MEng Individual Project

Department of Computing

Imperial College of Science, Technology and Medicine

On overlapping community-based networks: generation, detection, and their applications

Author:
C.H. Bryan Liu

Supervisors:
Dr Marc P. Deisenroth
Benjamin P. Chamberlain

June 2016

Submitted in partial fulfillment of the requirements for the MEng in Mathematics and Computer Science of Imperial College London

Abstract

Individuals connect in social and information networks connect more frequently when they share similar characteristics - this is known as homophily, or more commonly "birds of a feather flock together". As a result of homophily people form communities, a group of people that have a particular characteristic in common. As a result, characteristics can be infered by discovering communities. Early approaches to community detection assumed non-overlapping communities. However as humans often demonstrate multiple affiliations, overlapping community structure is more appropriate for modelling social networks. The study of overlapping community structures in social networks allows us to understand an individual's multiple traits from their connections with other individuals, leading to applications in finance, marketing and social science research. In this report, we investigate the generation of networks with overlapping communities and the detection of overlapping communities within.

Several graph generative models (e.g. Erdős and Rényi (1959) [16], Watts and Strogatz (1998) [48], Barabási and Albert (1999) [5], Holland et al. (1983) [25], Yang and Leskovec (2012) [53]) were proposed for real networks, yet they all lack the seemingly invariant properties: an overlapping community structure *and* the scale-free property (i.e. power-law distributed node degrees). Based on our observation that individuals who are more influential, engaged and participate more gain more connections within a community, we propose a new overlapping community-based graph generative model. We show our model, one of the first ever proposed, is capable of generating graphs with the scale-free property based on overlapping communities with power-law distributed sizes.

We then apply our graph generative process to extending the verification framework for the Cluster Affiliation Model for Big Networks (BigClam) proposed by Yang and Leskovec (2013) [54], a popular overlapping community detection algorithm. The framework accounts for extra network characteristics (e.g. the degree of overlap between communities, dispersion of community sizes), and can be applied to other overlapping community detection algorithms as is. We identify the three stages of BigClam and obtain their apparent runtime complexity. We also notice the final stage dominates BigClam's runtime for networks with a large number of communities, yet its computation is not parallelised across CPU threads. By parallelising the stage's computation, we speed up runtime by as much as 430%.

Acknowledgments

I would like to thank:

- Benjamin Chamberlain and Dr Marc Deisenroth for their invaluable guidance for the entire project,
- Dr Luis Muñoz González for his input during the initial stage of the project,
- Staff and students I have met during the past four years - I am afraid I am unable to thank each and every of them individually here, as it would probably add another 40 pages to the report,
- My sister, Nicole, for upholding the rules and constructs of the English Language in my report, and finally,
- My father, mother, and grandmother for their continuous love and support throughout the years, even though we are thousands of miles apart.

Contents

Chapter 1

Introduction

> *"Communities: A group of people living in the same place or having a particular characteristic in common." (Oxford English Dictionary, 2016) [41]*

This project aims to present techniques to overcome challenges in extracting information from social and information networks with an overlapping community structure. More specifically, we attempt to address the question, "Given interactions between individuals within a network, can we locate all (possibly overlapping) communities and their members in a better way?"

1.1 Motivation

Interactions between individuals, objects, and concepts have been observed since antiquity. They are generally studied via networks (graphs), where different entities are represented as nodes and interactions between entities are represented as edges (figure 1.1 provides an illustration of a social network featuring individual-individual interaction), with modern research into social networks beginning as early as the 1930s (Freeman, 2004) [18].

1.1.1 Community Detection

Networks representing real, complex systems demonstrate a few common characteristics, one being the community structure in networks as identified by Girvan and Newman (2002) [20]. The study of communities within networks remains a young research field, have only gained popularity in the past ten to fifteen years.

One of the main research areas on networks centres on community detection: given a network, are we able to locate communities within, and infer characteristics of an individual based on their community affiliations? The research problem is made more difficult as Fortunato (2010) points out that there is no universally accepted definition of a community [17]. A general

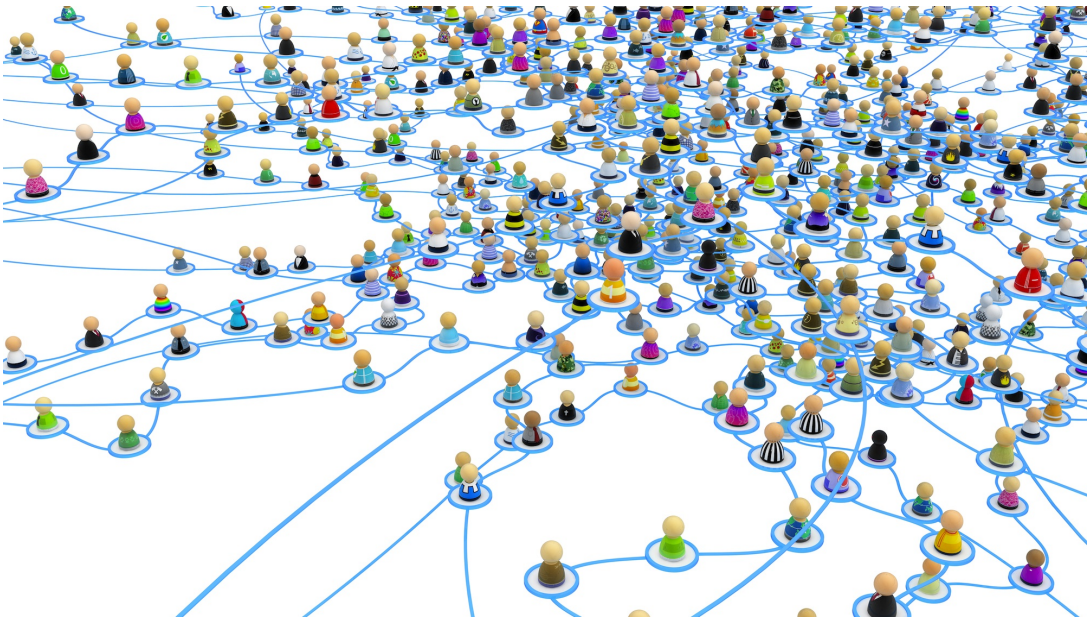


Figure 1.1: Illustration of a social network (individual-individual interaction).

guideline provided by Whang et al. (2013) defines community in the graph context as "a set of cohesive vertices that have more connections inside the set than outside" [50], though a wide range of interpretations of a community in a social context exist - it can be a user-created group on social networks (Yang and Leskovec, 2012) [53], or individuals separated by race or geographical location (Gleiser and Danon, 2003) [22].

We are interested in community detection due to its wide-range application and impact:

- Finance: Positive correlations are found between prices of different energy stocks
- Biomedical research: Interaction between proteins is more prominent in the same functional module
- Social networks: People are more likely to make friends with people in the same institution than those outside
- Recommender systems: Customers are likely to purchase goods from the same group of products

If stocks, proteins, people and goods are treated as entities and industries, functional modules, institutions and groups of products are treated as communities, a good solution for community detection can allow one to gain in financial markets, cure diseases, understand friendships, and increase sales simultaneously.

Relationship with Clustering

Cohen et al. (1997) showed that all finite graphs can be embedded in a three-dimentional Euclidean space $\mathbb{R}^3$ [13]. This implies detecting communities within a graph is equivalent to

detecting clusters within a Euclidean space, and thus advances in community detection can be easily translated to solutions to clustering problems and vice versa.

1.1.2 Overlapping Communities

We then consider the notion of overlapping communities, under which an entity is allowed, and usually assumed, to have multiple community affiliations. A university student may be affiliated to academics-based communities (e.g. the groups of Mathematics and Computer Science students), interest-based communities (e.g. a symphony orchestra, a cycling team, a London transport meetup group), and culture-based communities all at the same time.

