction nodes. The probability to die out, i.e. eventually connect to the giant and "kill" the finite component, is β. The chance that this happens is β^k and equivalently the chance this does not happen is $1-\beta^k$. Thus we add the cost for each q_k giving the term $\sum_{k\geq 1} q_k \ln(1-\beta^k)$. Equation 2.16 gives the formula for $K(\vec{q})$.

$$K(\vec{q}) = \left(\frac{1}{2}\sum_{k\geq 1} kq_k\right)\ln(1-\beta^2) - \sum_{k\geq 1} q_k \ln(1-\beta^k). \tag{2.16}$$

Now let us look at the rate function again and summarize what each part of the equation contributes in (2.17).

$$\underbrace{I(\vec{p},\vec{q})}_{\substack{\text{Cost of forcing}\\ E^{n,\varepsilon}(\vec{q})}} = \underbrace{H(\vec{q})}_{\substack{\text{Cost to force}\\ \text{giants degrees}}} + \underbrace{H(\vec{p}-\vec{q})}_{\substack{\text{Cost for finite}\\ \text{components}}} - \underbrace{H(\vec{p})}_{\substack{\text{Profit from}\\ \text{baseline}}} + \underbrace{K(\vec{q})}_{\substack{\text{Cost of forcing}\\ \text{a single component}}}. \tag{2.17}$$

Now that we have our expression for this rate function I in equation 2.10 from theorem 2.8, we can continue with its implication. The implication is rather easily said; the event $E^{n,\varepsilon}(\vec{q})$ is extremely rare. If one would like to research graphs with the event $E^{n,\varepsilon}(\vec{q})$, they would have to simulate a configuration model random graph, e^{nI} amount of times on average to have $E^{n,\varepsilon}(\vec{q})$ happen once. Since this is ridiculous (without quantum computing), we would like a better and faster way to simulate this. That is where the theory of importance sampling comes in.

2.4 Importance Sampling

Importance sampling is a way of evaluating properties of a certain distribution, by repeated random sampling[7] of a different distribution. As discussed in section 2.3.3 repeated random sampling is futile for an exponentially decaying probability. Thus one would like to sample from a different distribution, where a special event turns up with much higher probability, specifically with probability 1 almost surely.

In essence, we are looking for a way of building configuration model random graphs with the event $E^{n,\varepsilon}(\vec{q})$ always happening, while still keeping the process random. This is the main goal, to give a way of creating a configuration model random graph with the event $E^{n,\varepsilon}(\vec{q})$ happening almost surely.

[7] i.e. Monte Carlo method

Chapter 3

Coloured half-edges model

In this chapter we will design an augmented configuration model. This model, which we call the coloured half-edges model (CHEM for short) depends on a degree distribution (Def. 2.5) and a vector of desired giant component degree proportions. Furthermore, this model should achieve importance sampling for the rare event $E^{n,\varepsilon}(\vec{q})$ (2.9). The augmentation is a change in degree distribution, and a different matching of half-edges.

We will build the model in four steps. The first two steps focus on the degree distribution; the first section will explain how to manipulate an original degree distribution. The second section then complements this by building the 'bridge' between the distribution and the matching.
Section three and four explain the matching method and tweaking the matching to obtain a valid graph.
The final section of this chapter will give a step-by-step algorithm to make a CHEM graph.

3.1 Labelling

The first step of creating a coloured half-edges model random graph (CHEM-graph), is to create n nodes and label them. Before we can investigate the labelling, we take a step back and review some standard parameters. The giants size θ introduced in definition 2.15 is related to the event $E^{n,\varepsilon}(\vec{q})$ by this $\vec{q}$. They share the following simple relation shown in (3.1), for the typical degree proportions q_k^T.

$$\sum_{k=1}^{\infty} q_k^T = \theta \tag{3.1}$$

This means that increasing one or more q_k^T will lead to a giant component that is larger than θ. The challenge at hand is finding the typical degree proportions (q_k^T) and increasing them. In a configuration model random graph with n nodes, the giant component has approximately $n \cdot q_k^T$ degree k vertices.
This relates back to (3.1), the giant component consist of $n \cdot \sum q_k^T$ vertices, so the proportions in the giant component are clearly given by q_k^T.
Let us remark that the relation between q_k^T and p_k is given by the extinction rate z of the giant component (Z_{D_G}). In the limit the proportion of degree k vertices in the giant, is the probability of a degree k vertex multiplied by the probability of not all edges "dying". So we obtain the relation noted in (3.2).

$$q_k^T = p_k(1 - z^k) \tag{3.2}$$

Thus we only need to solve the fixed point equation $G_{Z_{D_G}}(z) = z$ to obtain the values of q_k^T. If we then select our own proportions q_k (often shortened to q_k as q_k^T represents the typical values), to be such that $\sum q_k > \theta$, we obtain a deviant giant. For this, we introduce the following condition:

$$\forall k \geq 0, \forall q_k \in \vec{q}, p_k \in \vec{p} : \; q_k^T \leq q_k \leq p_k. \tag{3.3}$$

When we talk about $\text{CHEM}_n(\vec{p}, \vec{q})$ graphs, $\vec{p}$ refers to some degree distribution (Def. 2.5) and some $\vec{q}$ that satisfies condition (3.3).

The next step is to make sure our q_k also make the giant a branching procedure. For this we solve (3.4) for x, which is comparable to the β from (2.14).

$$\sum_{k=1}^{n} k q_k = (1 - x^2) \sum_{k=1}^{n} \frac{k q_k}{1 - x^k}. \tag{3.4}$$

With this x, we relate q_k to p_k with an extra (new) normalization parameter based on x.

$$q_k = \frac{p_k(1-x^k)}{\gamma_k(x)} \iff \gamma_k(x) = \frac{p_k(1-x^k)}{q_k}. \tag{3.5}$$

Since p_k, q_k, and x are known, we can solve for $\gamma_k(x)$. This normalization parameter is crucial for the labelling. For now, we will continue with the next step, as the parameter γ will come back later.
The proportions of the giant component are now known, thus we can make a giant with said proportions. The next step is to introduce the labelling.
For a graph G_n of n nodes, we define the label $m_n(i,j)$ which denotes the amount of nodes that have i extinction half-edges, and j survival half-edges. Two conditions that must hold to build a graph of size n and degree proportion $\rho_k(n)$ are given below in equations 3.6 and 3.7.

$$n = \sum_{i=0}^{\infty}\sum_{j=0}^{\infty} m_n(i,j). \tag{3.6}$$

$$\rho_k(n) = \sum_{j=0}^{\infty} \frac{m_n(k-j,j)}{n}, \rho_k(n) \overset{n\to\infty}{\to} \rho_k \tag{3.7}$$

Next, we define the variable $\rho(i,j)$ which symbolizes the limiting proportion of vertices with i extinction half-edges and j survival half-edges. The conditions for this variable are shown in (3.8) and (3.9).

$$\sum_{i=0}^{\infty}\sum_{j=0}^{\infty} \rho(i,j) = 1. \tag{3.8}$$

$$\frac{m_n(k-j,j)}{n} \to \rho(k-j,j) \text{ as } n \to \infty. \tag{3.9}$$

Later on, these labels will be what we base our colouring on. For now, we will explore how to label the nodes. First, we remark that we mainly use $m_n(k-j,j)$ and $\rho(k-j,j)$. Secondly, note that the ρ_k should be equal to q_k. From there we calculate how large the proportions of different j values should be for the corresponding $m_n(k-j,j)$ labels. For every $m_n(k-j,j)$ there are $k+1$ ways of filling the brackets. For every j there are $k-j$ extinction edges and j survival edges, so for each j there is a total of $\binom{k}{j}$ ways to make such a node. This needs to be taken into account, so each $\rho(k-j,j)$ is calculated by taking this into account, along with the parameter x and $\gamma_k(x)$. Equation 3.10 shows the general formula for $\rho(k-j,j)$.

$$\rho(k-j,j) = \frac{p_k\binom{k}{j}x^k(1-x^j)}{\gamma_k(x)}. \tag{3.10}$$

In this way, we have assured that $\sum \rho(k-j,j) = q_k$, so when we multiply by n we find the amount of $m_n(k-j,j)$ labels we assign for every k and j.

To connect all these terms and formulae, and build some intuition, let us build a graph together. To illustrate how to compute x and $\rho(i,j)$ in the special case where the graph only has nodes of degree one, two, or three.

Example 3.1. Suppose we build a graph. There is some initial $\vec{p} = (p_1, p_2, p_3)$ degree distribution known to us. Step one is to find the typical proportion of degrees in the giant component, thus we need to find z.

$$z = G_{Z_{D_G}}(z) = z\frac{p_1}{\mathbb{E}[D_G]} + z^2\frac{2p_2}{\mathbb{E}[D_G]} + z^3\frac{3p_3}{\mathbb{E}[D_G]} \iff z(p1+2p_2+3p_3) = p_1z + 2p_2z^2 + 3p_3z^3 \tag{3.11}$$

$$\iff z = \frac{6p_3 - 2p_1}{6p_3} = 1 - \frac{p_1}{3p_3}.$$

As long as $3p_3 > p_1$ this solution is valid, also $z = 0$ is disregarded as it is a trivial solution. With this z the typical proportions can be calculated. We can now choose our q_k such that the sum is larger than the sum over q_k^T. Then we solve (3.12) for x.

$$\sum_{k=1}^{3} kq_k = (1-x^2)\sum_{k=1}^{3}\frac{kq_k}{1-x^k} \iff q_1 + 2q_2 + q_3 = (1-x^2)\left[\frac{q_1}{1-x} + \frac{2q_2}{1-x^2} + \frac{3q_3}{1-x^3}\right] \tag{3.12}$$

$$\iff x^2q_1 + x(q_1+3q_3) + q_1 = 0 \iff x = \frac{-q_1+3q_3) \pm \sqrt{9q_3^2 + 6q1q_3 - 3q_1^2}}{2q_1}.$$

Given that x should be positive, we can take the positive root to be our x in this case. With the formula given in (3.5) the normalization parameter $\gamma_k(x)$ can be calculated. Combining everything together allows us to find the proportions $\rho(k-j,j)$ for this instance of the graph, and hence the amount of each label to distribute. Figure 3.1 shows a possible outcome of this procedure.

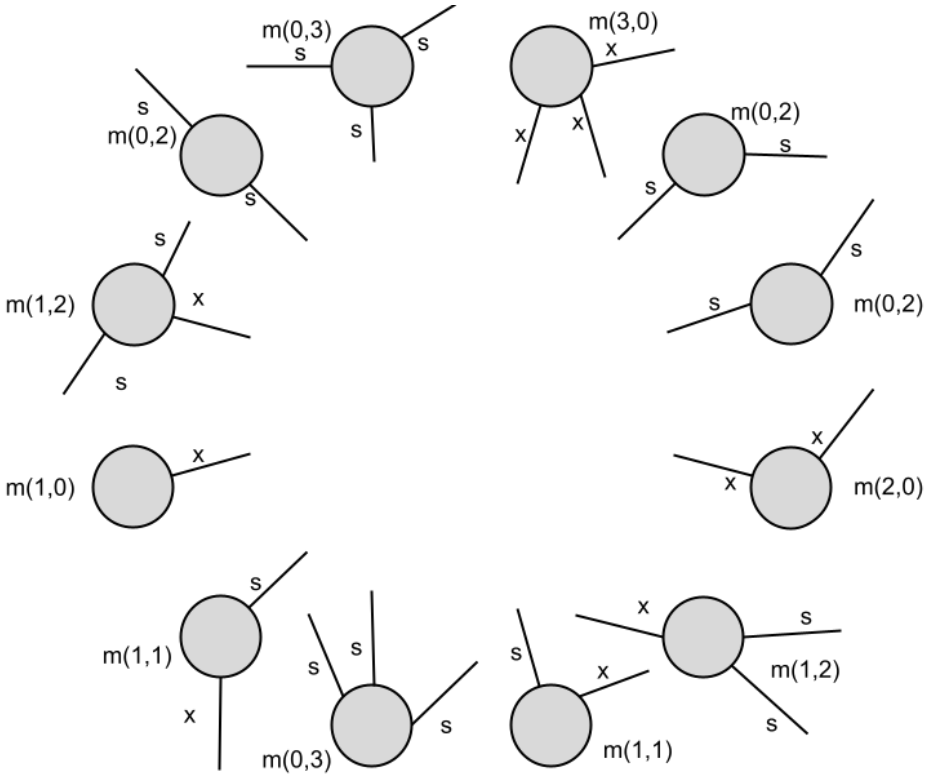


Figure 3.1: Possible start to an coloured half-edges model random graph, showing the degree, s, x labelling and the final $m_n(k-j,j)$ naming.

Now that all labels can be assigned, it is time for the next step: colouring half-edges. The next section will analyse this step of the coloured half-edges model.

3.2 Colouring

Colouring the half-edges is based on their label. The colouring has to do with the structure of the giant component. Typically the giant has a very dense, interconnected part, and then some appendages. In graph theory we consider such an interconnected part a k-core.

Definition 3.1 (k-core)**.** Let $G=(V,E)$ be a graph. A subgraph $S=(\tilde{V},\tilde{E})$ of G is called a k-core if

I S is *induces*: $\tilde{E}=\{\ \{u,v\}\in E \mid u,v\in\tilde{V}\}$,

II Every vertex of S has degree at least k in S: $\forall v\in\tilde{V},\ \deg_S(v)\geq k$,

III S is maximal with regard to II: there is no strictly larger $\tilde{\tilde{V}}\supset\tilde{V}$ such that the induced subgraph on $\tilde{\tilde{V}}$ also has minimum degree $\geq k$.

Remark 3.2.1. The intuitive way to think about a k-core, is to take a graph, and iteratively "pealing" off all the nodes with degree less than k. Thus after every peal, you update the degree and continue if necessary. So if we consider a 2-core, we remove all the nodes that have degree 0, and then all nodes with degree 1. We are then left with very connected pieces of the graph, since all remaining nodes have two or more connections.