Furthermore, Yang and Leskovec (2013) summarised observations on positive correlations between individual-individual interactions and community membership of said individuals:

> "[N]odes in the overlaps will have higher chance of being connected because they belong to multiple communities. [...] For example, people sharing multiple hobbies or belonging to several common institutions have a higher chance of becoming friends, researchers with many common interests are more likely to work together, and proteins involved in multiple common functional modules are more likely to interact." [54]

The empirical observations lead to in-depth study of overlapping community models and detection algorithms, with various approaches proposed (see Fortunato (2010) [17], who dedicated a section solely to overlapping community detection algorithms).

Ability to recover overlapping communities would allow one to gain more in-depth knowledge of each entity, having obtained all their community affiliations instead of one (possibly the most prominent) affiliation. However, overlapping community detection is considered more difficult than non-overlapping community detection, as one is required to infer multiple community affiliations for every individual with satisfactory accuracy.

Relationship with Feature Allocation

Recently Broderick et al. (2013) generalised the clustering problem, allowing each data point to be allocated an arbitrary number of features - groups that demonstrate common characteristics [9]. Combining the generalisation with the result on graph embedding by Cohen et al. (1997) (see section 1.1.1), we observe the strong link between the fields of overlapping community detection and feature allocation - with suitable adaptation, problems in both fields are equivalent. We will focus on overlapping community detection throughout the work.

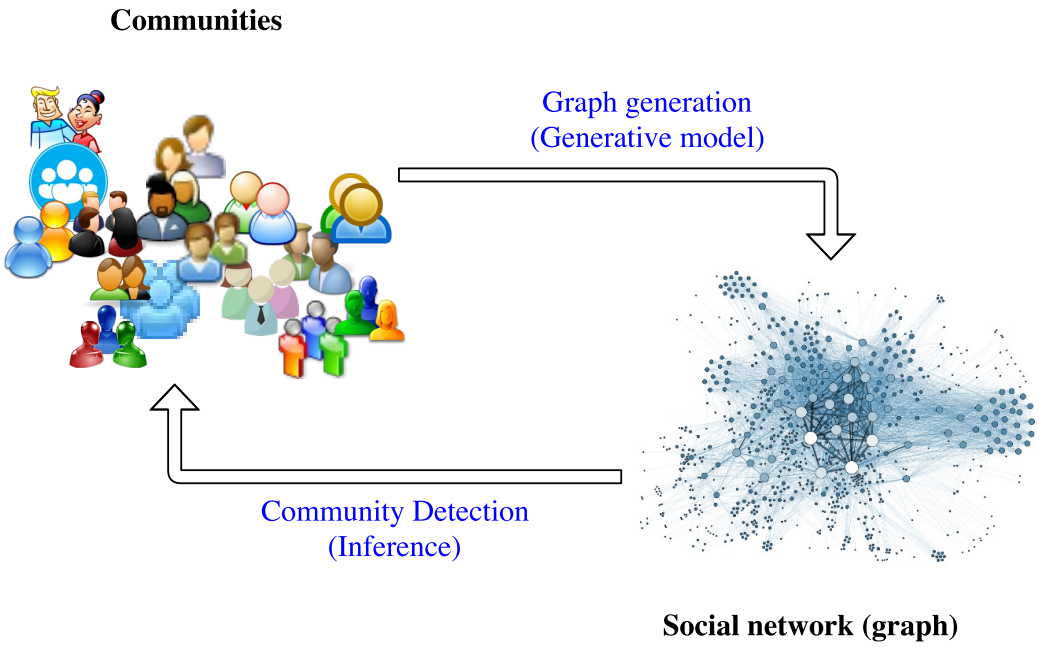


Figure 1.2: Illustration of the relationship between (overlapping) community-based graph generative models and (overlapping) community detection, both of which are the reverse process for the each other.

1.1.3 Graph Generation

We also approach the problem from the opposite direction, asking the question, "Given a set of overlapping communities and their members, can we generate interactions between individuals that resemble what is observed in real networks?"

The ability to generate real-like networks allows us to understand the underlying mechanism of society - how society works behind the scenes. Just like how models in physics try to replicate everyday life phenomena, if we are able to replicate features of real networks, we are one step closer to understanding motivating factors behind individual interactions. Moreover, developing a graph generative model can lead to the learning of latent features in a graph via Bayesian inference methods, which is equivalent to our original problem (see figure 1.2).

Network generation has fascinated generations of researchers, with one of the first contemporary models proposed by Erdős and Rényi (1959) [16]. The problem has been attempted by researchers including (but not limited to) Watts and Strogatz (1998) [48], Barabási and Albert (1999) [5], Holland et al. (1983) [25], and Yang and Leskovec (2012) [53]. Each of their work chip away less desirable properties in the models proposed by their predecessors, address extra features found in networks, and sometimes open up a new branch of study.

1.2 Objectives

The projects aims to obtain insights into some or all of the following open academic and commercial research problems on network generation and overlapping community detection:

1.2.1 A graph generative model based on overlapping communities

The ability to generate graphs that resemble real networks allows us to understand social dynamics and inherent properties of society, and to quantitatively evaluate network-related algorithms.

We aim to locate and/or develop a good graph generative model which is capable of generating graphs with the following characteristics:

- Scale-free property: node degrees following a power-law distribution
- Overlapping community structure

1.2.2 Existence of a good community model and detection algorithm

We interpret a good community model and detection algorithm to be that satisfying the following three properties:

- Able to detect overlapping communities accurately
- Able to scale with increasing network size
- Easy to interpret and be explained to laymen

Most of the state-of-the-art community detection algorithms satisfy only two out of three properties above, which is not ideal. We aim to locate and/or develop a good community model and detection algorithm, and evaluate the model and algorithm's fitness by applying it to real and synthetic networks.

1.3 Contribution

We summarise our contribution through this investigation. In the field of network generation, we:

- Showed the Community-Affiliation Graph Model (AGM) by Yang and Leskovec (2012) [53] exhibits properties considered undesirable for real networks
- Observed influential, engaged individuals with a high participation rate gain more connections within their communities

- Developed an overlapping community-based graph generative model, which accommodates both an overlapping community structure and the scale-free property (node degrees following a power-law distribution)

In the field of overlapping community detection, we:

- Extended the verification framework for the accuracy and scalability of the Cluster Affiliation Model for Big Networks (BigClam) by Yang and Leskovec (2013) [54], and overlapping community detection algorithms in general, taking various network characteristics into account
- Obtained the apparent runtime complexity for BigClam with the verification framework
- Sped up the dominating stage of BigClam by up to 430% and the overall runtime by 150%, hence saving hours of time spent waiting for results

Chapter 2

Background

We include a set of required preliminary knowledge and notation/nomenclature used throughout the investigation. We assume readers are familiar with basic concepts in linear algebra and statistics.

The chapter is organised as follows. Section 2.1 defines the extensions on basic concepts in vectors and matrices. We introduce concepts related to and nomenclature used in graph and network theory in section 2.2. Less common probability distributions used throughout the investigation are outlined in section 2.3, and an overview of a t-test and normal linear models are detailed in sections 2.4 and 2.5 respectively. We conclude the chapter with concepts related to computational optimisation in section 2.6.

2.1 Vectors and Matrices

We first introduce the notion of non-negative vectors and matrices, which are used to model the strength of an individual's affiliation to a community in chapters 4 and 5.

Definition 2.1. (Non-negative vectors and matrices)

A vector $\boldsymbol{v} = (v_i) \in \mathbb{R}^n$ is non-negative if all entries of $\boldsymbol{v}$ are non-negative.
(i.e. $v_i \geq 0 \quad \forall\, i \in \{1, ..., n\}$)

Similarly, A matrix $F = (F_{ij}) \in \mathbb{R}^{n \times m}$ is non-negative if all entries of F are non-negative.
(i.e. $F_{ij} \geq 0 \quad \forall\, i \in \{1, ..., n\},\ j \in \{1, ..., m\}$)

2.2 Graph and Network Theory

We then give a few definitions on graph and network theory used throughout our investigation.

Definition 2.2. (Adjacency matrix for an undirected graph)

Let $G = G(V, E)$ be an undirected graph, a matrix $A = (A_{ij}) \in \mathbb{R}^{|V|\times|V|}$ $(i, j \in V)$ is called an adjacency matrix with respect to G if

$$A_{ij} = \begin{cases} 1 & \text{If } (i,j) \in E \\ 0 & \text{Otherwise} \end{cases} . \tag{2.1}$$

Definition 2.3. (Conductance of a subset of nodes for an undirected graph)

Let $G = G(V, E)$ be an undirected graph, $A = (A_{ij})$ be an adjacency matrix with respect to G. Moreover let $S \subseteq V$ be a subset of nodes in the graph, and $\bar{S} = V \setminus S$ be all other nodes in the graph.

The conductance of S is

$$\varphi(S) = \frac{\sum_{i\in S,\, j\notin S} A_{ij}}{\min(\mathcal{A}(S), \mathcal{A}(\bar{S}))} , \tag{2.2}$$

where $\mathcal{A}(S) = \sum_{i\in S} \sum_{j\in V} A_{ij}$

In general, $\varphi(S)$ could be interpreted as the proportion of (undirected) edges from nodes within S to nodes outside S.

Definition 2.4. (Neighbours, neighbourhood and locally minimal neighbourhood)

Let $G = G(V, E)$ be a graph.

The neighbours $\mathcal{N}(u)$ of node $u \in V$ is the set of nodes which shares an edge with u under G (i.e. $\mathcal{N}(u) = \{v : v \in V,\, (u, v) \in E\}$).

A neighbourhood $N(u)$ of node $u \in V$ is the set containing u and u's immediate neighbours in G (i.e. $N(u) = \{u\} \cup \mathcal{N}(u)$).

A neighbourhood $N(u)$ is said to be locally minimal if it has lower conductance than all $N(v)$, $v \in \mathcal{N}(u)$ (i.e. $\varphi(N(u)) \leq \varphi(N(v)) \; \forall\, v \in \mathcal{N}(u)$).

Definition 2.5. (Nomenclature on "network")

A network is a collection of the following:

- The set of individuals (denoted V)
- The set of communities (denoted C)
- The set of community memberships/ individual affiliations between individuals and communities (denoted M)

- The set of connections between individuals (edges) (denoted E)

and any attributes or information associated with, or derivable from any combination of the sets above.

In graph theory terms, a network is a collection of the visible units, the latent (hidden) units, the latent-visible links, and the visible-visible links on a graph, plus any information derivable from the units and connections. Figure 2.1 shows a small network with four individuals (visible units), two communities (hidden units), six memberships/ affiliations (latent-visible links), and five edges between individuals (visible-visible link).

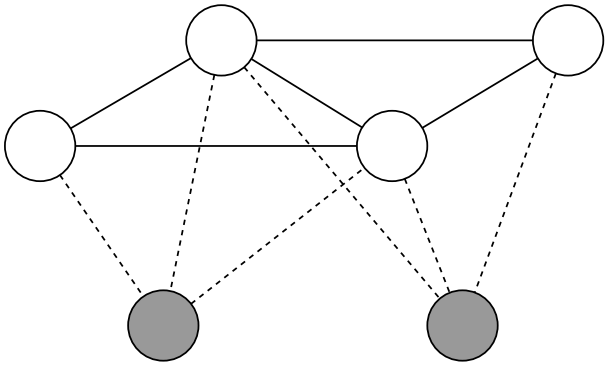


Figure 2.1: A small network with four individuals (white circles), two communities (grey circles), six memberships/ affiliations (dashed lines), and five edges between individuals (solid lines).

2.3 Probability Distributions

The investigation also relies on the following distributional results. They include power-law distributions, the Pareto-family distributions, the Poisson's binomial distribution, and the mixed Poisson distribution.

2.3.1 Power Law Distribution

Definition 2.6. (Power law distributions)

Let X be a random variable with positive real number support. X is said to follow a power law distribution with shape parameter α if the probability density function (pdf) (or probability mass function (pmf) for discrete X) satisfies the following:

$$f_X(x) \propto x^{-\alpha}, \quad x \in \mathbb{R}^+ \, . \tag{2.3}$$

Furthermore, we say X asymptotically follows a power law distribution with shape parameter

α if its pdf/ pmf satisfies the following:

$$f_X(x) \propto L(x)x^{-\alpha} \,, \tag{2.4}$$

where L is some slowly varying function (i.e. $\lim_{x\to\infty} \frac{L(cx)}{L(x)} = 1 \quad \forall c > 0$).

Theorem 2.7. *(Scale-invariant property of power law distributions)*

Power law distributions are scale invariant - the shape of the associated probability density function (pdf) remains unchanged upon scaling the argument by a constant factor.

Proof. Given a power law-distributed random variable X, with associated pdf $f_X(x) \propto x^{-\alpha}$. A scalar transformation on the pdf's argument $x \mapsto cx$ yields:

$$f_X(cx) \propto (cx)^{-\alpha} = c^{-\alpha}x^{-\alpha} \propto x^{-\alpha} \,, \tag{2.5}$$

which is a power law distribution with the same shape parameter.

For a asymptotically power law-distributed random variable X with associated pdf $f_X(x) \propto L(x)x^{-\alpha}$. A scalar transformation on the pdf's argument $x \mapsto cx$ yields:

$$f_X(cx) \propto L(cx)(cx)^{-\alpha} \to L(x)c^{-\alpha}x^{-\alpha} \propto L(x)x^{-\alpha}, \quad x \to \infty \,, \tag{2.6}$$

which is asymptotically power law-distributed with the same shape parameter. □

Example 2.8. (Scale-free network)

A *scale-free network* is a network in which the degree of each node, K, follows a power law distribution:

$$Pr(K = k) = \frac{k^{-s}}{\zeta(s)}, \quad k \in \mathbb{Z}^+ = \{1, 2, ..., \} \,, \tag{2.7}$$

where $\zeta(s) = \sum_{k=1}^{\infty} k^{-s}$ is the Riemann zeta function.

2.3.2 Pareto Distributions

Pareto Distribution (more unambiguously *Type I Pareto Distribution* according to Arnold (2014) [4]) is a continuous distribution which follows the power law. It can be generalised to a hierarchy of Pareto models to improve fitting of data, especially towards the tail end of the distribution. We focus on two types of Pareto distributions in this chapter.

Definition 2.9. (Type I Pareto Distribution)

A random variable X follows *Type I Pareto Distribution* if it follows the following cumulative distribution function (cdf) and probability distribution function (pdf):

$$F_X(x) = Pr(X \leq x) = 1 - \left(\frac{x}{\sigma}\right)^{-\alpha}, \quad x > \sigma \,, \tag{2.8}$$

$$f_X(x) = \alpha\sigma^{\alpha}x^{-(\alpha+1)} \propto x^{-(\alpha+1)}, \quad x > \sigma \,, \tag{2.9}$$

where $\sigma > 0$ is the scale parameter and $\alpha > 0$ is the shape parameter. We denote $X \sim Pareto(I)(\sigma, \alpha)$ if X satisfies the above.

Remark. Type I Pareto Distribution is also known as the *classical Pareto model*. Note σ determines both the scale and location (minimum attainable value) of a Type I Pareto random variable.

Definition 2.10. (Type II Pareto Distribution)

A random variable X follows *Type II Pareto Distribution* if it follows the following cdf and pdf:

$$F_X(x) = 1 - \left(1 + \frac{x-\mu}{\sigma}\right)^{-\alpha}, \quad x > \mu \,, \tag{2.10}$$

$$f_X(x) = \frac{\alpha}{\sigma}\left(1 + \frac{x-\mu}{\sigma}\right)^{-(\alpha+1)}, \quad x > \mu \,, \tag{2.11}$$

where $\mu \in \mathbb{R}$ is the location parameter, $\sigma > 0$ is the scale parameter and $\alpha > 0$ is the shape parameter. We denote $X \sim Pareto(II)(\mu, \sigma, \alpha)$ if a random variable X satisfies the above.