For the whole graph, besides the giant there are some smaller components, those are made up from the loose extinction only nodes. Thus all nodes with an $m_n(k,0$ label (with $k\geq 2$ are outside of the giant, and those will be Colour D. The half-edges for the giant component are a bit more elaborate. The dense interconnected part is made up of Colour A, and the appendages of Colour B and Colour C. For each combination of k and j in a label, there is a set way of colouring the

extinction and/or survival half-edges. The symbol $\sim$ shows that half-edges will be a certain colour. Equations 3.13–3.15 show the colouring rules.

$$\forall j \geq 2 \quad \forall (k-j) \geq 0: \quad s \sim \text{Colour A}, x \sim \text{Colour B}. \tag{3.13}$$

$$\forall (k-j) \geq 1 \quad j = 1: \quad s \sim \text{Colour C}, x \sim \text{Colour B}. \tag{3.14}$$

$$\forall (k-j) \geq 1 \quad j = 0 \quad x \sim \text{Colour D}. \tag{3.15}$$

When making a CHEM random-graph, we need to make sure the $m_n(i,j)$ satisfy some conditions, such that the half-edges can be matched later on.

I The number of Colour A half-edges is even:

$$\sum_{j=2}^{\infty} \sum_{i=0}^{\infty} m_n(i,j) j \in 2\mathbb{N}. \tag{3.16}$$

II The number of Colour D half-edges is even:

$$\sum_{i=1}^{\infty} m_n(i,0) i \in 2\mathbb{N}. \tag{3.17}$$

III The number of Colour B half-edges is equal to the number of Colour C half-edges.

$$\sum_{j=1}^{\infty} \sum_{i=1}^{\infty} m_n(i,j) i = \sum_{i=0}^{\infty} m_n(i,1). \tag{3.18}$$

With these rules, Figure 3.1 from Example 3.1 can be continued: the labelled framework can be coloured. Figure 3.2 is the coloured version of the labelled graph (Fig. 3.1).

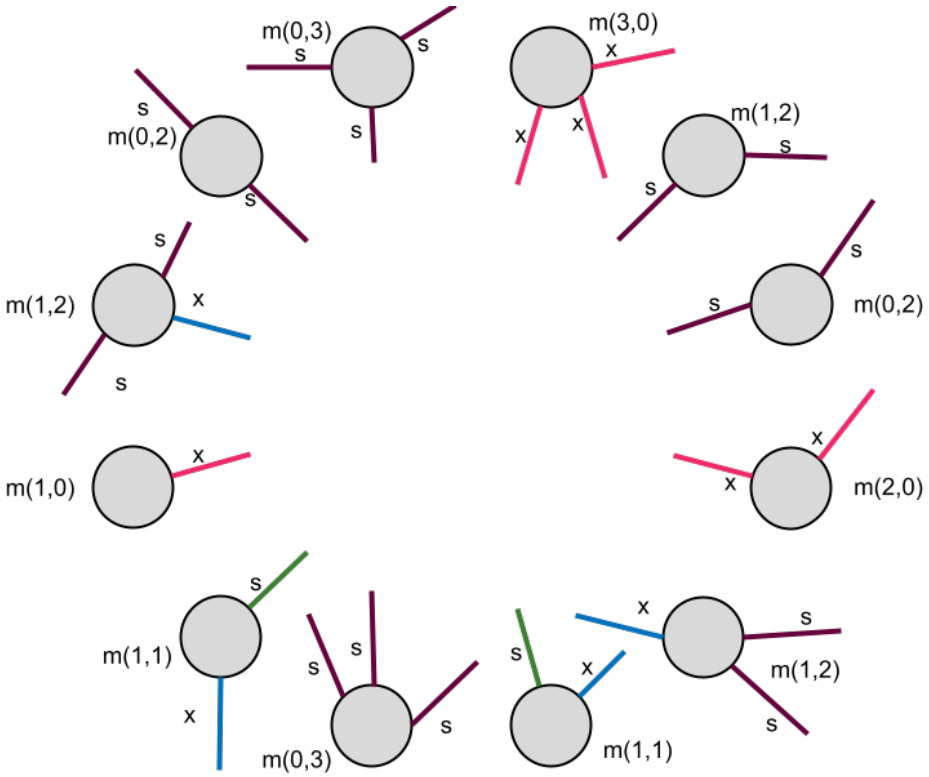


Figure 3.2: CHEM graph with labels and coloured half-edges.

3.3 Matching

Theoretically, matching the coloured half-edges is the last step of the CHEM procedure. Sometimes the half-edges are a bit out of balance, but we will discuss that at length in Section 3.4. The matching procedure is done by following three rules. The rules are simple, Colour A half-edges always connect to other Colour A half-edges. Colour B half-edges always connect to Colour C half-edges, and vice-versa. Lastly, Colour D half-edges always connect to Colour D half-edges. This is of course a random matching, so select a random half-edge in the graph, then based on its colour randomly select any of the appropriate coloured half-edges and connect them.
Figure 3.2 is nicely coloured, so we implement these three rules and start matching the edges. Rearranging the nodes a bit yields our CHEM-random graph shown in Figure 3.3. Two problems stand out from this graph.

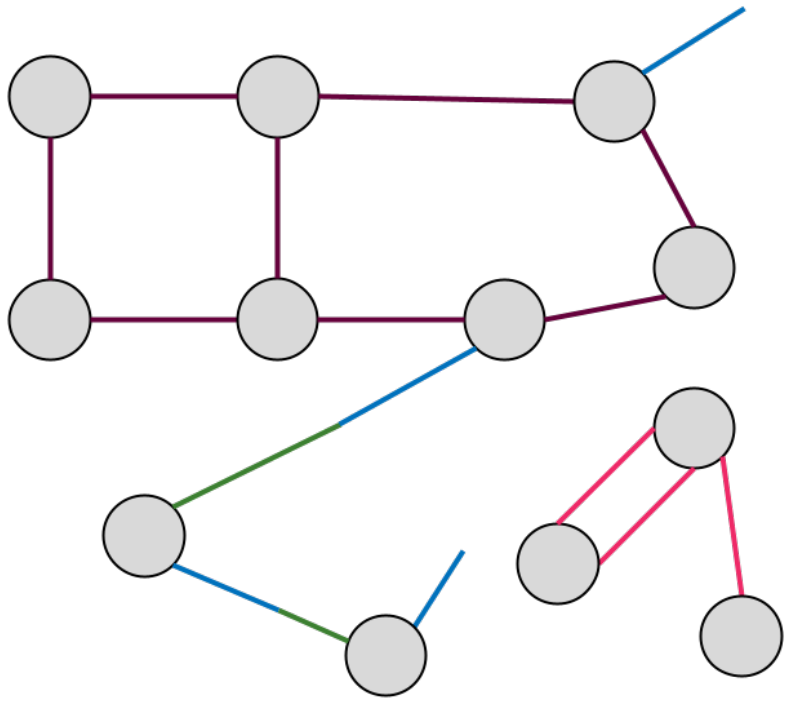


Figure 3.3: Matched, un-tweaked, coloured half-edges model random graph of 12 nodes.

There are two Colour B half-edges unconnected, and there are two nodes which have a double connection between them (Colour D nodes). Section 3.4 will focus on the unconnected half-edges, while this section will discuss the simplicity of the graph.
The CHEM random graph should be a simple graph, so no self-loops or multi-edges (see Remark 2.1.1). The easiest way to solve those problems is by simply removing those "illegal" edges. A paper from 2013 solved this problem for a directed configuration model random graph.[25] The authors proved that in the limit changing a "small" amount of edges does not influence the empirical degree distribution. First, analysing the amount of edges that are non-simple is in order.

Let S be the total amount of self-loops in an instance of a CHEM-graph. Furthermore, let $L(n)$ be the amount of half-edges in a graph. When it causes no ambiguity, we write L instead of $L(n)$. To make a self-loop, the algorithm has to pick two half-edges from the same node. When it has selected a half-edge, the probability of picking another half-edge from that node is $\frac{1}{L-1}$, since it already selected one edge. Then we consider the fact that it could select those two edges in $\binom{d_i}{2}$ ways, by shuffling between all the half-edge combinations of the node. Considering this, we obtain 3.19 for the expected number of self-loops.

$$\mathbb{E}[S] = \sum_{i=1}^{n} \binom{d_i}{2} \frac{1}{L-1} = \sum_{i=1}^{n} \frac{d_i(d_i-1)}{2(L-1)} = \frac{1}{2L-2} \sum_{i=1}^{n} d_i(d_i-1). \tag{3.19}$$

Note that the total amount of half-edges L in a graph is defined as $\sum_{i=1}^{n} d_i$. Substituting this in (3.19) and rearranging some terms yields (3.20).

$$\mathbb{E}[S] = \frac{1}{2L}\sum_{i=1}^{n} d_i^2 - \frac{1}{2}. \tag{3.20}$$

Theorem 3.1. *Let G_n be a CHEM-graph with n nodes where D_G is the degree distribution of the graph. Let S denote the total amount of self-loops in G_n. If D has finite second moment then,*

$$\mathbb{E}[S] = \Theta(1).$$

Thus the total amount of self-loops is a finite amount, independent of n.

Proof. Assume $\mathbb{E}[D^2] < \infty$. By the strong law of large numbers we obtain:

$$\frac{1}{n}\sum_{k=1}^{n} d_i^2 \overset{a.s.}{\to} \mathbb{E}[D^2] < \infty. \tag{3.21}$$

So equivalently, we can write

$$\sum_{k=1}^{n} d_i = n\mathbb{E}[D] + o(n) = O(n). \tag{3.22}$$

$$\sum_{k=1}^{n} d_i^2 = n\mathbb{E}[D^2] + o(n) = O(n). \tag{3.23}$$

Combining these identities, we calculate the following:

$$\frac{1}{2L}\sum_{k=1}^{n} d_i^2 = \frac{1}{2\sum d_i}\sum_{k=1}^{n} d_i^2 = \frac{n\mathbb{E}[D^2] + n\cdot o(n)}{2n\mathbb{E}[D] + 2n\cdot o(n)} = \frac{\mathbb{E}[D^2]}{2\mathbb{E}[D]}. \tag{3.24}$$

$$\iff \mathbb{E}[S] = \frac{O(n) + o(n)}{2O(n) + 2o(n)} = \Theta(1).$$

□

The amount of self-loops is thus constant, independent of the total number of nodes. Now let M be the total amount of multi-edges in an CHEM-graph. The goal is to count the amount of parallel edges between two nodes, denoted by m. Start by selecting two nodes at random: i, j. Very similar to the self-loop case, the probability of connecting them once is given by $\frac{1}{L-1}\binom{d_i}{2}\binom{d_j}{2}$. For more than one connection, say m, the probability becomes something of the form

$$\frac{m!}{(L-1)(L-3)\cdots(L-(2m-1))}\binom{d_i}{m}\binom{d_j}{m}.$$

The only thing missing from this equation is the probability of not having more than m matches, since we are looking at *exactly* m multi-edges. Let us write $R_{i,j}^{(m)}$ for $\mathbb{P}$(No remaining half-edge from "i" is matched to a remaining half-edge of "j").

Lemma 3.1. *Let G_n be a CHEM-graph, M the number of multi-edges in the graph, and $R_{i,j}^{(m)}$ the probability that the amount of multi-edges between nodes "i" and "j" is* exactly *m. Then $\forall m \in \mathbb{N}$ we have*

$$R_{i,j}^{(m)} = 1 - O\left(\frac{d_i + d_j}{L}\right). \tag{3.25}$$

Proof. Consider $1 - R_{i,j}^{(m)}$, the probability that one of the $d_i - m$ half-edges of i *does* match one of the $d_j - m$ half-edges of j. Label the remaining half-edges of i (WLOG) $v_1, v_2, \cdots v_{d_i - m}$. Suppose we start matching those remaining half-edges, say (WLOG) in order of increasing index. The point at which v_r is matched, $2(r-1)$ half-edges have already been matched. The total number of "free" edges left is $L - 2m - 2(r-1)$, of which $d_j - m$ belong to node j. Thus Equation 3.26 shows the probability of matching v_r to one of node j's remaining edges.

$$\frac{d_j - m}{L - 2m - 2(r-1)} \leq \frac{d_j}{L - 2m - 2(r-1)} \leq \frac{d_j}{L - 2m - 2(d_i - m - 1)} \leq \frac{d_j}{L - 2d_i}. \tag{3.26}$$

If we calculate this probability over every r, we get the probability that there is at least one r for which node i and j have an extra edge between them. This gives us Equation 3.27.

$$1 - R_{i,j}^{(m)} \leq \sum_{r=1}^{d_i - m} \frac{d_j}{L - 2d_i} = \frac{d_j(d_i - m)}{L - 2d_i} \leq \frac{d_i d_j}{L - 2d_i} \leq \frac{d_i d_j}{L/2}. \tag{3.27}$$

Finally, we use the fact that $d_i d_j \leq \frac{1}{2}(d_i + d_j)^2$, we find a bound of $O(\frac{d_i+d_j}{L})$. Symmetrically we would have found the same bound when we would have started with node j. Thus the conclusion is indeed,

$$R_{i,j}^{(m)} = 1 - O\left(\frac{d_i + d_j}{L}\right).$$

□

Furthermore, we state lemma 3.2. This lemma says that multi-edges made up of two parallel edges are the dominant term. Thus once can essentially ignore other types of multi-edges when counting, which is precisely what we want.

Lemma 3.2. *Let G_n be a CHEM-graph, M the number of multi-edges in the graph.*

$$O(\mathbb{P}(\textit{Exactly } m \textit{ between } i, j)) = O(\mathbb{P}(\textit{Exactly } 2 \textit{ between } i, j)).$$

Proof. For any $m \in \mathbb{N}$, we have

$$\mathbb{P}(\text{Exactly } m \text{ edges between } i \text{ and } j) = \frac{\binom{d_i}{m}\binom{d_j}{m} m!}{(L-1)(L-3)\cdots(L-(2m-1))} R_{i,j}^{(m)}. \tag{3.28}$$

By Lemma 3.1 $R_{i,j}^{(m)}$ is of order $1 - O\left(\frac{d_i+d_j}{L}\right)$. We calculate
$\mathbb{E}[\text{Number of matches between } i \text{ and } j] = (m-1)\mathbb{P}(\text{Exactly } m \text{ edges between } i \text{ and } j))$. For the case $m = 2$ this becomes:

$$\binom{d_i}{2}\binom{d_j}{2} \frac{2!}{(L-1)(L-3)} \left(1 - O\left(\frac{d_i + d_j}{L}\right)\right). \tag{3.29}$$

As L grows larger with n, the leading term in Equation 3.29 is:

$$\frac{d_i(d_i-1)d_j(d_j-1)}{2L^2} = O\left(\frac{d_i^2 d_j^2}{2L^2}.\right) \tag{3.30}$$

For the case where $m \geq 3$, we remark the following things:

$$(L-1)(L-2)\cdots(L-(2m-1)) = L^m \left(1 + O\left(\frac{m^2}{L}\right)\right), \tag{3.31}$$

$$\binom{d_i}{m}\binom{d_j}{m} m! \leq \frac{d_i^m d_j^m}{m!}. \tag{3.32}$$

Substituting (3.31) and (3.32) into (3.28), yields (3.33).

$$\mathbb{P}(\text{exactly } m \text{ edges}) \leq \frac{d_i^m d_j^m}{m!} \cdot \frac{1}{L^m\left(1 + O\left(\frac{m^2}{L}\right)\right)} \cdot \left(1 - O\left(\frac{d_i + d_j}{L}\right)\right) \tag{3.33}$$

$$d_i d_j \text{ is at least of order } O(L) \text{ thus } \left(\frac{d_i d_j}{L}\right)^m = O\left(\frac{1}{L^m}\right). \tag{3.34}$$

Recombining terms in (3.33), and summing over all $m \geq 3$, where $d_i d_j / L$ is at most $O(L^{-3})$, yields (**??**).

$$\sum_{m=3}^{min\{d_i, d_j\}} \mathbb{P}(\text{exactly } m \text{ edges}) = O\left(\frac{d_i^3 d_j^3}{L^3}\right). \tag{3.35}$$

Comparing (3.30) to (3.35) shows that all the contribution of multi-edges with three or more connections between nodes is of smaller order than the double parallel edges between nodes. Thus the $m = 2$ term strictly dominates the $m \geq 3$ contributions.