Remark. Unlike Type I Pareto distribution, in which the minimum attainable value of the distribution is fixed after the degree of scaling is determined (or vice versa), Type II Pareto distribution separates the role of scale and location parameters, and hence is more flexible as it allows one to specify both the minimum attainable value and the degree of scaling.

Type II Pareto distribution demonstrates the following property:

Theorem 2.11. *Type II Pareto Distribution asymptotically follows the power law.*

Proof. Notice the pdf of Type II Pareto can be expressed as:

$$\begin{aligned} f_X(x) &= \frac{\alpha}{\sigma}\left(1 + \frac{x-\mu}{\sigma}\right)^{-(\alpha+1)} = \frac{\alpha}{\sigma}\left[\frac{x}{\sigma}\left(\frac{\sigma}{x} + 1 - \frac{\mu}{x}\right)\right]^{-(\alpha+1)} \\ &= \frac{\alpha}{\sigma}\left(\frac{x}{\sigma}\right)^{-(\alpha+1)}\left(\frac{\sigma-\mu}{x} + 1\right)^{-(\alpha+1)} \\ &\propto \underbrace{\left(\frac{\sigma-\mu}{x} + 1\right)^{-(\alpha+1)}}_{L(x)} x^{-(\alpha+1)} \,, \end{aligned} \tag{2.12}$$

where $L(x)$ is a slowly varying function:

$$\lim_{x\to\infty}\frac{L(cx)}{L(x)}=\lim_{x\to\infty}\frac{\left(\frac{\sigma-\mu}{cx}+1\right)^{-(\alpha+1)}}{\left(\frac{\sigma-\mu}{x}+1\right)^{-(\alpha+1)}}=\frac{(0+1)^{-(\alpha+1)}}{(0+1)^{-(\alpha+1)}}=1\ . \tag{2.13}$$

By definition 2.6, Type II Pareto distribution follows the power law asymptotically. □

We also explore the relationship between Types I and II Pareto distributions:

Theorem 2.12. *(Transformation of Pareto random variables)*

Let $X\sim Pareto(I)(\sigma,\alpha)$ be a Type I Pareto random variable, the transformed random variable $Y=\frac{\sigma'}{\sigma}(X-\sigma)+\mu'$ follows a Type II Pareto distribution (i.e. $Y\sim Pareto(II)(\mu',\sigma',\alpha)$):

Proof. The cdf for Y is

$$\begin{aligned}F_Y(y)&=Pr(Y\le y)=Pr\left(\frac{\sigma'}{\sigma}(X-\sigma)+\mu'\le y\right)\\&=Pr\left(X\le\frac{\sigma}{\sigma'}(y-\mu')+\sigma\right)=F_X\left(\frac{\sigma}{\sigma'}(y-\mu')+\sigma\right)\\&=1-\left(\frac{\frac{\sigma}{\sigma'}(y-\mu')+\sigma}{\sigma}\right)^{-\alpha}\\&=1-\left(1+\frac{y-\mu'}{\sigma'}\right)^{-\alpha},\end{aligned} \tag{2.14}$$

which, by equation 2.10, shows $Y\sim Pareto(II)(\mu',\sigma',\alpha)$. □

We will use Type II Pareto distribution (with rounding to give a discrete distribution) to describe the distribution of the number of members in a community for communities in a network.

We finally define a Type IV Pareto distribution for completeness.

Definition 2.13. (Type IV Pareto Distribution)

A random variable X follows *Type IV Pareto Distribution* if it follows the following cdf:

$$F_X(x)=1-\left(1+\left(\frac{x-\mu}{\sigma}\right)^{\frac{1}{\gamma}}\right)^{-\alpha},\quad x>\mu\ , \tag{2.15}$$

where $\mu\in\mathbb{R}$ is the location parameter, $\sigma>0$ is the scale parameter, $\gamma>0$ is the inequality parameter and $\alpha>0$ is the shape parameter. We denote $X\sim Pareto(IV)(\mu,\sigma,\gamma,\alpha)$ if a random variable X satisfies the above.

Remark. A Type III Pareto distribution is a Type IV Pareto distribution with α fixed at one, i.e. $Pareto(III)(\mu, \sigma, \gamma) \equiv Pareto(IV)(\mu, \sigma, \gamma, 1)$

2.3.3 Poisson's Binomial Distribution

The Poisson's binomial distribution (Poisson binomial distribution, or Poisson-binomial distribution in some literature) is an extension of the binomial distribution. According to Wang (1993), it was first considered by Siméon Poisson [47], and models the probability of achieving k successes in n independent trials with *different* success probabilities.

The distribution is mathematically defined by Wang (1993) [47] as follows.

Definition 2.14. Let $n \in \mathbb{Z}^+$ and $k \in \{0, 1, ..., n\}$, the set $\mathcal{A}_k$ is defined as:

$$\mathcal{A}_k = \{A |\ A \subseteq \{1, ..., n\}, |A| = k\}\ , \tag{2.16}$$

where $|\cdot|$ denotes the cardinality function.

Remark. In other words, $\mathcal{A}_k$ is the subset of the power set of $\{1, ..., n\}$ with cardinality k. We can interpret an element of $\mathcal{A}_k$ as the possible indices of success events which lead to k successes in total.

Definition 2.15. (PDF of Poisson's binomial distribution)

A random variable X follows the Poisson's binomial distribution with n trials and success probabilities $p_1, p_2, ..., p_n$ if it follows the following probability density function (pdf):

$$\mathbb{P}(X = k) = \sum_{A \in \mathcal{A}_k} \prod_{i \in A} p_i \prod_{j \in A^c} (1 - p_j)\ , \tag{2.17}$$

where $\mathcal{A}_k$ is that defined in definition 2.14, and $A^c = \{1, 2, ..., n\} \setminus A$.

The probability can be interpreted as follows. Given k successes, we consider all possible combinations of k event indices ($\mathcal{A}_k$). For each possible combination, we multiply the success probability of events with indices included in the combination with the failure probability of events with indices not included in the combination, which is possible due to independence between trials. We then sum up the probability of all possible combinations to obtain the probability of k successes by the law of total probability.

Remark. If $p_i = p\ \forall i$, then X follows a standard binomial distribution: $\mathbb{P}(X = k) = \binom{n}{k} p^k (1 - p)^{n-k}$.

2.3.4 Mixed Poisson Distribution

The mixed Poisson distribution is the resultant distribution of a mixture of Poisson distributions with different rate parameters λ_i.

For a finite number of mixture components with parameters $\lambda_1, \lambda_2, ..., \lambda_n$, the distribution is defined as follows:

Definition 2.16. (PMF of mixed Poisson distribution with finite mixture components)

Let X be a random variable, X is said to follow the mixed Poisson distribution with a finite number of mixture components (with parameters $\lambda_1, \lambda_2, ..., \lambda_n$) if it follows the following probability mass function (PMF):

$$\mathbb{P}(X = x) = \frac{1}{n}\left(\frac{e^{-\lambda_1}{\lambda_1}^x}{x!} + \frac{e^{-\lambda_2}{\lambda_2}^x}{x!} + ... + \frac{e^{-\lambda_n}{\lambda_n}^x}{x!}\right) \tag{2.18}$$

Remark. The probability mass of the mixed Poisson distribution at x is the sum of the probability mass of the first, second, ..., n^{th} mixture component at x, normalised by $\frac{1}{n}$ as each mixture component is a probability distribution and hence sums to one.

In the case where the number of mixture components is infinite, and their corresponding parameter follows a probability distribution, Karlis and Xekalaki (2005) give the following probability mass function for the mixed Poisson distribution [27].

Definition 2.17. (PMF of mixed Poisson distribution with infinite mixture components)

Let $g(\lambda)$ be a probability density function (PDF), and X be a random variable. X is said to follow a mixed Poisson distribution with infinite mixture components, with mixture component parameters λ following the distribution characterised by the PDF $g(\lambda)$, if it follows the following probability mass function (PMF):

$$\mathbb{P}(X = x) = \int_0^\infty \frac{e^{-\lambda}\lambda^x}{x!} g(\lambda)\,\mathrm{d}\lambda \tag{2.19}$$

Remark. The PMF is already normalised as $g(\lambda)$ is a PDF, and hence integrates to one.

2.4 Welch's t-test

We introduce the Welch's t-test, proposed by Welch (1947) as a more robust alternative to the Student's t-test under two groups of samples with different variances and sample sizes [49]. It is the default t-test used by the statistical package `R`.

The test is based on the Welch's t-statistic defined as follows.

Definition 2.18. (Welch's t-statistic)

Given two independent groups of samples, the first (second) group with sample mean $\bar{X}_1$ ($\bar{X}_2$), sample variance s_1^2 (s_2^2), and sample size N_1 (N_2) respectively. The Welch's t-statistic is defined as:

$$t = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{s_1^2}{N_1} + \frac{s_2^2}{N_2}}} \,. \tag{2.20}$$

Theorem 2.19. *t defined in definition 2.18 follows a Student's t-distribution, i.e.*

$$t \sim t_\nu \,, \tag{2.21}$$

$$\nu \approx \frac{\left(\frac{s_1^2}{N_1} + \frac{s_2^2}{N_2}\right)^2}{\frac{s_1^4}{N_1^2(N_1-1)} + \frac{s_2^4}{N_2^2(N_2-1)}} \,, \tag{2.22}$$

where ν denotes the degrees of freedom associated with the t-distribution.

Proof. See Welch (1947) [49]. □

The Welch's t-statistic motivates a hypothesis test to see if the underlying true mean of the two groups are equal. If the corresponding p-value for the calculated Welch's t-statistic is lower than α, then we reject the null hypothesis at a α significance level, and accept the alternate hypothesis (that the true difference in mean is not equal to zero in a two-tailed test, or greater/less than zero in a one-tailed test).

2.5 Normal Linear Models

Consider a dataset with n data points, in which data point i contains a random, observed result of interest Y_i, and p deterministic, observed covariates $x_{ij},\ j \in \{1, ..., n\}$. According to Lau (2016) [30], the value of Y_i can be modelled as a linear combination of x_{ij} and a set of unknown constant parameters, plus a stochastic error:

Definition 2.20. (Normal Linear Models)

A linear model is defined as:

$$\boldsymbol{Y} = X\boldsymbol{\beta} + \boldsymbol{\epsilon} \tag{2.23}$$

$$\iff Y_i = x_{i1}\beta_1 + x_{i2}\beta_2 + ... + x_{ip}\beta_p + \epsilon_i \quad \forall i \in \{1, ..., n\} \,, \tag{2.24}$$

where

- $\boldsymbol{Y} = (Y_1, ..., Y_n)^T \in \mathbb{R}^n$ is the random vector of observations, called the response.
- $X = (x_{ij}) \in \mathbb{R}^{n \times p}$ is the design matrix (known) which contains the covariates (predictors). Let $r := \text{rank}(X)$.
- $\boldsymbol{\beta} = (\beta_1, ..., \beta_p)^T \in \mathbb{R}^p$ is the unknown parameter vector.
- $\boldsymbol{\epsilon} = (\epsilon_1, ..., \epsilon_n)^T \in \mathbb{R}^n$ is the unobserved error vector such that $\mathbb{E}(\boldsymbol{\epsilon}) = 0$.

Furthermore, if $\boldsymbol{\epsilon} \sim N(0, \sigma^2 I_n)$ for some $\sigma^2 > 0$ (i.e. observation errors are uncorrelated and have the same variance), then the linear model is normal. The normal linear model can be written as

$$\boldsymbol{Y} \sim N(X\boldsymbol{\beta}, \sigma^2 I_n) \ , \tag{2.25}$$

where I_n is the $n \times n$ identity matrix.

We present below some key results and their preliminaries in normal linear models, which will be used throughout our investigation. The results are drawn and adapted from the lectures given by Lau (2016) [30]. Proofs for cited results are omitted here, and can be found in most of the textbooks on normal linear models.

Theorem 2.21. *(Maximum likelihood estimator for unknown parameter vector)*

Assume full rank for $X \in \mathbb{R}^p$ (i.e. rank$(X) = p$), The maximum likelihood/ least squares estimator for the unknown parameter vector $\boldsymbol{\beta}$ in the model defined in definition 2.20 is given by

$$\hat{\boldsymbol{\beta}} = (X^T X)^{-1} X^T \boldsymbol{Y} \ . \tag{2.26}$$

Definition 2.22. (Fitted values) The vector of fitted value is given as $\hat{\boldsymbol{Y}} = X\hat{\boldsymbol{\beta}}$, where $\hat{\boldsymbol{\beta}}$ is that introduced in theorem 2.21.

Definition 2.23. (Residuals) The vector of residuals is given as $\boldsymbol{e} = \boldsymbol{Y} - \hat{\boldsymbol{Y}}$.

Definition 2.24. (Residual sum of squares) The residual sum of squares is defined as RSS $= \boldsymbol{e}^T \boldsymbol{e}$, where $\boldsymbol{e}$ is that defined in definition 2.23.

Theorem 2.25. *(Distribution of least square linear estimator)*

Assume a linear model setting as defined in definition 2.20, and full rank for X. For any $\boldsymbol{c} \in \mathbb{R}^p$:

$$\frac{\boldsymbol{c}^T\hat{\boldsymbol{\beta}} - \boldsymbol{c}^T\boldsymbol{\beta}}{\sqrt{\boldsymbol{c}^T(X^TX)^{-1}\boldsymbol{c}\frac{RSS}{n-p}}} \sim t_{n-p} \ , \tag{2.27}$$

the student-t distribution with $n - p$ degrees of freedom.

Remark. The result allows construction of hypothesis tests to determine if a component of the least squares estimator $\hat{\boldsymbol{\beta}}$ should be zero.

Theorem 2.26. *(Distribution of residual sum of squares under different linear models)*

Let $X \in \mathbb{R}^{n\times r}$ and $X_0 \in \mathbb{R}^{n\times s}$, $p > s$ be two design matrices such that span$(X_0) \subset$ span(X), and assume two linear models, $\boldsymbol{Y} = X\boldsymbol{\beta} + \boldsymbol{\epsilon}$ and $\boldsymbol{Y} = X_0\boldsymbol{\beta} + \boldsymbol{\epsilon}$ (i.e. the latter is a sub-model of the former). Let RSS be the residual sum of squares in the full model, and RSS_0 be the residual sum of squares in the sub-model. Then

$$\frac{RSS_0 - RSS}{RSS}\,\frac{n-r}{r-s} \sim F_{r-s,n-r} \ , \tag{2.28}$$

the F-distribution with degrees of freedom $r - s$ and $n - r$.

Remark. The result allows construction of hypothesis tests to determine if the full model would bring significant improvements in RSS compared to a sub-model.

2.6 Optimisation

We conclude by including some topics in computational optimisation, including convex optimisation problems and associated results in optimality, as well as the Newton's Method of finding the optimal point/root of a function.

2.6.1 Convex Optimisation

Consider the following convex optimisation problem (a subset of optimisation problem):

$$\min_x f(x) \tag{2.29}$$

$$\text{s.t. } x \in S \ , \tag{2.30}$$

where f is a convex function on S defined in definition 2.28, and S is a convex (constraint) set defined in definition 2.27.

Definition 2.27. (Convex set)

A set $S \subset \mathbb{R}^n$ is said to be convex if every point on the line connecting any two points $x, y \in S$ is itself in S, i.e.

$$x, y \in S \implies \alpha x + (1-\alpha)y \in S \quad \forall \alpha \in [0,1] \, . \tag{2.31}$$

Definition 2.28. (Convex function)

Let $f : S \to \mathbb{R}$ be a function defined on a convex set $S \subset \mathbb{R}^n$, then f is convex on S if the line segment connecting $f(x)$ and $f(y)$ at any two points $x, y \in S$ lies above the function between x and y, i.e.