□

Theorem 3.2. *Let G_n be an CHEM-graph with n nodes where D_G is the degree distribution of the graph. Let M denote the total amount of multi-edges in G_n. If D has finite second moment then,*

$$\mathbb{E}[M] = \Theta(1).$$

Thus the total amount of multi-edges is a finite amount, independent of n.

Proof.

$$\mathbb{E}[\text{excess parallel edges between } i,j] = \sum_{m=2}^{\min\{d_i,d_j\}} (m-1)\mathbb{P}(\text{Exactly } m \text{ edges between } i \text{ and } j). \tag{3.36}$$

From Lemma 3.2 we know that this probability is dominated by the term $m = 2$. Thus for large enough n we can rewrite (3.36) to:

$$\mathbb{E}[\text{excess parallel edges between } i,j] = \mathbb{E}[\text{two parallel edges between } i,j] + O\left(\frac{d_i^3 d_j^3}{L^3}\right) = \tag{3.37}$$

$$\frac{\binom{d_i}{m}\binom{d_j}{m}}{(L-1)(L-3)} + o\left(\frac{1}{L^2}\right).$$

Combining (3.36) with (3.37) gives the expected value of M.

$$\mathbb{E}[M] = \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\binom{d_i}{m}\binom{d_j}{m}}{(L-1)(L-3)} \approx \frac{1}{2L^2}\sum_{i<j} d_i(d_i-1)d_j(d_j-1). \tag{3.38}$$

The last term in (3.38) can be split into two sums as follows, $\frac{1}{2}[(\sum d_i(d_i-1))^2 - \sum(d_i(d_i-1))^2]$. Working this term into (3.38), yields the following equation:

$$\mathbb{E}[M] = \frac{1}{4L^2}\left[\left(\sum_{i=1}^{n} d_i(d_i-1)\right)^2\right] - O\left(\frac{\sum_i^n d_i^2(d_i-1)^2}{L^2}\right) \text{ as } n \to \infty. \tag{3.39}$$

The first term in (3.39) can be rewritten a bit, where we also use the assumption that $\mathbb{E}[D^2] < \infty \iff \sum d_i^2 = O(L)$.

$$\left(\frac{1}{L}\sum_{i=1}^{n} d_i(d_i-1)\right) = \frac{\sum d_i^2 - \sum d_i}{L} = \frac{\sum d_i^2}{L} - 1 = \lambda - 1 \implies \mathbb{E}[M] = \frac{(\lambda-1)^2}{4} = \Theta(1). \tag{3.40}$$

Where λ in (3.40) comes from $\lim \frac{1}{L}\sum d_i^2 = \lambda$.

□

We have proven that the amount of self-edges and multi-edges in our CHEM graph is constant, thus we can surely say that we can make our graph simple without disrupting the degree distributions. Let us formalize this in a theorem.

Theorem 3.3. *Let G_n be an CHEM random graph of n vertices, and degree distribution $\vec{p}$. By removing self-loops and multi-edges from G_n we obtain the simple CHEM graph $\tilde{G}_n$ with degree distribution $\vec{\tilde{p}}$. If $\mathbb{E}[D^2] < \infty$ then*

$$|\vec{p} - \vec{\tilde{p}}| \to 0 \text{ as } n \to \infty. \tag{3.41}$$

Proof. By Theorem 3.1 and Theorem 3.2 we have that $\mathbb{E}[S+M] = \Theta(1)$. Thus we remove a constant number of self-loops and multi-edges, so the degree distribution $\vec{\tilde{p}}$ is given by

$$\tilde{p}_k = \frac{np_k - \Theta(1)}{n} \overset{n\to\infty}{\to} p_k, \tag{3.42}$$

for every $\tilde{p}_k$ in $\vec{\tilde{p}}$ and for every p_k in $\vec{p}$. □

Making a CHEM graph simple does not influence the degree distribution in the limit. Thus the graph in Figure 3.3 can be made simple, yielding Figure 3.4.
The last issue visible in Figure 3.4 is the two loose Colour B half-edges. These signal that we generated either too many Colour B half-edges, or too few Colour C half-edges. The next section will focus on tweaking the CHEM-graph, such that it is possible to 'complete' the matching.

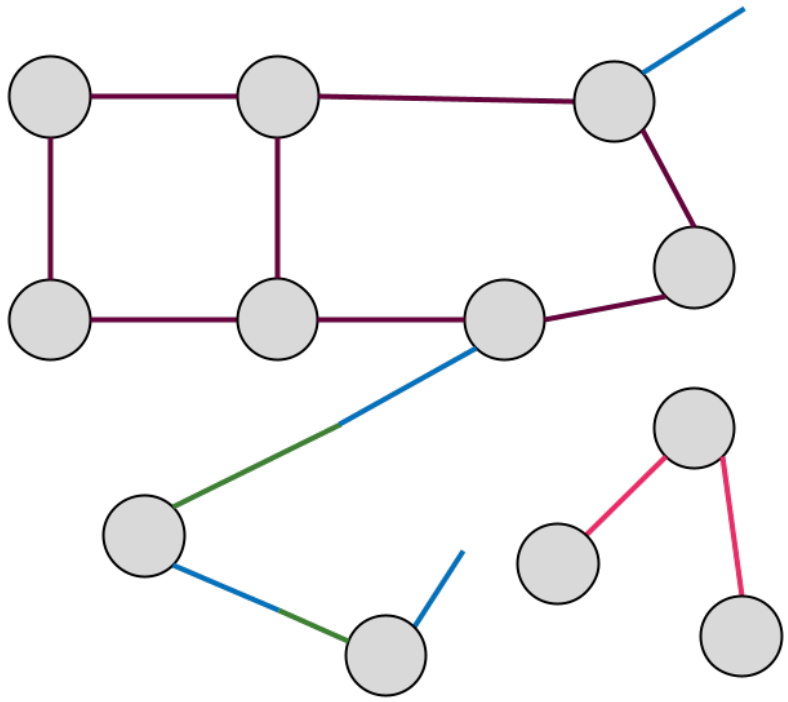


Figure 3.4: Matched, un-tweaked, CHEM graph of 12 nodes, made simple.

3.4 Tweaking

The labelling of a CHEM graph has to satisfy conditions (3.16)-(3.18). However, due to fluctuations/rounding errors the conditions are not always satisfied when creating the labelling $m_n(i,j)$. The maximum error in the Colour A and Colour D half-edges is one for each, which would make them uneven. For the Colour B and Colour C half-edges, the maximum error is K where K is the amount of degrees in the graph, which we will state in the following lemma.

Lemma 3.3. *Let G_n be a CHEM graph of n nodes. Let Δ denote the difference in the amount of Colour B and Colour C half-edges. Then for all n, $|\Delta| \leq K$ where K is the amount of different degrees in G_n.*

Proof. $m_n(i,j) = \lfloor n\rho(i,j) \rfloor = \lfloor n\rho(k-j,j) \rfloor$. Because $m_n(i,j)$ is always an integer, and we can only calculate $\rho(k-j,j)$ via (3.10). All Colour B half-edges come from labels with at least one survival edge. Thus for all K degrees $K-1$ can have Colour B half-edges. All Colour C half-edges come from labels with at most one survival edge. Thus all Colour C half-edges come from two of the total degrees. There is one label that is shared by Colour B and Colour C half-edges, which is the $m_n(1,1)$ label. This case can be ignored, because any error in this label will add/remove as many Colour B as Colour C half-edges.
In the worst-case scenario, for each $\rho(i,j)$ with $j \geq 1$ we have:

$$|m_n(i,j) - \rho(i,j)| < 1,$$

and for each $\rho(i,j)$ with $j \leq 1$

$$|m_n(i,j) - \rho(i,j)| < 1.$$

This shows that the over-/undershoot of Colour B/Colour C half-edges is in any case less than K, since e.g. the overshoot of Colour B is at most $K-1$ and the undershoot of Colour C is at most 1 (or vice-versa). Thus we conclude that $\Delta \leq K$ □

Since the amount of degrees in a CM/CHEM graph is independent of the amount of vertices, the amount of half-edges that need to be tweaked is of order 1.

The goal of our tweaking algorithm is to make sure the labelling satisfies conditions (3.16)-(3.18) in one manoeuvre. Thus all conditions will be "fixed" at the same time. Before we get into the algorithm, we need a very useful lemma regarding graphs.

Lemma 3.4 (Handshake lemma[26]). *Let $G = (V, E)$ be a finite undirected graph. Then*

$$\sum_{v \in V} \deg(v) = 2|E|. \tag{3.43}$$

Equivalently, the amount of nodes of odd degree is even.

The handshake lemma tells us that the amount of half-edges is always even. We can relate this to the difference in Colour B and Colour C half-edges ($= \Delta$). If Δ is even, then the amount of Colour A half-edges plus the amount of Colour D half-edges is also even.
Recall that we write $p_2(n)$ (p_2 in short) for the proportion of vertices in a graph G_n with degree equal to two. Now let us introduce our tweaking algorithm.

Definition 3.2 (Tweaking algorithm). Consider a CHEM random-graph G_n with limit degree distribution $\vec{p} = \{p_k\}_{k=1}^{\infty}$. Furthermore, let $\vec{q} = \{q_k\}_{k=1}^{\infty}$ be such that, for all k we have $0 \leq q_k \leq p_k$.
Assume that $p_2 \neq 0, q_2 \neq 0$, and that n is large enough such that $n\rho(1,1)$ is sufficiently large. Denote the difference in Colour B and Colour C half-edges by Δ, and denote by $\hat{A}, \hat{D}$ the amount of Colour A and Colour D half-edges, respectively. For a labelling of the graph which does not satisfy conditions (3.16)-(3.18), we consider the following cases;

I Suppose $\Delta > 0$.

- i Suppose $\hat{A}, \hat{D}$ are both even.
  Randomly choose Δ many $m_n(1,1)$ nodes, and relabel them to $m_n(0,1)$.
- ii Suppose $\hat{A}$ is even, $\hat{D}$ is odd.
  We know that $\hat{A} + \hat{D}$ is odd, so Δ is odd too. Thus we can write $\Delta = 2\ell + 1$ for some $\ell \in \mathbb{Z}$. Randomly choose $2\ell + 2$ many $m_n(1,1)$ nodes, and change them to $m_n(0,1)$. Then randomly choose one $m_n(0,1)$ node and change it to $m_n(1,0)$.
- iii Suppose $\hat{A}$ is odd, $\hat{D}$ is even.
  We know that $\hat{A} + \hat{D}$ is odd, so Δ is odd too. Thus we can write $\Delta = 2\ell + 1$ for some $\ell \in \mathbb{Z}$. Randomly choose one $m_n(1,j)$ node, where $j \geq 2$, and relabel it to $m_n(0, j+1)$. Then choose 2ℓ many $m_n(1,1)$ nodes and relabel them to $m_n(0,1)$.

II Suppose $\Delta < 0$.

- i Suppose $\hat{A}, \hat{D}$ are both even.
  Randomly choose Δ many $m_n(0,1)$ nodes, and relabel them to $m_n(1,1)$.
- ii Suppose $\hat{A}$ is even, $\hat{D}$ is odd.
  We know that $\hat{A} + \hat{D}$ is odd, so Δ is odd too. Thus we can write $\Delta = 2\ell + 1$ for some $\ell \in \mathbb{Z}$. Randomly choose 2ℓ many $m_n(0,1)$ nodes, and change them to $m_n(1,1)$. Then randomly choose one additional $m_n(0,1)$ node and change it to $m_n(1,0)$.
- iii Suppose $\hat{A}$ is odd, $\hat{D}$ is even.
  We know that $\hat{A} + \hat{D}$ is odd, so Δ is odd too. Thus we can write $\Delta = 2\ell + 1$ for some $\ell \in \mathbb{Z}$. Randomly choose one $m_n(0,j)$ label with $j \geq 3$ and change it to $m_n(1, j-1)$. Then randomly choose 2ℓ many $m_n(0,1)$ and relabel them to $m_n(1,1)$.

In this way any way of not satisfying conditions (3.16)-(3.18) is covered. Thus we are able to tweak any labelling such that the new labelling does satisfy (3.16)-(3.18).

Getting back to Example 3.1 for the last time, we see that Figure 3.4 (originally Fig. 3.1) is a variant of case (I:i), thus we use the tweaking algorithm to change two $m_{12}(1,1)$ labels to $m_{12}(0,1)$. If we then rematch we obtain Figure 3.5.

3.5 CHEM graph algorithms

Let us summarize the key aspects of this chapter, and give a formal way of creating a CHEM-random graph.

Definition 3.3 (CHEM labelling algorithm). Let $n \in \mathbb{N}$ and consider a known degree distribution $\vec{p}(n) = \{p_k(n)\}_{k=0}^{\infty}$. Create a labelling $\mathcal{L}_{CHEM}(n)$ by;

I Find z that solves:

$$z = G_{Z_{D_G}}(z).$$

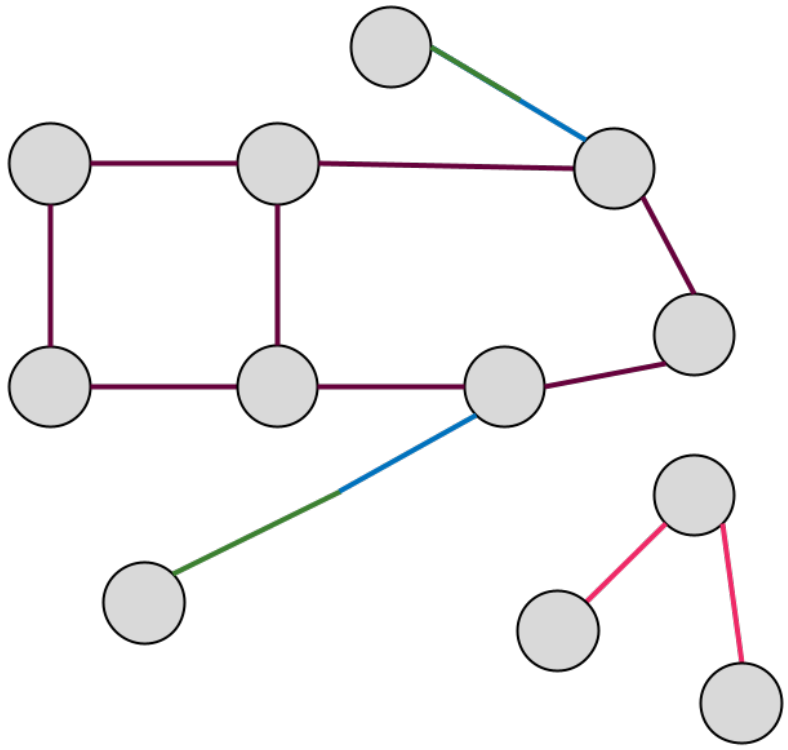


Figure 3.5: Matched and tweaked CHEM-random graph of 12 vertices.

II Calculate q_k^T for every k:

$$q_k^T = p_k(n)(1 - z^k).$$

III Select q_k such that $0 \leq q_k \leq p_k(n)$.

IV Find x that solves:

$$\sum_{k=1}^{n} q_k = (1 - x^2) \sum_{k=1}^{n} \frac{q_k}{1 - x^k}.$$

V Compute $\gamma_k(x)$:

$$\gamma_k(x) = \frac{p_k(1 - x^k)}{q_k}.$$

VI Calculate the fraction of label:

$$\rho(k - j, j) = \frac{p_k \binom{k}{j} x^{k-j}(1 - x)^j}{\gamma_k(x)}.$$

VII Make $m_n(k - j, j)$ labels by:

$$m_n(k - j, j) = \lfloor n\rho(k - j, j) \rfloor \ \forall k, j.$$

VIII Use the tweaking algorithm to make sure this labelling satisfies conditions (3.16)-(3.18).

Definition 3.4 (CHEM colouring algorithm)**.** Let $\mathcal{L}_{CHEM}(n)$ be a tweaked labelling. For each $m_n(i, j) \in \mathcal{L}_{CHEM}(n)$ group it according to

I) $j \geq 1$.

- $j \geq 2$
  The labelled node gets i Colour B half-edges and j Colour A half-edges.
- $j = 1$
  The labelled node gets j Colour C half-edges and i Colour B half-edges.

II) $j = 0$.

- $i > 0$.
  The labelled node gets i Colour D half-edges.

Applying these steps yields a coloured CHEM foundation-graph[1] according to the labelling $\mathcal{L}_{CHEM}(n)$.

Definition 3.5 (CHEM matching algorithm)**.** Let $\tilde{G}_n$ be a CHEM foundation-graph. Match half-edges in $\tilde{G}_n$ by the following steps.

I For every Colour A half-edges, match it with another Colour A half-edge.

II For every Colour B half-edge, match it with a Colour C half-edge and vice-versa.

III For every Colour D half-edge, match it with another Colour D half-edge.

When the CHEM matching algorithm is applied to a CHEM foundation-graph, a CHEM random graph is obtained. As discussed in Section 3.3 one can freely make this a simple CHEM graph by removing the self-loops and multi-edges, since it will not affect the properties of the graph (for large n).

Definition 3.6 (CHEM random graph)**.** For some $n \in \mathbb{N}$ let $\vec{p}$ be a degree distribution, and let $\vec{q}$ be a vector of proportions satisfying condition (3.3). A $\text{CHEM}_n(\vec{p}, \vec{q})$ random graph, is a graph created by using the three CHEM algorithms: Def. 3.3, Def. 3.4, and Def. 3.5.
We often drop the $(\vec{p}, \vec{q})$ part, since in the general cases we do not give an explicit $\vec{p}$ or $\vec{q}$ anyway.

[1] Since the graph consists only of coloured half-edges it is not a true graph yet.

Chapter 4

Explicit CHEM graphs

This chapter will explore the CHEM graphs that we created. We want to know if CHEM graphs indeed reach the Bhamidi lower-bound entropy (Thm. 2.8) for example. Furthermore, we will simulate making CHEM graphs to test convergence of the degree proportions and giant density.

4.1 Entropy

Definition 4.1 (Remainder term)**.** Consider a labelling $\mathcal{L}_{CHEM}(n)$. We define the remainder term $r_k(x)$ by:

$$r_k(x) = \gamma_k(x) - (1 - x^k). \tag{4.1}$$

Definition 4.2 (Half edges count)**.** Let G_n be an CHEM$_n$ random graph. We denote by H_A, H_B, H_C, H_D the amount of half-edges For each type of half-edge: Colour A, Colour B, Colour C, and Colour D respectively in G_n.

Before we express these half-edge counts, we introduce some short hand notation.

Definition 4.3 (Sum shorthand)**.** Let $\vec{p} = \{p_k\}_{k\geq 0}$ be a proportion vector, and let $\{z_k\}_{k\geq 0}$ be a sequence larger than 0. We introduce the following notation:

$$H_x = \sum_{k=1}^{\infty} \frac{kp_k}{z_k} x^{k-2}. \tag{4.2}$$

$$N_x = \sum_{k=1}^{\infty} \frac{kp_k}{z_k}. \tag{4.3}$$

Now, let us derive expressions for these $H_{A,B,C,D}$ parameters. The idea for each is the same, for each k, j we take the proportion of half-edges and count the permutations.

$$H_A = \sum_{k=1}^{\infty} \frac{p_k}{\gamma_k(x)} \underbrace{\sum_{j=2}^{k} \binom{k}{j}(1-x)^j x^{k-j} \cdot j}_{=\mathbb{E}[\mathrm{Bin}(k,1-x)]-\binom{k}{1}x^{k-1}(1-x)} = \sum_{k=1}^{\infty} \frac{kp_k}{\gamma_k(x)}(1-x)(1-x^{k-1})$$

$$= (1-x)\sum_{k=1}^{\infty} \frac{kp_k}{\gamma_k(x)}(1-x^{k-1}) = (1-x)\left[\sum_{k=1}^{\infty} \frac{kp_k}{\gamma_k(x)} - \sum_{k=1}^{\infty} \frac{kp_k}{\gamma_k(x)}x^{k-1}\right] = (1-x)\left[N_x - xH_x\right]. \tag{4.4a}$$

$$H_B = \sum_{k=1}^{\infty} \frac{p_k}{\gamma_k(x)} \underbrace{\sum_{j=1}^{k} \binom{k}{j}(1-x)^j x^{k-j} \cdot (k-j)}_{=E[\mathrm{Bin}(k,x)]-\binom{k}{0}x^k k} = \sum_{k=1}^{\infty} \frac{kp_k}{\gamma_k(x)}(x-x^k).$$

$$= x\sum_{k=1}^{\infty} \frac{kp_k}{\gamma_k(x)}(1-x^{k-1}) = x\left[\sum_{k=1}^{\infty} \frac{kp_k}{\gamma_k(x)} - \sum_{k=1}^{\infty} \frac{kp_k}{\gamma_k(x)}x^{k-1}\right] = x\left[N_x - xH_x\right]. \tag{4.4b}$$

$$H_C = \sum_{k=1}^{\infty} \frac{p_k}{\gamma_k(x)} \sum_{j=1}^{k} \binom{k}{1}(1-x)x^{k-1} = \sum_{k=1}^{\infty} \frac{kp_k}{\gamma_k(x)}(x^{k-1}-x^k).$$

$$= x\sum_{k=1}^{\infty} \frac{kp_k}{\gamma_k(x)}(x^{k-2}-x^{k-1}) = x\left[\sum_{k=1}^{\infty} \frac{kp_k}{\gamma_k(x)}x^{k-2} - \sum_{k=1}^{\infty} \frac{kp_k}{\gamma_k(x)}x^{k-1}\right] = x[H_x - xH_x]. \tag{4.4c}$$

Finally, since we defined the remainder term (Def. 4.1) exactly for this, we give the following simple expression for H_D.

$$H_D = \sum_{k=1}^{\infty} \frac{p_k r_k(x)}{\gamma_k(x)}. \tag{4.5}$$

Now we look at the following lemma which introduces the consistency equation, which will help simplify the expressions of the half-edge counts.

Lemma 4.1 (Consistency equation)**.** *For CHEM graphs it is essential to have equally many Colour B half-edges as Colour C half-edges as aforementioned. This means that $H_B = H_c$ which is equivalent to $H_x = N_x$.*

Proof. This is immediately obvious from the expression H_B and H_C themselves. See (4.4b) and (4.4c). □

Definition 4.4 (Co-incision event)**.** Let $\mathcal{E}(\{p(k-j,j\}_{k\geq 1, j\leq k})$ be the event that a configuration model random graph coincides with an equally large CHEM random graph.

Theorem 4.1 (CHEM entropy)**.** *Let G_n be a $\mathrm{CHEM}_n(\vec{p},\vec{q})$ random graph. The rate function/entropy of G_n is given by:*

$$\frac{1}{2}(1-x^2)H_x \ln(H_x) + \frac{1}{2}H_D \ln(H_D) - \frac{1}{2}\mathbb{E}[D]\ln(\mathbb{E}[D]) + \sum_{k=1}^{\infty} p_k \ln(\gamma_k(x)) - \sum_{k=1}^{\infty} \frac{p_k r_k(x)}{\gamma_k(x)} \ln(r_k(x)). \tag{4.6}$$

Proof. The rate function is calculated by $\frac{1}{n}\mathbb{P}(\mathcal{E}(\{p(k-j,j\}_{k\geq 1, j\leq k})$. This probability consists of the following two parts.

$$P_1 = \prod_{k=1}^{\infty} \left(\frac{(np_k)!}{\prod_{j=0}^{k}(n\rho(k-j,j))!} \prod_{j=0}^{k} \binom{k}{j}^{n\rho(k-j,j)} \right). \tag{4.7}$$

$$P_2 = \frac{H_A!! \cdot H_B! \cdot H_D!!}{(n\mathbb{E}[D])!!}. \tag{4.8}$$

Let us first focus on $\frac{1}{n}\ln(P_1)$. Using the product property of logarithms, we immediately start with:

$$\frac{1}{n}\ln(P_1) = \frac{1}{n}\sum_{k=1}^{\infty}\left(\ln(np_k!) - \sum_{j=0}^{k}\ln([n\rho(k-j,j)]!) + \sum_{j=0}^{k} n\rho(k-j,j)\ln\binom{k}{j}\right).$$

Here we use the Sterling approximation for the log of a factorial.

$$= \frac{1}{n}\sum_{k=1}^{\infty}\left(np_k\ln(np_k) - np_k - \sum_{j=0}^{k}[n\rho(k-j,j)\ln(n\rho(k-j,j)) - n\rho(k-j,j)] + \sum_{j=0}^{k} n\rho(k-j,j)\ln\binom{k}{j}\right).$$

We note that $\sum_{j=0}^{k} n\rho(k-j,j) = np_k$.

$$= \sum_{k=1}^{\infty}\left(p_k\ln(p_k) - \sum_{j=0}^{k}\rho(k-j,j)\ln(\rho(k-j,j)) + \rho(k-j,j)\ln\binom{k}{j}\right).$$

$$= \sum_{k=1}^{\infty}\left(p_k\ln(p_k) - \sum_{j=0}^{k}\rho(k-j,j)\ln(\frac{\rho(k-j,j)}{\binom{k}{j}})\right).$$

Rewriting $\rho(k-j,j)$ yields the following:

$$= \sum_{k=1}^{\infty} p_k\ln(p_k) - \sum_{k,j}\frac{p_k}{\gamma_k(x)}\binom{k}{j}x^{k-j}(1-x)^j\ln\left(\frac{p_k}{\gamma_k(x)}x^{k-j}(1-x)^j\right).$$

$$= \sum_{k=1}^{\infty} p_k\ln(p_k) - \sum_{k=1}^{\infty}\sum_{j=1}^{k}\frac{p_k}{\gamma_k(x)}\binom{k}{j}x^{k-j}(1-x)^j\ln\left(\frac{p_k}{\gamma_k(x)}x^{k-j}(1-x)^j\right) - \sum_{k=1}^{\infty}\frac{p_k r_k(x)}{\gamma_k(x)}\ln\left(\frac{p_k r_k(x)}{\gamma_k(x)}\right). \tag{4.9}$$

We collect terms for each logarithm, to see what the final expression will be. Denote by $C\{\ln(\cdots)\}$ the coefficient of $\ln(\cdots)$.

$$C\{\ln(p_k)\} = \left[p_k - \sum_{j=1}^{k}\left(\frac{p_k}{\gamma_k(x)}\binom{k}{j}x^{k-j}(1-x)^j\right) - \frac{p_k r_k(x)}{\gamma_k(x)}\right] = \left[p_k - \frac{p_k}{\gamma_k(x)}\left(\sum_{j=1}^{k}\binom{k}{j}x^{k-j}(1-x)^j - r_k(x)\right)\right].$$

Note that the definition of $r_k(x)$ allows us to rewrite $\gamma_k(x)$ to $\gamma_k(x) = r_k(x) + (1-x^k)$.