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)y \quad \forall \alpha \in [0,1] \, . \tag{2.32}$$

Chong and Zak (2013) showed any local minimiser of the convex optimisation problem is a global minimiser of the convex optimisation [12].

2.6.2 Newton's Method

Newton's Method (also known as the Newton-Raphson Method) is a iterative method for finding good approximations of a maximum/minimum of a function. It has since been adapted to find the root of a function.

Chong and Żak (2013) [12] listed the iterative formula to find the optimum $\arg\min f(x)/\arg\max f(x)$:

$$x^{(k+1)} = x^{(k)} - \frac{f'(x^{(k)})}{f''(x^{(k)})}, \quad k \geq 0 \, , \tag{2.33}$$

where f' and f'' is the first and second derivatives of $f \in \mathcal{C}^2$, a function which is at least twice-differentiable, respectively.

Equation 2.33 motivates the use of Newton's algorithm to find the root of a function. Notice a necessary condition for f to attain a (local) maximum/minimum is $f'(x) = 0$. Hence the equation can be used to find the root of g if we let $g = f'$ and use the following iterative formula:

$$x^{(k+1)} = x^{(k)} - \frac{g(x^{(k)})}{g'(x^{(k)})}, \quad k \geq 0 \, , \tag{2.34}$$

where g' is the derivative of $g \in \mathcal{C}^1$, a function which is at least once-differentiable, and $x^{(0)}$ is

the initial point for Newton's Method.

As discussed by Chong and Żak (2013) [12], Misener (2016) [36] and Parpas (2016) [42], Newton's method may not converge (it might diverge or cycle), though for functions which are convex/concave in its domain, or initial points sufficiently close to the solution, the method converges quickly at a quadratic rate.

Chapter 3

Related Work

In this chapter we present a number of established work/state-of-the-art in the fields of graph generation and community detection. We describe four graph generative models and outline some of their properties in section 3.1. This is followed by a description of a number of community detection algorithms in section 3.2, some of which are studied in depth.

3.1 Graph & Network Generation

We first present four graph generative models: the Erdős–Rényi model, the Watts–Strogatz model, the Barabási–Albert model, and the Stochastic Block Model. For each model, we give the model definition, describe the edge generation process, and state properties shown by others.

3.1.1 Erdős–Rényi model

First introduced by Gilbert (1959) [19], and also independently by Erdős and Rényi (1959) [16], the Erdős–Rényi model is one of the first contemporary attempts in modelling a network with a random graph. The model has two common variants, which are defined as follow.

Definition 3.1. (Erdős and Rényi (1959) - Graph selection variant of Erdős–Rényi model)

The graph selection variant of Erdős–Rényi model generates a graph $G_{ER}(n, M)$ by choosing uniformly at random from the collection of all graphs with n nodes and M edges.

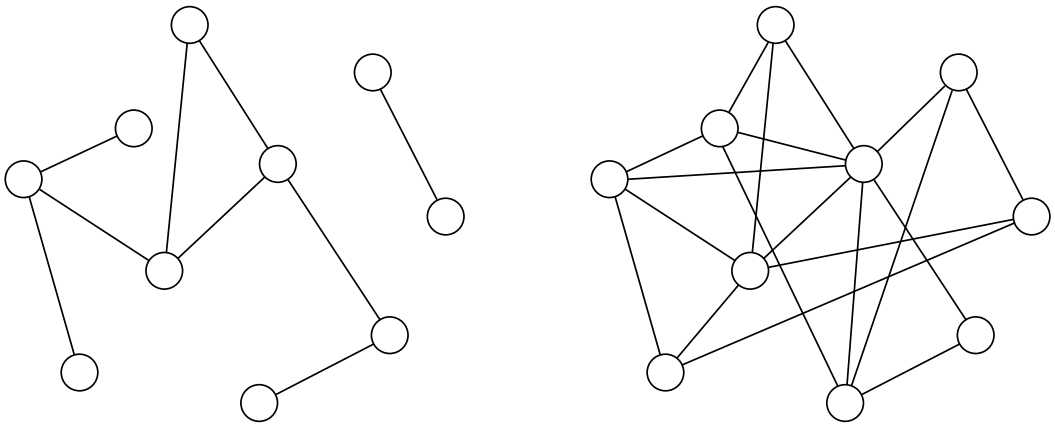
Remark. The collection of all graphs with n nodes and M edges has cardinality $\binom{n(n-1)/2}{M}$, as one have that many ways to place the M edges amongst $\frac{n(n-1)}{2}$ possible edges.

Definition 3.2. (Gilbert (1959) - Edge formation variant of Erdős–Rényi model)

The edge formation variant of Erdős–Rényi model, generates a graph $G_{ER}(n, p)$ by forming an edge between any two nodes with probability p, independent of other edges.

Remark. Graphs generated under the definition will have M edges with probability $p^M(1-p)^{\binom{n}{2}-M}$.

Figure 3.1 shows two graphs generated from the $G_{ER}(n, p)$ model with different connection probabilities p.



(a) $p = 0.2$ **(b)** $p = 0.4$

Figure 3.1: Sample graphs from $G_{ER}(n = 10, p)$, the edge formation variant of Erdős–Rényi model. Notice a higher p leads to more edges in the graph in general, and it is not a requirement for a $G_{ER}(n, p)$ graph to have exactly $\binom{n}{2}p$ edges as each edge is formed independently.

We will show from first principles that the node degrees of a $G_{ER}(n, p)$ graph follows the binomial distribution, and hence can be approximated by a Poisson distribution for large n and small p. The result is also shown by Newman (2001) [37].

We can link the two variants of the model by observing $p \approx M/\binom{n}{2}$. Edalat (2016) has shown assuming M is fixed, the clustering coefficient (the average probability for two neighbours of a node to connect themselves across all nodes) is p, which tends to zero for large n [15]. This is considered a undesirable property when modelling a real network, as real network usually demonstrates clustering behaviour.

3.1.2 Watts–Strogatz model

The Watts-Strogatz model is proposed by Watts and Strogatz (1998) to address the low clustering coefficient and lack of triadic closure (closed triangle between an three nodes) in the Erdős–Rényi model [48]. The model constructs a graph by first forming a regular lattice and conduct probabilistic edge rewiring, defined as follows and illustrated in figure 3.2.

Definition 3.3. (Regular lattice)

A regular ring lattice (with periodic boundary condition) with n nodes and number of neighbours $k \ll n$ is a graph with the following adjacency matrix:

$$A_{ij} = \begin{cases} 1 & \text{if } |i-j| \mod \left(n-1-\frac{k}{2}\right) \leq \frac{k}{2} \\ 0 & \text{otherwise} \end{cases}, \; i,j \in \{1,...,n\} \tag{3.1}$$

In other words, a regular ring lattice is a graph with nodes arranged in a ring, and each node connecting with the immediate $k/2$ neighbours on each side.

Definition 3.4. (Watts and Strogatz (1998) - Watts-Strogatz model)

The Watts-Strogatz model generates a graph with n nodes and average node degree k, denoted $G_{WS}(n,k)$, via the following procedures:

1. Generate a regular ring lattice with n nodes and k neighbours as defined in definition 3.3
2. For each edge (i,j) in the regular ring lattice, remove one end of the edge (say node j) and reconnect to a node (say node k) with probability p. The node which the edge is being rewired to, k, is uniformly selected at random, subject to $k \neq i$ (no self-loop) and $k \neq k'$ where the edge (i,k') exists before rewiring (no duplicated edge).