Furthermore, $\sum_{j=1}^{k}\binom{k}{j}x^{k-j}(1-x)^j = 1-x^k$.

$$= \left[p_k - \frac{p_k}{\gamma_k(x)}((1-x^k) + r_k(x))\right] = \left[p_k - \frac{p_k}{\gamma_k(x)}\gamma_k(x)\right] = 0. \tag{4.10}$$

$$C\{\ln(\gamma_k(x)\} = \left[\sum_{j=1}^{k}\left(\frac{p_k}{\gamma_k(x)}\binom{k}{j}x^{k-j}(1-x)^j\right) + \frac{p_k r_k(x)}{\gamma_k(x)}\right]$$

$$= \left[\frac{p_k}{\gamma_k(x)}\left(\sum_{j=1}^{k}\binom{k}{j}x^{k-j}(1-x)^j + r_k(x)\right)\right] = \frac{p_k}{\gamma_k(x)}\gamma_k(x) = p_k \tag{4.11}$$

$$C\{\ln(1-x)\} = \left[-\sum_{k=1}^{\infty}\sum_{j=1}^{k}\frac{p_k}{\gamma_k(x)}\binom{k}{j}x^{k-j}(1-x)^j\cdot j\right]$$

We recognize here the expectation of $\mathrm{Bin}(k,(1-x))$

$$= \sum_{k=1}^{\infty}\frac{p_k}{\gamma_k(x)}k(1-x) = -(1-x)H_x. \tag{4.12}$$

$$C\{\ln(x)\} = \left[-\sum_{k=1}^{\infty}\sum_{j=1}^{k} \frac{p_k}{\gamma_k(x)} \binom{k}{j} x^{k-j}(1-x)^j \cdot (k-j) \right]$$

We recognize here the expectation of $\text{Bin}(k, x)$ where the first term is missing ($j = 0$).

$$= \sum_{k=1}^{\infty} \frac{p_k}{\gamma_k(x)} kx(1-x) = -x(1-x)H_x. \tag{4.13}$$

Let us now do the same thing for $\frac{1}{n}\ln(P_2)$. We will use the simpler forms for the $H_{A,B,C,D}$ parameters shown in (4.4a) - (4.5). Moreover, we have $N_x = H_x$ from Lemma 4.1. We use this fact to simplify the $H_{A,B,C,D}$ parameters even further.

$$H_A = (1-x)[N_x - xH_x] = (1-x)H_x(1-x) = (1-x)^2 H_x. \tag{4.14}$$

$$H_B = xH_x(1-x) = (x-x^2)H_x. \tag{4.15}$$

With these simplified forms, we start expanding $\frac{1}{n}\ln(P_2)$, using the product property again to start with the following:

$$\begin{aligned}
\frac{1}{n}\ln(P_2) &= \frac{1}{n}\sum_{k=1}^{\infty}\left(\frac{n}{2}H_A\ln(H_A) - \frac{n}{2}H_A + nH_B\ln(H_B) + \frac{n}{2}H_D\ln(H_D) - \frac{1}{2}n\mathbb{E}[D]\ln(\mathbb{E}[D]) + \frac{1}{2}\mathbb{E}[D]\right). \\
&= \sum_{k=1}^{\infty}\left(\frac{1}{2}(1-x)^2 H_x \ln\left((1-x)^2 H_x\right) + (x-x^2)H_x\ln\left(x(1-x)H_x\right) + \frac{1}{2}H_D\ln(H_D)\right). \\
&+ \sum_{k=1}^{\infty}\frac{1}{2}\mathbb{E}[D] - \frac{1}{2}\mathbb{E}[D]\ln(\mathbb{E}[D]) - \frac{1}{2}\left(H_A + 2H_B + H_D\right).
\end{aligned}$$

We use the fact that $\mathbb{E}[D] = H_A + H_B + H_C + H_D$ to obtain $\mathbb{E}[D] = H_A + 2H_B + H_D$.

$$= \sum_{k=1}^{\infty}\left(\frac{1}{2}(1-x)^2 H_x \ln\left((1-x)^2 H_x\right) + (x-x^2)H_x\ln\left(x(1-x)H_x\right) + \frac{1}{2}H_D\ln(H_D) - \frac{1}{2}\mathbb{E}[D]\ln(\mathbb{E}[D])\right). \tag{4.16}$$

Now we again start collecting terms for each logarithm.

$$C\{\ln(H_x)\} = \left[\frac{1}{2}(1-x)^2 H_x + (x-x^2)H_x\right] = \frac{1}{2}(1-x^2)H_x. \tag{4.17}$$

$$C\{\ln(x)\} = x(1-x)H_x. \tag{4.18}$$

$$C\{\ln(1-x)\} = \left[(1-x^2)H_x + (x-x^2)H_x\right] = (1-x)H_x. \tag{4.19}$$

Note that, (4.12) cancels with (4.18) and the same goes for (4.13) cancelling (4.19). Thus, the remaining terms are:

$$p_k\ln(\gamma_k(x)), -\frac{p_k r_k(x)}{\gamma_k(x)}\ln(r_k(x)), \frac{1}{2}(1-x^2)H_x\ln(H_x), \frac{1}{2}H_D\ln(H_D), -\frac{1}{2}\mathbb{E}[D]\ln(\mathbb{E}[D]). \tag{4.20}$$

Which finished the proof. □

CHEM graphs should realise an entropy which exactly recovers the entropy lower-bound given as the Bhamidi lower-bound, to be a true importance sampling of this rare event.

Theorem 4.2. *Let G_n be a $\text{CHEM}_n(\vec{p}, \vec{q})$ random graph. Then the entropy of G_n exactly recovers the lower bound given by Bhamidi* et al. *in Theorem 2.8.*

Proof. From Theorem 4.1 we have the entropy of a CHEM graph. Let us now show that this entropy is exactly the rate function $I(\vec{p},\vec{q})$ by rewriting the rate function.

Before we start we remark some identities, which will make it much easier to rewrite certain terms.

$$q_k = \frac{p_k(1-x^k)}{\gamma_k(x)}. \tag{4.21}$$

$$p_k - q_k = \frac{p_k r_k(x)}{\gamma_k(x)}. \tag{4.22}$$

$$\begin{aligned} H(\vec{q}) &= \sum_{k=1}^{\infty} q_k \ln(q_k) - \frac{1}{2}\left(\sum_{k=1}^{\infty} kq_k\right)\ln\left(\sum_{k=1}^{\infty} kq_k\right). \\ &= \sum_{k=1}^{\infty} \frac{p_k(1-x^k)}{\gamma_k(x)} \ln\left(\frac{p_k(1-x^k)}{\gamma_k(x)}\right) - \frac{1}{2}\left(\sum_{k=1}^{\infty} k\frac{p_k(1-x^k)}{\gamma_k(x)}\right)\ln\left(\sum_{k=1}^{\infty} k\frac{p_k(1-x^k)}{\gamma_k(x)}\right). \\ &\text{Note that } \sum_{k=1}^{\infty} \frac{kp_k(1-x^k)}{\gamma_k(x)} = N_x - x^2 H_x = (1-x^2)H_x. \\ &= \sum_{k=1}^{\infty} \frac{p_k(1-x^k)}{\gamma_k(x)} \ln\left(\frac{p_k(1-x^k)}{\gamma_k(x)}\right) - \frac{1}{2}(1-x^2)H_x \ln\left((1-x^2)H_x\right). \end{aligned} \tag{4.23}$$

$$\begin{aligned} -H(\vec{p}) &= -\sum_{k=1}^{\infty} p_k \ln(p_k) + \frac{1}{2}\left(\sum_{k=1}^{\infty} kp_k\right)\ln\left(\sum_{k=1}^{\infty} kp_k\right). \\ &= -\sum_{k=1}^{\infty} p_k \ln(p_k) + \frac{1}{2}\left(\mathbb{E}[D]\right)\ln\left(\mathbb{E}[D]\right). \end{aligned} \tag{4.24}$$

$$\begin{aligned} H(\vec{p}-\vec{q}) &= \sum_{k=1}^{\infty} (p_k - q_k)\ln(p_k - q_k) - \frac{1}{2}\left(\sum_{k=1}^{\infty} k((p_k - q_k)\right)\ln\left(\sum_{k=1}^{\infty} k((p_k - q_k)\right). \\ &\text{Recall the definition of } H_D = \sum_{k=1}^{\infty} \frac{kp_k r_k(x)}{\gamma_k(x)}. \\ &= \sum_{k=1}^{\infty} \frac{p_k r_k(x)}{\gamma_k(x)} \ln\left(\frac{p_k r_k(x)}{\gamma_k(x)}\right) - \frac{1}{2}H_D \ln(H_D). \end{aligned} \tag{4.25}$$

We see that we have already recovered the terms $\frac{1}{2}\mathbb{E}[D]\ln(\mathbb{E}[D])$ and $-\frac{1}{2}H_D\ln(H_D)$. Let us now collect coefficients of the logarithms as we did before.

$$C\{\ln(H_x)\} = -\frac{1}{2}(1-x^2)H_x. \tag{4.26}$$

$$\begin{aligned} C\{\ln(\gamma_k(x))\} &= \left[-\frac{p_k(1-x^k)}{\gamma_k(x)} - \frac{p_k r_k(x)}{\gamma_k(x)}\right]. \\ &= -\left[\frac{p_k\left((1-x^k) + r_k(x)\right)}{\gamma_k(x)}\right] = -\frac{p_k\gamma_k(x)}{\gamma_k(x)} = -p_k. \end{aligned} \tag{4.27}$$

$$C\{\ln(r_k(x)\} = \frac{p_k r_k(x)}{\gamma_k(x)}. \tag{4.28}$$

$$C\{\ln(p_k)\} = \left[\frac{p_k\left(1-x^k\right)}{\gamma_k(x)} - p_k + \frac{p_k r_k(x)}{\gamma_k(x)}\right] = \frac{p_k}{\gamma_k(x)}\left[(1-x^k) + r_k(x) - \gamma_k(x)\right]$$
$$= \frac{p_k}{\gamma_k(x)}[0] = 0. \tag{4.29}$$

Finally we look at $K(\vec{q})$.

$$K(\vec{q}) = \left(\frac{1}{2}\sum_{k=1}^{\infty} k q_k\right)(1-x^2) - \sum_{k=1}^{\infty} q_k \ln\left(1-x^k\right).$$
$$= \frac{1}{2}(1-x^2)H_x - \sum_{k=1}^{\infty}\frac{p_k(1-x^k)}{\gamma_k(x)}\ln\left(1-x^k\right). \tag{4.30}$$

We now collect the remaining coefficients.

$$C\{\ln\left(1-x^k\right)\} = \left[\frac{p_k\left(1-x^k\right)}{\gamma_k(x)} - \frac{p_k\left(1-x^k\right)}{\gamma_k(x)}\right] = 0 \tag{4.31}$$

So the terms that we have left over at the and of rewriting are:

$$-\frac{1}{2}(1-x^2)H_x, -p_k\ln(\gamma_k(x)), \frac{p_k r_k}{\gamma_k(x)}\ln(r_k(x)), -\frac{1}{2}H_D\ln(H_D), \frac{1}{2}\mathbb{E}[D]\ln(\mathbb{E}[D]). \tag{4.32}$$

Which are exactly the negative five terms that make up the CHEM graph entropy. If we look back at Theorem 2.8 we see that the rate function is given as $-I(\vec{p},\vec{q})$, thus we where looking for the negative of the CHEM entropy, which finished the proof. □

4.2 Simulation

Testing our CHEM graphs is important. We want to see if it indeed realises the event $E^{n,\varepsilon}(\vec{q})$, and if so around what n it does. We program the CHEM graph algorithms into Python[27], which is done in three parts.

First, we find x (the solution to (3.4)) and z (the extinction rate of the giant component). Furthermore, we compute the typical giant density θ and typical giant degree distribution q_k^T. With those variables and parameters we pre-compute the proportions $\rho(i,j)$ needed for the CHEM graph, based on an initial degree proportion $\vec{p}$. Readers are referred to Appendix A.1 for details.

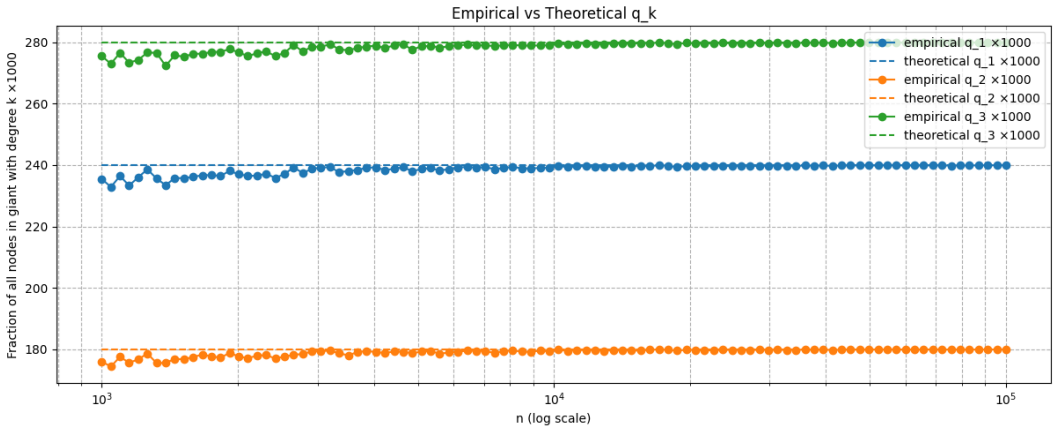
A CHEM random graph can then be generated for a starting degree proportion $\vec{p}$, a size n and a goal giant degree proportion $\vec{q}^{\mathcal{T}}$. Since this is a proof of concept, we limit the code and set a maximum degree of $k=3$. The full encoded CHEM can be found in Appendix A.2.

To test the code, we rely on the Monte Carlo method: We choose a $\vec{p}$ and a $\vec{q}^{\mathcal{T}}$. Then, for a range of n values, we make R CHEM-graphs and track the empirical giant density and degree distribution. The code for this program is located at Appendix A.3.

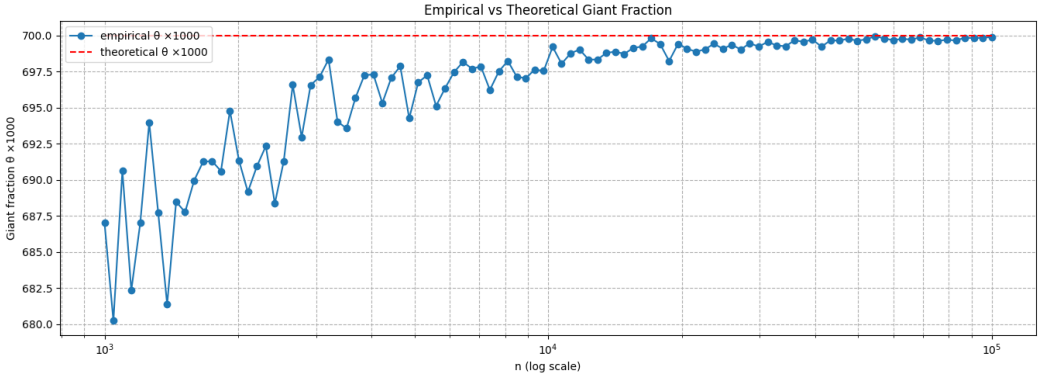
4.2.1 Validation

To validate the encoded CHEM model, we want to test two things in particular. If we pick $\vec{q}^{\mathcal{T}} = \vec{q}^{T}$ we would expect a $CM_n(\vec{d})$ graph to be outputted, where $\vec{d}$ is such that the degree proportions $\vec{p}$ match that of the CHEM model.

So, if we input a degree proportion vector $\vec{p} = \{0.5, 0.2, 0.5\}$ for example, we can calculate the typical giant density and degree proportion via (3.1) and (3.2) respectively. In this instance, we have $\theta = 0.697$ and $q_1^T = \{0.222, 0.138, 0.249\}$. Let us validate this by setting $\vec{q}^{T} = \vec{q}^{\mathcal{T}}$.
We run simulations from $n = 1.000$ to $n = 100.000$ in 100 steps, and for each n value we take the average of 10 simulations. Figure 4.1 shows the giant density graph (Fig. 4.1b) and the giant degree proportions (Fig. 4.1a).
Figure 4.1 shows that the encoded CHEM$_n$ graph converges in empirical properties to the theoretical properties of a CM$_n$ graph. Furthermore, we need to verify that the theoretical values of the encoded CHEM match a configuration



(a) Mean empirical giant degree proportions in a $CHEM_n$ graph of ten simulations at different n values.



(b) Mean empirical giant density in a $CHEM_n$ graph of ten simulations at different n values.

Figure 4.1: Validation of the encoded CHEM by comparing empirical to theoretical values.

model. To this end the *Networkx* package[28] us utilized to generate a CM graph, from which we check the giant density and degree proportions. The code of this module is located at Appendix A.4. Figure 4.2 shows the mean empirical giant density and degree proportion of a CM graph for different n values, where for each n the average over ten simulations is taken. The weighted averages (Appendix A.5.1 for details) are extremely comparable to the theoretical values of the CHEM graph.
To qualify the validation of the encoded CHEM we present some test statistics.

Definition 4.5 (Root mean square deviation / RMSD [29][30])**.** Let $X_1, \cdots, X_n$ be a sample with true mean x_t, then the RMSD of the sample is given by

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - x_t)^2}. \tag{4.33}$$

Low RMSD scores signify a small deviation between the sample and the true mean.

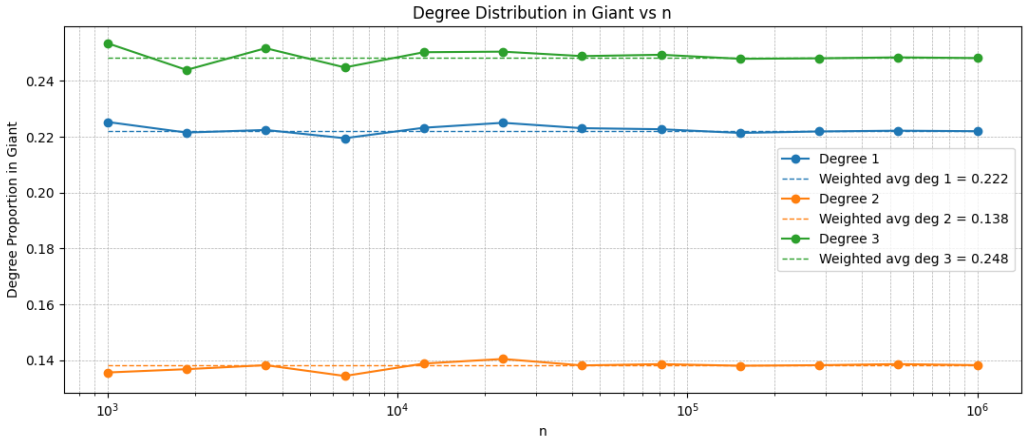
Definition 4.6 (Pearson's Chi-squared [31])**.** Let $\{p_k^{emp}\}_{k\geq 1}$ be the empirical degree distributions and $\{p_k^{th}\}_{k\geq 1}$ the theoretical degree distribution. Define the test statistic χ^2 as

$$\chi^2 := \sum_{k\geq 1}\frac{(p_k^{emp} - p_k^{th})^2}{p_k^{th}}. \tag{4.34}$$

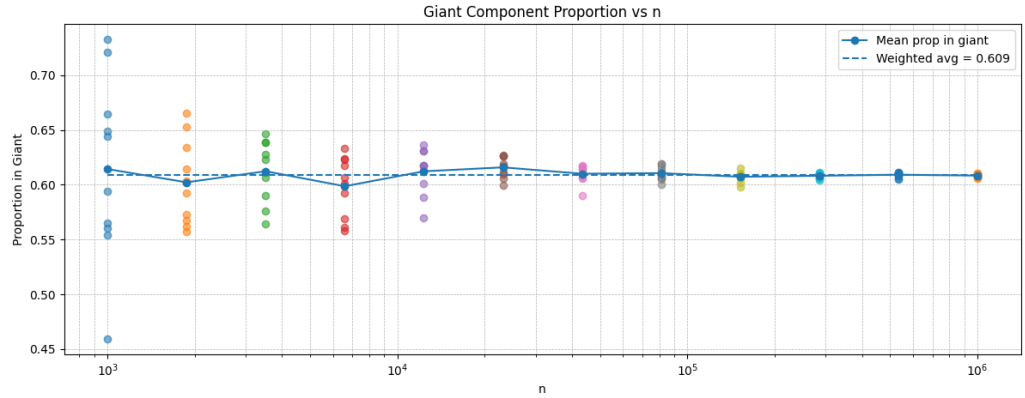
The Pearson's Chi-squared test are on the null-hypothesis: the empirical results are i.i.d. drawn from the theoretical distribution. We then test our statistic on the lower-tail i.e. $\mathbb{P}(\chi^2_{theory} \leq \chi^2_{emp})$. Thus low p-value correspond to a good fit of the empirical values.
To ensure the encoded CHEM works as intended, let us set the statistical significance at $\alpha = 0.05$. In Table 4.1 we present the RMSD score, and p-value of the Pearson's Chi-squared test for the CHEM graphs shown in Figure 4.1.

Statistic	**RMSE**	$p-$**value**
Giant	0.006	0.057
q_1	0.002	0.035
q_2	0.002	0.029
q_3	0.002	0.035

Table 4.1: Validation statistics for giant density and degree proportions for CHEM graphs seen in Fig. 4.1.



(a) Mean empirical giant degree proportions in a CM_n graph of ten simulations at different n values. Dashed line represents the weighted average over mean degree proportions.



(b) Mean empirical giant density in a CM_n graph of ten simulations at different n values. Coloured dots show the giant density in each of the ten simulations per n value. Dashed line represents the weighted average over mean giant density.

Figure 4.2: Mean empirical values for a CM_n graph of different sizes n.

4.2.2 Results

In the previous section we validated the encoded CHEM graph. Let us now use this code to (try) simulate a deviant giant. Since we validated the code for $\vec{p} = \{0.5, 0.2, 0.3\}$ let us keep this degree distribution. The corresponding, typical, giant degree proportions are $\vec{q} = \{0.22, 0.138, 0.249\}$, we now select our desired giant degree proportions. When selecting our $\vec{q}$ we have to keep in mind conditions (3.3) and (**??**).
The target giant degree proportion vector $\{0.24, 0.18, 0.28\}$ satisfies both aforementioned conditions, thus we move forwards with this vector $\vec{q}$.

Here we run simulations from $n = 1.000$ up to $n = 10.000.000$ in 100 steps, to get clear results. For each n-value we run 10 simulations and take the average, to get rid of any weird fluctuations. Figure 4.3 shows the giant degree proportions, along with a dashed line representing the target proportions we defined.
Figure 4.4 shows the giant density, also with a dashed line representing the sum of our q_k values, as that is the target density. We can see that for large n-values the CHEM graphs converge nicely to the target values, showing that we have (numerically) achieved importance sampling for the event $E^{n,\varepsilon}(\vec{q})$.

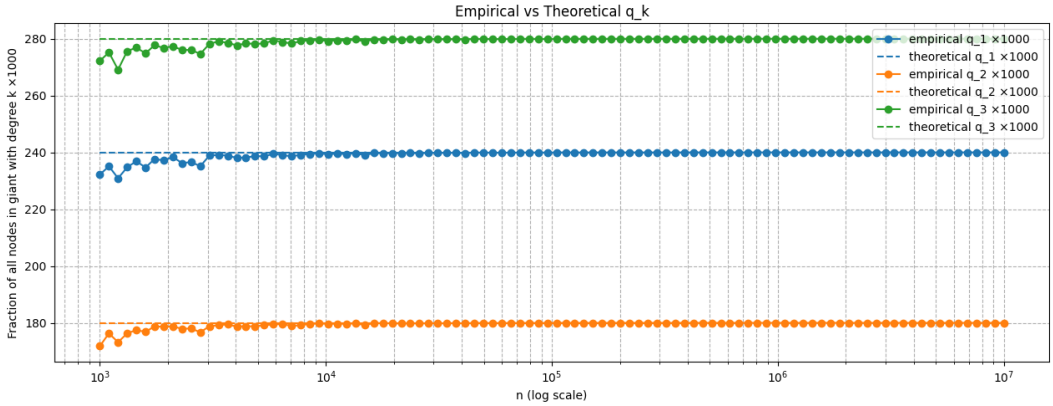


Figure 4.3: Mean empirical giant degree proportion in a $\mathrm{CHEM}_n(\{0.5, 0.2, 0.3\}, \{0.24, 0.18, 0.28\})$ graph of ten simulations at different n-values.

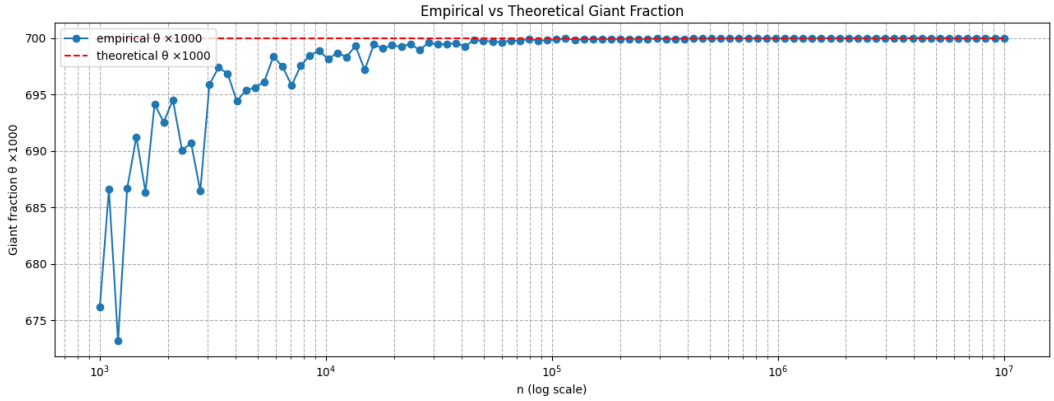


Figure 4.4: Mean empirical giant density in a $\mathrm{CHEM}_n(\{0.5, 0.2, 0.3\}, \{0.24, 0.18, 0.28\})$ graph of ten simulations at different n-values.

Chapter 5

Conclusion & Discussion

In this thesis, we developed the Coloured Half-Edges Model (CHEM), a construction designed to perform importance sampling for the rare event that the size of the giant component in configuration model graphs deviates. Specifically, we focused on the event $E^{n,\varepsilon}(\vec{q})$, were we force the giant to have atypical degree proportions and thus a deviating giant distribution.
We began by introducing the CHEM algorithm in detail, explaining how to construct graphs that mimic the degree distribution of the configuration model while being biased toward a desired (atypical) giant size. This approach offers a way of increasing the likelihood of observing rare events without fundamentally changing the nature of the graphs created.
We then showed that the entropy of the resulting graphs matches the rate function for the large deviation principle (LDP) as derived by Bhamidi *et al.*, confirming that CHEM targets the correct exponential tilt. This is a key result, as it demonstrates that the model is not just heuristically motivated but is also theoretically significant.
Finally, we conducted simulations of CHEM graphs across varying parameters. These simulations confirmed the model's effectiveness: the generated graphs consistently exhibited the prescribed rare component size distributions, validating both the theoretical and practical viability of the method.

This work provides one of the first constructive methods to simulate rare events in random graphs from the configuration model using importance sampling. While the LDP describes the asymptotic decay rate of these rare events, standard Monte Carlo methods are inefficient in capturing them. CHEM bridges this gap by enabling direct simulation of graphs with atypical giants, which opens the door for empirical studies of rare but critical structural behaviours in networks.

While the results are of importance, several limitations should be acknowledged:

- **Graph types:** The current formulation is specific to the undirected configuration model. Extending CHEM to more general models (e.g. directed graphs, models with clustering, or community structure) may require significant adaptation.
- **Focus on the giant:** The model targets deviations in the size of the giant component. While this is a central quantity in random graph theory, other types of rare events (e.g., large diameters, multiple giants, spectral outliers) remain outside CHEM's current scope.
- **Restricted simulations:** The simulations of CHEM graphs where limited to a degree distribution where only degree one, two, and three nodes are considered.

Several future directions for research around this topic naturally emerge from this thesis:

- **Alternative rare events:** It would be valuable to investigate whether the principles of CHEM can be adapted to simulate other rare events in the configuration model, like clustering or suppressed giant scenarios.
- **Empirical validation:** Applying CHEM to real-world networks, or to synthetic networks mimicking empirical distributions, could provide insight into whether rare graph structures observed in practice correspond to LDP predictions.
- **Information-theoretic bounds:** The entropy framework used in CHEM invites connections to information theory. It would be interesting to explore whether there are tighter bounds or optimality guarantees that can be derived for such importance sampling schemes.
- **Expanding simulations:** Extending CHEM simulations to larger degree distributions will be valuable as most real-life networks/graphs feature a higher maximum degree than three.

This thesis contributes both a theoretical and practical advancement in the study of rare events in random graphs. By introducing an effective sampling method that aligns with the underlying large deviation structure, we provide researchers with a new tool for probing the tails of the configuration model. In a world where extreme behaviours often carry the greatest consequences — from network failures to viral outbreaks — such tools are not just mathematically interesting but increasingly essential.