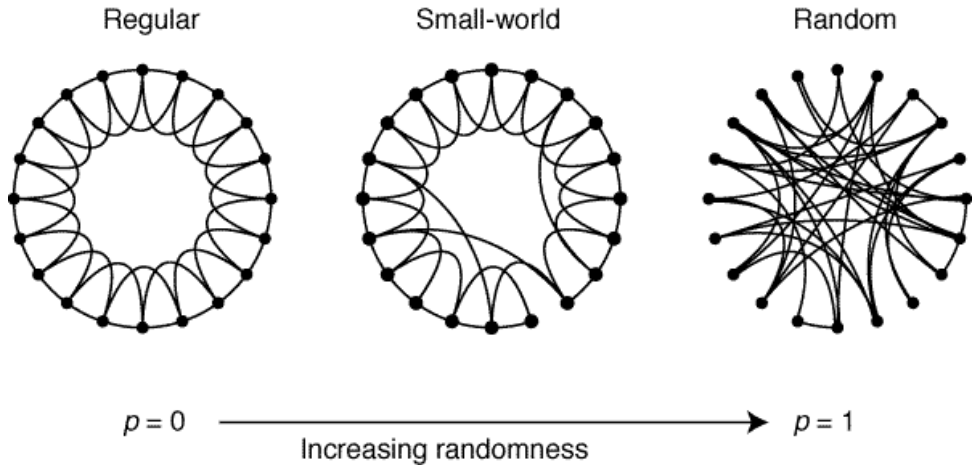


Figure 3.2: Watts and Strogatz (1998) - Probalistic edge rewiring for interpolating between a regular ring lattice and a random network [48].

Watts and Strogatz showed that for p in and around $[0.01, 0.1]$, the resultant graph gives a high clustering coefficient and low average path length (the average length of the shortest path between a pair of nodes), collectively known as the small-world properties and found in real networks.

The Watts-Strogatz model does not demonstrate the scale-free property, which node degrees follow a power-law distribution. Under a $G_{WS}(n,k)$ graph, the majority of nodes will still have around k connections after probabilistic rewiring. This is in contravention of the node degree distribution for real network, where the majority of the population have only a handful of connections, and a handful of individuals have the majority of connections.

3.1.3 Barabási–Albert model

The Barabási-Albert model, proposed by Barabási and Albert (1999), originates from the study of links between web pages on the world wide web. It describes a growing network (increasing number of nodes over time) where nodes gain edges by preferential attachment - the more connections you have, the more likely you will get new connections, or more formally:

Definition 3.5. (Barabási and Albert (1999) - Barabási-Albert model) The Barabási-Albert model generates a growing graph with n_0 initial nodes, denoted $G_{BA}(n_0)$, via the following procedures:

1. Initialise a connected network with n_0 nodes
2. For each new node being added to the network already with n nodes, connect the new node $n+1$ with an existing node $i \leq n$ with probability:

$$p_i = \frac{k_i}{\sum_j k_j} , \tag{3.2}$$

   where k_i is the degree for node i just before the new node is added to the network

The following properties make the Barabási-Albert model more desirable in modelling real networks when compared to random graphs. Barabási and Albert (1999) has shown the node degree of the Barabási-Albert model follows a power-law distribution:

$$\mathbb{P}(\text{Node degree} = k) \sim k^{-3} . \tag{3.3}$$

Furthermore, it has been shown by Albert and Barabási (2002) that the average path length of graphs generated by the Barabási-Albert model is lower than that generated by the Erdős–Rényi model [2]. Klemm and Eguíluz (2002) has also obtained the clustering coefficient which is higher than the Erdős–Rényi model [28].

However, while the Barabási-Albert model demonstrates a "hub" behaviour - nodes which possess the highest numbers of connections, it does not account for any community structure, which is common in real networks.

3.1.4 Stochastic Block Model

First developed by Holland et al. (1983) [25], the Stochastic Block Model divide nodes into disjoint sets (known as communities) and specify the connection probability of two nodes based on which community they belonged to. More formally:

Definition 3.6. (Stochastic Block Model)

Given the number of nodes n, the set of r communities $\{C_i\}_{i\in\{1,\dots,r\}}$ which partitions the nodes

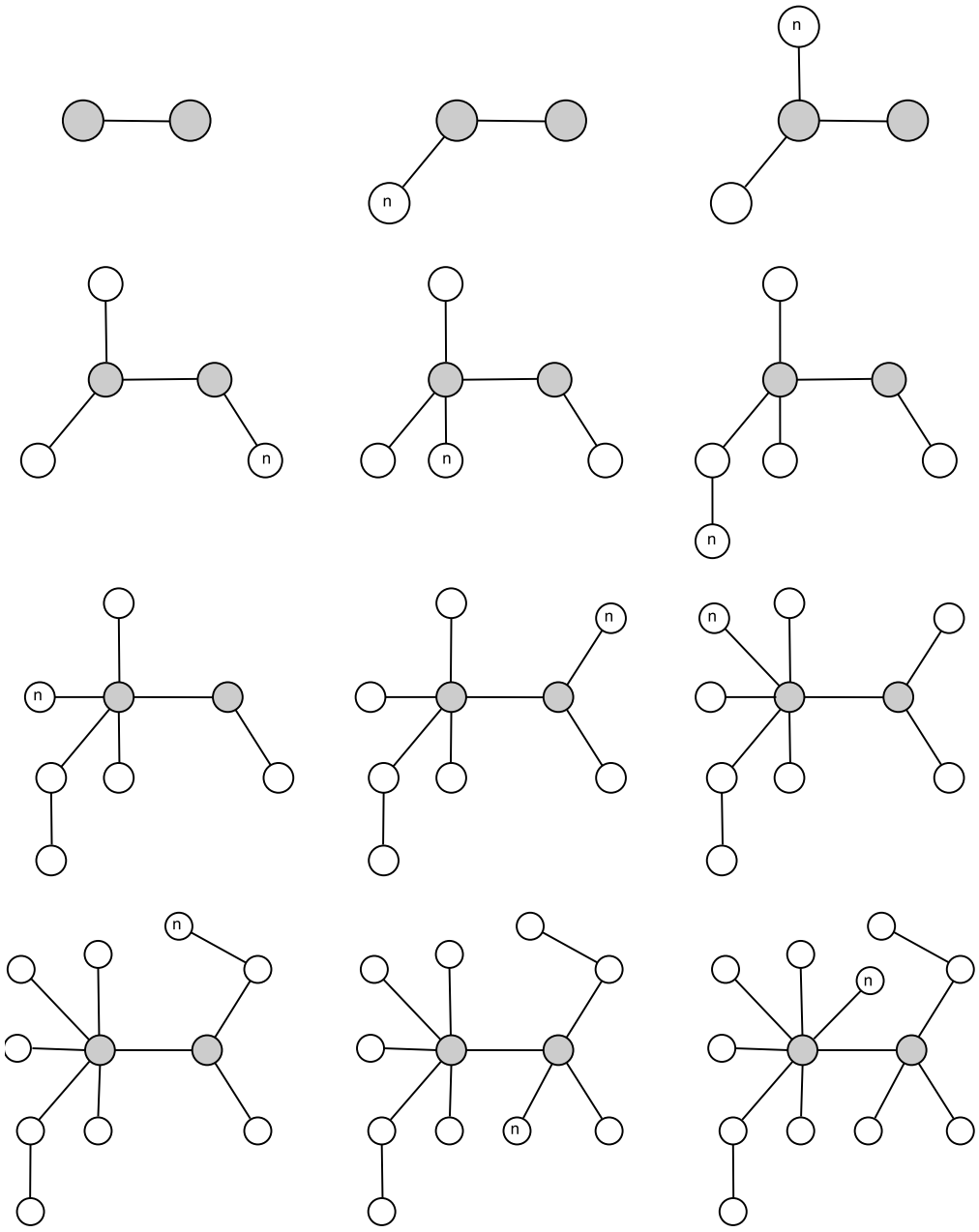


Figure 3.3: A growing network generated by the specification of the Barabási-Albert model with two initial nodes (coloured in grey). The timeline is shown left-right top-down, where a new node (labelled "n") is added at each instance. Notice the "hub" behaviour demonstrated by the left initial node - it gains more connections than other nodes due to the large number of connections it possesses at all times.