Bibliography

[1] M. Dyble, J. Thompson, D. Smith, *et al.*, "Networks of food sharing reveal the functional significance of multilevel sociality in two hunter-gatherer groups," *Curr. Biol.*, vol. 26, no. 15, pp. 2017–2021, Aug. 2016.

[2] N. B. Ellison, C. Steinfield, and C. Lampe, "The benefits of facebook "friends:" social capital and college students' use of online social network sites," *Journal of Computer-Mediated Communication*, vol. 12, no. 4, pp. 1143–1168, 2007.

[3] M. Newman, *Networks: An Introduction*. Oxford University Press, 2010.

[4] B. Bollobás, "A probabilistic proof of an asymptotic formula for the number of labelled regular graphs," *European Journal of Combinatorics*, vol. 1, no. 4, pp. 311–316, 1980. DOI: 10.1016/S0195-6698(80)80030-8.

[5] M. Molloy and B. Reed, "The size of the giant component of a random graph with a given degree sequence," *Combinatorics, Probability and Computing*, vol. 7, no. 3, pp. 295–305, 1998. DOI: 10.1017/S0963548398003526.

[6] S. Bhamidi, A. Budhiraja, P. Dupuis, and R. Wu, "Rare event asymptotics for exploration processes for random graphs," *Annals of Applied Probability*, vol. 32, no. 2, pp. 1112–1178, 2022. DOI: 10.1214/21-AAP1704.

[7] L. Andreis, W. König, and R. I. A. Patterson, "A large-deviations principle for all the cluster sizes of a sparse erdős–rényi graph," *Random Structures & Algorithms*, vol. 56, no. 4, pp. 1130–1180, 2020. DOI: 10.1002/rsa.20976.

[8] L. Andreis, W. König, H. Langhammer, and R. I. A. Patterson, "A large-deviations principle for all the components in a sparse inhomogeneous random graph," *Probability Theory and Related Fields*, vol. 186, pp. 521–620, 2023. DOI: 10.1007/s00440-022-01180-7.

[9] A. Agazzi, L. Andreis, R. I. A. Patterson, and D. Renger, "Large deviations for markov jump processes with uniformly diminishing rates," *Stochastic Processes and their Applications*, vol. 154, pp. 140–175, 2023. DOI: 10.1016/j.spa.2022.07.005.

[10] R. Jorritsma and B. Zwart, "Large deviations of the giant component in scale-free inhomogeneous random graphs," *arXiv preprint*, vol. arXiv:2407.01224, 2024. [Online]. Available: https://arxiv.org/abs/2407.01224.

[11] J. A. Bucklew, "An introduction to rare event simulation," *Springer Series in Statistics*, 2004. DOI: 10.1007/b97416.

[12] A. Guyader and H. Touchette, "Efficient large-deviation estimation based on importance sampling," *Journal of Statistical Physics*, vol. 181, no. 3, pp. 551–586, 2020. DOI: 10.1007/s10955-020-02502-y.

[13] A. Dembo, A. Montanari, and N. Sun, "Importance sampling for large deviations in random graphs," *arXiv preprint arXiv:1811.00895*, 2018.

[14] P. Erdős and A. Rényi, "On random graphs i," *Publicationes Mathematicae*, vol. 6, pp. 290–297, 1959.

[15] E. N. Gilbert, "Random graphs," *Ann. Math. Stat.*, vol. 30, no. 4, pp. 1141–1144, Dec. 1959.

[16] I. Benjamini and O. Schramm, "Recurrence of distributional limits of finite planar graphs," *Electronic Journal of Probability*, vol. 6, pp. 1–13, 2001. DOI: 10.1214/EJP.v6-96.

[17] R. van der Hofstad, *Random Graphs and Complex Networks: Volume 2*. Cambridge University Press, 2023, ISBN: 9781009266400. DOI: 10.1017/9781009266424.

[18] T. E. Harris, *The Theory of Branching Processes*. Springer, 1963.

[19] R. W. van der Hofstad, *Random Graphs and Complex Networks. Volume 1* (Cambridge Series in Statistical and Probabilistic Mathematics). Cambridge University Press, 2017, vol. 43, ISBN: 978-1-107-17287-6. DOI: 10.1017/9781316779422.

[20] B. Bollobás, *Random Graphs*, 2nd. Cambridge: Cambridge University Press, 2001.

[21] S. Janson and M. J. Luczak, "A new approach to the giant component problem," *ArXiv e-prints*, vol. 0707.1786, 2007, arXiv:0707.1786 [math.CO]. [Online]. Available: http://arxiv.org/abs/0707.1786v1.

[22] H. Cramér, "Sur un nouveau théorème-limite de la théorie des probabilités," *Actualités Scientifiques et Industrielles*, no. 736, pp. 5–23, 1938.

[23] A. Dembo and O. Zeitouni, *Large Deviations Techniques and Applications* (Applications of Mathematics), 2nd. New York: Springer, 1998, vol. 38.

[24] Wikipedia contributors, *Intermediate value theorem — wikipedia, the free encyclopedia*, Accessed: 2025-06-20, 2024. [Online]. Available: https://en.wikipedia.org/wiki/Intermediate_value_theorem.

[25] N. Chen and M. Olvera-Cravioto, "Directed random graphs with given degree distributions," en, *Stoch. Syst.*, vol. 3, no. 1, pp. 147–186, Jun. 2013.

[26] L. Euler, "Solutio problematis ad geometriam situs pertinentis," *Commentarii Academiae Scientiarum Imperialis Petropolitanae*, vol. 8, pp. 128–140, 1736, Reprinted in Opera Omnia, Series I, Volume 24.

[27] Python Software Foundation, *Python language reference, version 3.11*, Python Software Foundation, 2023. [Online]. Available: https://www.python.org/.

[28] A. A. Hagberg, P. J. Swart, and D. Schult, *NetworkX: Network Analysis in Python*, https://networkx.org/, Python package, version 3.1.5, 2023.

[29] Wikipedia contributors, *Root mean square deviation*, https://en.wikipedia.org/wiki/Root_mean_square_deviation, Accessed: 2025-06-17, 2025.

[30] R. J. Hyndman and A. B. Koehler, "Another look at measures of forecast accuracy," *International Journal of Forecasting*, vol. 22, no. 4, pp. 679–688, 2006.

[31] K. Pearson, "On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 50, no. 302, pp. 157–175, 1900.

Chapter A

Encoded CHEM

A.1 Precomputation code

```
    #Bhamidi solver
from math import *
import numpy as np
from scipy.optimize import brentq

def find_extinction_rate(q1, q2 ,q3):
    D = sqrt(3*q3**2 - 2*q1*q3 - q1**2)*sqrt(3)
    a = q1
    b = q1 - 3*q3
    x1 = (-b + D)/(2*a)
    x2 = (-b - D)/(2*a)
    return 1-min(x1,x2)

def get_proportion(p1, p2, p3, q1,q2,q3):
   x = find_extinction_rate(q1,q2,q3)
   beta1 = p1 * (1-x) / q1
   beta2 = p2 * (1-x**2) / q2
   beta3 = p3 * (1-x**3) / q3

   p01 = p1 * (1-x)/beta1
   p10 = p1 - q1

   p02 = p2 * (1-x)**2/beta2
   p11 = p2 * 2 * (1-x)*x /beta2
   p20 = p2 - q2

   p03 = p3 * (1-x)**3/beta3
   p12 = p3 * 3 * x * (1-x)**2 /beta3
   p21 = p3 * 3 * x**2 * (1-x) /beta3
   p30 = p3 - q3
   return p10, p01, p02, p11, p20, p03, p12, p21, p30


def q_maker(p1,p2,p3):
    z = 1-(6*p3 - 2*p1)/ (6*p3)
    q1 = p1 * (1-z)
    q2 = p2 * (1-z**2)
    q3 = p3 * (1-z**3)
    return q1, q2, q3

# p1,p2,p3 = 0.5,0.2,0.3
# q1,q2,q3 = q_maker(p1,p2,p3)
# print(sum(get_proportion(p1,p2,p3,q1,q2,q3)))
```

```
## 1. Check average degree
def find_theta(p1,p2,p3):
    z = 1- (6*p3 - 2*p1)/ (6*p3)
    theta = p1*(1-z) + p2*(1-z**2) + p3*(1-z**3)
    return theta


def giant_fraction_3(p1, p2, p3, tol=1e-12):
    """
    Compute the giant-component fraction     for a degree law
    that puts mass p1,p2,p3 on degrees 1,2,3 (and p0=0).
    """
    # build P = [p0, p1, p2, p3]
    P = np.array([0.0, p1, p2, p3], dtype=float)
    ks = np.arange(len(P))
    mean_k = np.dot(ks, P)
    if mean_k <= 1:
        return 0.0

    # G(s) = sum_k p_k s^k
    def G(s):
        return np.dot(P, s**ks)

    # G1(s) = (G'(s)) / G'(1) = sum_{k>=1} k p_k s^(k-1) / mean_k
    def G1(s):
        return np.dot(ks[1:] * P[1:], s**(ks[1:] - 1)) / mean_k

    # find x in (0,1): G1(x) = x
    lo, hi = tol, 1.0 - tol
    x_star = brentq(lambda x: G1(x) - x, lo, hi, xtol=tol, rtol=tol)

    # giant fraction = 1 - G(x_star)
    return 1.0 - G(x_star)

def lift_q_above_theta(p1, p2, p3, epsilon=1e-6, tol=1e-12):
    # 1) Compute old q and true theta
    q_old = np.array(q_maker(p1, p2, p3), dtype=float)
    theta = giant_fraction_3(p1,p2,p3)
    q = q_old.copy()

    # 2) Pack p into an array
    p = np.array([p1, p2, p3], dtype=float)

    # 3) Set strict target
    target = theta + epsilon
    # but cannot exceed sum(p); cap it if necessary
    target = min(target, p.sum())

    # 4) Compute how much to add
    delta = target - q.sum()
    if delta <= tol:
        # already     theta + epsilon (or no room left)
        return q

    # 5)  W a t e r filling  loop
    cap = p - q
    active = cap > tol
    while delta > tol and active.any():
```

```
        C = cap[active].sum()
        inc = np.zeros_like(q)
        inc[active] = cap[active] * (delta / C)

        # apply and clip
        q += inc
        q = np.minimum(q, p)

        # recompute
        delta = target - q.sum()
        cap = p - q
        active = cap > tol

    return q

def lift_q_above_theta2(p1, p2, p3, eps = 0.000001, seed= None):
    q1, q2, q3 = q_maker(p1,p2,p3)
    q3N = np.random.uniform(q3,p3)
    q2N = np.random.uniform(q2,p2)
    q1N = np.random.uniform(q1,q3-eps)
    Q=[q1N, q2N, q3N]
    return Q
```

A.2 Main code for CHEM graphs

```
    import igraph as ig
import numpy as np
from Bhamidi_solver import get_proportion, q_maker
from collections import Counter
from scipy.optimize import brentq
import matplotlib.pyplot as plt

# Helper to partition integer
def random_partition_integer(total, parts, rng):
    if parts == 1:
        return [total]
    if total == 0:
        return [0] * parts
    if total < parts:
        arr = [1] * total + [0] * (parts - total)
        rng.shuffle(arr)
        return arr
    cuts = sorted(rng.choice(np.arange(1, total), size=parts - 1, replace=False))
    return [a - b for a, b in zip(cuts + [total], [0] + cuts)]

# Generate initial split proportions for (i,j) node labels
def fix_and_split_proportions(p1, p2, p3, n, q1=None, q2=None, q3=None, seed=None):
    rng = np.random.default_rng(seed)
    p10, p01, p02, p11, p20, p03, p12, p21, p30 = get_proportion(p1, p2, p3, q1, q2, q3)
    prop_array = np.array([p10, p01, p02, p11, p20, p03, p12, p21, p30], dtype=float)
    prop_array[prop_array < 0] = 0
    prop_array /= prop_array.sum()
    counts = np.round(prop_array * n).astype(int)
    split = {
        (1,0): counts[0], (0,1): counts[1], (0,2): counts[2],
        (1,1): counts[3], (2,0): counts[4], (0,3): counts[5],
        (1,2): counts[6], (2,1): counts[7], (3,0): counts[8],
    }
```

```
    return split, [q1, q2, q3]

# Map (i,j) to stub labels
def get_stubs_per_node(i, j):
    return {
        (0,1): ['C'], (1,0): ['D'],
        (0,2): ['A','A'], (1,1): ['B','C'], (2,0): ['D','D'],
        (0,3): ['A','A','A'], (1,2): ['B','A','A'],
        (2,1): ['B','B','C'], (3,0): ['D','D','D']
    }.get((i,j), [])

# Tweaking algorithm implementation
def tweak_labels(split, node_labels, rng):
    B = sum(split[lbl] * get_stubs_per_node(*lbl).count('B') for lbl in split)
    C = sum(split[lbl] * get_stubs_per_node(*lbl).count('C') for lbl in split)
       = B - C
    A = sum(i * count for (i,j), count in split.items())
    D = sum(j * count for (i,j), count in split.items())

    def choose_and_relabel(frm, to, k):
        if k <= 0 or split.get(frm,0) == 0:
            return
        candidates = [node for node, lbl in node_labels.items() if lbl == frm]
        k2 = min(k, len(candidates))
        if k2 == 0:
            return
        chosen = rng.choice(candidates, size=k2, replace=False)
        for node in chosen:
            node_labels[node] = to
        split[frm] -= k2
        split[to] = split.get(to,0) + k2

    if    > 0:
        if A%2==0 and D%2==0:
            choose_and_relabel((1,1),(0,1),  )
        elif A%2==0 and D%2==1:
               =(  -1)//2
            choose_and_relabel((1,1),(0,1),2*   +2)
            choose_and_relabel((0,1),(1,0),1)
        else:
               =(  -1)//2
            for (i,j),cnt in split.items():
                if i==1 and j>=2 and cnt>0:
                    choose_and_relabel((1,j),(0,j+1),1)
                    break
            choose_and_relabel((1,1),(0,1),2*   )
    elif    < 0:
        abs =-
        if A%2==0 and D%2==0:
            choose_and_relabel((0,1),(1,1),abs )
        elif A%2==0 and D%2==1:
               =(abs  -1)//2
            choose_and_relabel((0,1),(1,1),2*   )
            choose_and_relabel((0,1),(1,0),1)
        else:
               =(abs  -1)//2
            for (i,j),cnt in split.items():
                if i==0 and j>=3 and cnt>0:
```

```
                    choose_and_relabel((0,j),(1,j-1),1)
                    break
            choose_and_relabel((0,1),(1,1),2*    )
    return split, node_labels

# Build and analyze graph using node_labels
# Now signature takes node_labels instead of split
def fast_linear_rule_constrained_config_model_igraph(node_labels, seed=None):
    rng = np.random.default_rng(seed)
    n_nodes = len(node_labels)
    stub_dict = {'A': [], 'B': [], 'C': [], 'D': []}

    for node, lbl in node_labels.items():
        for s in get_stubs_per_node(*lbl):
            stub_dict[s].append(node)
    for s in stub_dict:
        rng.shuffle(stub_dict[s])

    edges = []
    # Pair A with A and D with D
    for label in ['A', 'D']:
        stubs = stub_dict[label]
        for idx in range(0, len(stubs) - 1, 2):
            edges.append((stubs[idx], stubs[idx+1]))
    # Pair B with C
    b_stubs, c_stubs = stub_dict['B'], stub_dict['C']
    rng.shuffle(b_stubs); rng.shuffle(c_stubs)
    for u, v in zip(b_stubs, c_stubs):
        edges.append((u, v))

    g = ig.Graph(n_nodes)
    g.add_edges(edges)

    comps = g.connected_components()
    comp_sizes = comps.sizes()
    idx = np.argmax(comp_sizes) if comp_sizes else -1
    giant = comps.subgraph(idx) if idx >= 0 else None

    stats = {
        'nodes': g.vcount(),
        'edges': g.ecount(),
        'components': len(comp_sizes),
        'giant_size': giant.vcount() if giant else 0,
        'degree_sequence': g.degree()
    }
    return g, stats, giant

# Theoretical giant breakdown (unchanged)
def giant_degree_breakdown(p1, p2, p3, tol=1e-12):
    P = np.array([0.0, p1, p2, p3])
    ks = np.arange(len(P))
    mean_k = np.dot(ks, P)
    if mean_k <= 1:
        return {'theta': 0, 'all_k_in_giant': np.zeros_like(P)}
    def G(s): return np.dot(P, s**ks)
    def G1(s): return np.dot(ks[1:] * P[1:], s**(ks[1:]-1)) / mean_k
    x = brentq(lambda x: G1(x) - x, tol, 1 - tol)
    theta = 1 - G(x)
```

```
    all_k = P * (1 - x**ks)
    return {'theta': theta, 'all_k_in_giant': all_k}

# Main execution
if __name__ == "__main__":
    p1, p2, p3 = 0.5, 0.2, 0.3
    q1, q2, q3 = 0.24, 0.18, 0.28
    n = 10000
    seed = None
    rng = np.random.default_rng(seed)

    # 1) initial split & lift
    split, Q = fix_and_split_proportions(p1, p2, p3, n, q1, q2, q3, seed)

    # 2) build node_labels
    node_labels = {}
    idx = 0
    for lbl, cnt in split.items():
        for _ in range(cnt):
            node_labels[idx] = lbl
            idx += 1

    # 3) tweak node_labels & split
    split, node_labels = tweak_labels(split, node_labels, rng)

    # 4) build graph from node_labels
    g, stats, giant = fast_linear_rule_constrained_config_model_igraph(node_labels, seed)

    # remaining analysis unchanged


    # Theoretical breakdown (classical)
    breakdown = giant_degree_breakdown(p1, p2, p3)
    original_q = np.array(q_maker(p1, p2, p3))
    lifted_q = np.array(Q)

    # Empirical breakdown inside the simulated giant
    degs_gc = giant.degree()
    counts = Counter(degs_gc)
    total_gc = len(degs_gc)
    empirical_props = {k: counts.get(k, 0) / n for k in sorted(counts)}

    # —— Nice, single-block printout ——
    print("\n\n=== Configuration Model Analysis Results ===")
    print(f"Total nodes: {stats['nodes']:,}    Edges: {stats['edges']:,}")
    print(f"Number of components: {stats['components']}")
    print(f"Giant component size: {stats['giant_size']:,} "
          f"({stats['giant_size']/stats['nodes']:.4%} of nodes)")
    print(f"\nComputed theoretical giant fraction (  ): {breakdown['theta']:.6f}")

    print("\nTheoretical degree proportions within giant:")
    for k in range(1, len(breakdown['all_k_in_giant'])):
        prop = (breakdown['all_k_in_giant'][k] / 1
                if breakdown['theta'] > 0 else 0)
        print(f"  degree {k}: {prop:.4%}")

    print("\nEmpirical degree proportions within simulated giant:")
    for k, prop in empirical_props.items():
```

```
        print(f"  degree {k}: {prop:.4%}")

    print("\nOriginal q_k values:   ", original_q)
    print("Lifted   q_k values:   ", lifted_q)
    print(f"Sum of lifted q_k:     {lifted_q.sum():.6f}")
```

A.3 MonteCarlo code

```
import numpy as np
import matplotlib.pyplot as plt
from collections import Counter
from Bhamidi_solver import lift_q_above_theta2
from CHEM_Simulation import fix_and_split_proportions,

fast_linear_rule_constrained_config_model_igraph, tweak_labels

#           Fixed parameters
p1, p2, p3 = 0.5, 0.2, 0.3
seed = None

# compute the lifted q_k once
q1, q2, q3 = 0.24, 0.18, 0.28
# lift_q_above_theta2(p1, p2, p3)
theo_theta = q1 + q2 + q3
theo_q = {1: q1, 2: q2, 3: q3}

# range of n to test (log space from 1e3 to 1e5, 50 points)
n_values = np.logspace(3, 5, 100, dtype=int)

# how many independent simulations per n
T = 10

# storage for empirical averages
emp_theta = []
emp_qk = {1: [], 2: [], 3: []}

#           Simulation loop
for n in n_values:
    print(f"Now at n = {n}")
    theta_sum = 0.0
    q_sums = {1: 0.0, 2: 0.0, 3: 0.0}

    for _ in range(1,T+1):
        print(f"T = {_}")
        # build split using the same lifted q's
        split, _ = fix_and_split_proportions(p1, p2, p3, n, q1, q2, q3)
        node_labels = {}
        idx = 0
        for lbl, cnt in split.items():
            for _ in range(cnt):
                node_labels[idx] = lbl
                idx += 1

        rng = np.random.default_rng(seed)
        split, node_labels = tweak_labels(split, node_labels, rng)

        g, stats, giant = fast_linear_rule_constrained_config_model_igraph(node_labels)
```

```
        # empirical giant fraction
        theta_sum += stats['giant_size'] / n

        # empirical q_k = (# nodes of degree k in giant) / n
        degs = giant.degree()
        cnt = Counter(degs)
        for k in (1, 2, 3):
            q_sums[k] += cnt.get(k, 0) / n

    # record the averages
    emp_theta.append(theta_sum / T)
    for k in emp_qk:
        emp_qk[k].append(q_sums[k] / T)

#           Plot 1: Empirical vs theoretical
#           plotting with  y axis  scaled  1000  for extra precision
scale_factor = 1000

# Plot 1: Empirical vs Theoretical
plt.figure(figsize=(8,5))
plt.plot(n_values, np.array(emp_theta)*scale_factor, 'o-', label=f'empirical      {scale_factor
plt.hlines(theo_theta*scale_factor,
           xmin=n_values[0], xmax=n_values[-1],
           colors='r', linestyles='--', label=f'theoretical      {scale_factor}')
plt.xscale('log')
plt.xlabel('n (log scale)')
plt.ylabel(f'Giant fraction      {scale_factor}')
plt.title('Empirical vs Theoretical Giant Fraction')
plt.legend()
plt.grid(True, which='both', ls='--')
plt.tight_layout()

# Plot 2: Empirical vs theoretical q_k
plt.figure(figsize=(8,5))
for k, color in zip((1,2,3), ('C0','C1','C2')):
    plt.plot(n_values, np.array(emp_qk[k])*scale_factor, 'o-', color=color,
             label=f'empirical q_{k}   {scale_factor}')
    plt.hlines(theo_q[k]*scale_factor,
               xmin=n_values[0], xmax=n_values[-1],
               colors=color, linestyles='--', label=f'theoretical q_{k}   {scale_factor}')
plt.xscale('log')
plt.xlabel('n (log scale)')
plt.ylabel(f'Fraction of all nodes in giant with degree k   {scale_factor}')
plt.title('Empirical vs Theoretical q_k')
plt.legend()
plt.grid(True, which='both', ls='--')
plt.tight_layout()

plt.show()
```

A.4 Configuration model code

```
    import networkx as nx
import numpy as np
import matplotlib.pyplot as plt

def generate_config_model(n, p_list, degree_values=[1,2,3], seed=None):
    """
```

```
    Generate a simple configuration-model graph with a given degree distribution.
    """
    if seed is not None:
        np.random.seed(seed)
    p = np.array(p_list)
    if not np.isclose(p.sum(), 1):
        raise ValueError("Probabilities must sum to 1.")
    degrees = np.random.choice(degree_values, size=n, p=p)
    while degrees.sum() % 2 != 0:
        idx = np.random.randint(n)
        degrees[idx] = np.random.choice(degree_values, p=p)
    Gm = nx.configuration_model(degrees, seed=seed)
    G = nx.Graph()
    G.add_nodes_from(Gm.nodes())
    for u, v in Gm.edges():
        if u != v:
            G.add_edge(u, v)
    return G

def analyze_giant_component(G, n):
    """
    Return (prop_in_giant, degree_dist_within_giant).
    """
    comps = list(nx.connected_components(G))
    if not comps:
        return 0.0, {}
    giant = max(comps, key=len)
    size_gc = len(giant)
    prop_gc = size_gc / n
    degs = [d for _, d in G.subgraph(giant).degree()]
    uniq, cnt = np.unique(degs, return_counts=True)
    dist = {k: v/n for k, v in zip(uniq, cnt)}
    return prop_gc, dist

if __name__ == "__main__":
    # parameters
    p_list = [0.5, 0.2, 0.3]
    degree_values = [1, 2, 3]
    n_list = np.unique(np.logspace(3, 6, 12, dtype=int))
    T = 10
    seed = None

    # collect results
    all_gc_props = []
    all_deg_props  = {k: [] for k in degree_values}

    for n in n_list:
        gc_runs = []
        deg_runs = {k: [] for k in degree_values}
        for _ in range(T):
            G = generate_config_model(n, p_list, degree_values, seed)
            pgc, dd = analyze_giant_component(G, n)
            gc_runs.append(pgc)
            for k in degree_values:
                deg_runs[k].append(dd.get(k, 0.0))
        all_gc_props.append(gc_runs)
        for k in degree_values:
            all_deg_props[k].append(deg_runs[k])
```

```
    # compute weighted average weights
    weights = n_list / n_list.sum()

    # Plot 1: giant proportion vs n
    plt.figure()
    for i, n in enumerate(n_list):
        plt.scatter([n]*len(all_gc_props[i]), all_gc_props[i], alpha=0.6)
    mean_gc = [np.mean(x) for x in all_gc_props]
    line_gc, = plt.plot(n_list, mean_gc, 'o-', label="Mean prop in giant")
    wavg_gc = np.dot(mean_gc, weights)
    plt.hlines(wavg_gc, n_list[0], n_list[-1],
               colors=line_gc.get_color(), linestyles='--',
               label=f"Weighted avg = {wavg_gc:.3f}")
    plt.xscale('log')
    plt.xlabel('n')
    plt.ylabel('Proportion in Giant')
    plt.title('Giant Component Proportion vs n')
    plt.grid(True, which='both', linestyle='--', linewidth=0.5)
    plt.legend()
    plt.tight_layout()

    # Plot 2: degree distribution vs n with weighted averages
    plt.figure()
    for k in degree_values:
        means_k = [np.mean(r) for r in all_deg_props[k]]
        line_k, = plt.plot(n_list, means_k, 'o-', label=f"Degree {k}")
        wavg_k = np.dot(means_k, weights)
        plt.hlines(wavg_k, n_list[0], n_list[-1],
                   colors=line_k.get_color(), linestyles='--',
                   linewidth=1, label=f"Weighted avg deg {k} = {wavg_k:.3f}")
    plt.xscale('log')
    plt.xlabel('n')
    plt.ylabel('Degree Proportion in Giant')
    plt.title('Degree Distribution in Giant vs n')
    plt.grid(True, which='both', linestyle='--', linewidth=0.5)
    plt.legend()
    plt.tight_layout()

    plt.show()
```

A.5 Validation

A.5.1 Weighted averages

Figure 4.2 featured weighted averages for the empirical proportions; these were calculated in the following manner:

For each graph size n that was simulated, the average over ten simulations was calculated and weighted by a factor w_i

$$w_i = \frac{n_j}{\sum_{i \geq 1} n_j}. \tag{A.1}$$

A.5.2 Validation code

```
    #           Validity tests
emp_theta = np.array(emp_theta)

# 1) RMSE for giant proportion
giant_rmse = np.sqrt(np.mean((emp_theta - theo_theta)**2))
```

```
# 2) Pearson and goodnessp for giant proportion (df=1)
giant_chi2 = np.sum((emp_theta - theo_theta)**2 / (theo_theta + 1e-12))
giant_p    = chi2.cdf(giant_chi2, df=1)  # low = better fit

# 3) RMSE for q_k values
q_rmse = {}
# 4) Pearson and goodnessp for each q_k (df=1 each)
q_p    = {}
for k in (1,2,3):
    arr       = np.array(emp_qk[k])
    q_rmse[k] = np.sqrt(np.mean((arr - theo_q[k])**2))
    chi2_k    = np.sum((arr - theo_q[k])**2 / (theo_q[k] + 1e-12))
    q_p[k]    = chi2.cdf(chi2_k, df=1)

# 5) Composite GOF stays as before
avg_q_rmse    = np.mean(list(q_rmse.values()))
composite_gof = 0.5 * giant_rmse + 0.5 * avg_q_rmse

print("\nValidation Statistics (low p = good):")
print(f"  Giant RMSE:  {giant_rmse:.6f}")
print(f"  Giant goodnessp:  {giant_p:.6f}")
for k in (1,2,3):
    print(f"  q_{k} RMSE:  {q_rmse[k]:.6f}")
    print(f"  q_{k} goodnessp:  {q_p[k]:.6f}")
print(f"  Composite GOF:  {composite_gof:.6f}")
```