(i.e. $C_i \cap C_j = \varnothing \ \forall \, i \neq j$, $\bigcup_{i=1}^{r} C_i = \{1, ..., n\}$), and the inter-community connection probability matrix $P = (P_{ij}) \in \mathbb{R}^{r \times r}$, the Stochastic Block Model generates a graph, denoted as $G_{SBM}(n, \{C_i\}, P)$, with the following edge formation probability between nodes u and v, $u \neq v$:

$$\mathbb{P}((u, v) | u \in C_i, v \in C_j) = P_{ij} \ , \tag{3.4}$$

where (u, v) denotes the edge between nodes u and v.

Stochastic Block Model is one of the few graph generative models which takes the community structure into account (see figure 3.4), albeit it does not consider overlapping communities.

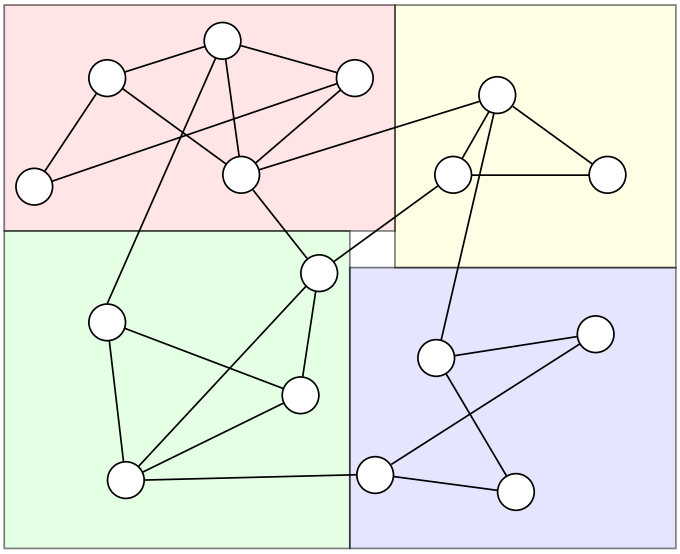


Figure 3.4: A 16-node network generated by the Stochastic Block model with 4 blocks (coloured rectangles) and its associated inter-community connection probability matrix (not shown in figure).

3.2 Community Detection

We also outline existing work on community detection in networks. Communities to be detected can be overlapping or non-overlapping under different models.

3.2.1 Community-Affiliation Graph Model (AGM)

Based on "[t]he observation that community overlaps are more densely connected than the non-overlapping parts", Yang and Leskovec (2012) introduced the Community-Affiliation Graph Model (AGM) for overlapping network community detection [53]. We summarise the model definition and working mechanisms below.

Model Definition

Yang and Leskovec define the underlying model of AGM as a bipartite community affiliation, denoted $B(V, C, M)$ and illustrated in figure 3.5:

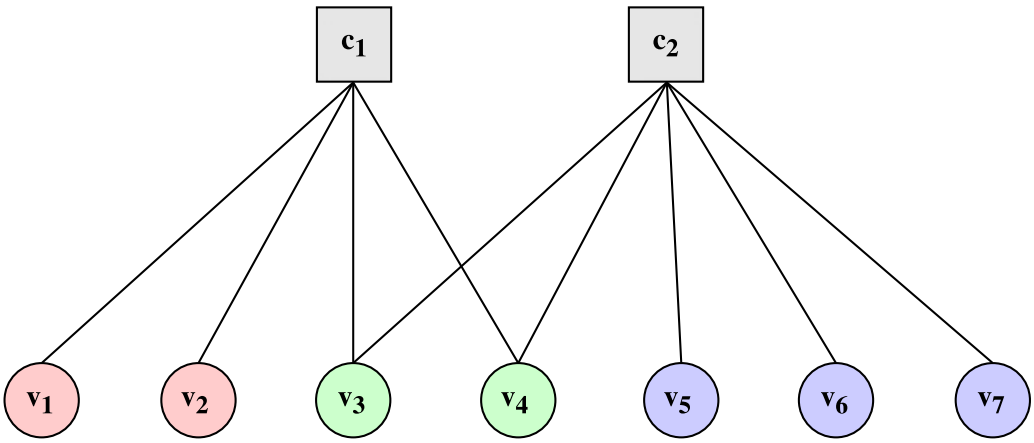


Figure 3.5: A bipartite community affiliation graph representing a network with 2 communities (square nodes, $c_i \in C$) and 7 individuals (circle nodes, $v_i \in V$). Each edge between a square node and a circle node represents a membership/affiliation between a community and an individual.

V: Individuals The set of individuals within a social network, corresponding to the vertices in the social network's graph representation.

C: Communities The set of underlying communities within a social network. The communities are the hidden nodes in the social network's graph representation, which we are interested in uncovering given any social network graphs.

The model also assigns a parameter p_c to each community $c \in C$, which denotes the probability for two individuals in the same community c to connect with each other. We collate the probabilities via the following definition:

Definition 3.7. (Intra-community connection probabilities)

A non-negative vector $\boldsymbol{p} = (p_c)_{c \in C}$, where p_c denotes the intra-community connection connection probability for community c, is referred as the intra-community connection probabilities vector.

M: Membership/ Affiliation The set of undirected edges $\{(v, c) : v \in V, c \in C\}$ between individuals and communities, representing an individual's affiliation towards a community, or equivalently a community membership of an individual. There are no edges between two individuals, nor edges between two communities.

Graph Generation with AGM

Given a bipartite community affiliation model with intra-community connection probabilities, Yang and Leskovec generate a network $G(V, E)$ using the following definition.

Definition 3.8. Given a bipartite community affiliation model $B(V, C, M)$, and the intra-community connection probabilities $\boldsymbol{p}$, AGM generates an edge $(u, v) \in E$ between two individual nodes $u, v \in V$ with probability:

$$p(u, v) = 1 - \prod_{k \in C_{uv}} (1 - p_k) \ , \tag{3.5}$$

where $C_{uv} \subseteq C$ is the set of communities which u and v are both members of.

Definition 3.8 allows two individuals u and v to have an independent chance to connect for each shared affiliation, with probability determined by the community in question. This results in a higher probability for individuals with larger number of mutual community memberships to connect.

We will show in section 4.4 that the generative process in definition 3.8 would not yield node degrees following a power-law distribution, which is often observed in real networks.

ϵ-community Under definition 3.8, it is impossible for two individuals $u, v \in V$ with no mutual community affiliation to connect with each other in the generated network (as $F_{uc} F_{vc} = 0$ $\forall c \in C$). To preserve the possibility of two individuals connecting at random without sharing any communities, the graph generative method assumes an extra ϵ-community, which connects any two nodes with probability $\epsilon = \frac{2|E|}{|V|(|V|-1)}$, which is the probability of an edge forming in a random network.

Community Detection with AGM

We then briefly discuss the reverse process - given a social network graph $G(V, E)$, how do Yang and Leskovec uncover the bipartite community affiliation model $B(V, C, M)$ and the intra-community connection probabilities $\boldsymbol{p}$. The resultant optimisation problem, as discussed by Yang and Leskovec (2013), is "a combinatorial optimisation problem that is hard to solve" [54]. Hence methods in finding a global optimum is less likely to be scalable and possess less value to be discussed thoroughly.

Yang and Leskovec employed a Gibb's sampling-like approach in approximating $B(V, C, M)$ and $\boldsymbol{p}$ which achieves the highest likelihood:

$$\arg\max_{B, \boldsymbol{p}} L(B, \boldsymbol{p}) = \prod_{(u,v) \in E} p(u, v) \prod_{(u,v) \notin E} (1 - p(u, v)) \ , \tag{3.6}$$

where B is the short-hand for $B(V, C, M)$, and E is the set of edges in the given network. They alternate between keeping $B(V, C, M)$ fixed and update $\boldsymbol{p}$, and keeping $\boldsymbol{p}$ fixed and update

$B(V, C, M)$. They update $\boldsymbol{p}$ by reformulating the log-likelihood as a convex function, which enables use of gradient ascent or Newton's method to obtain a global optimum for $\boldsymbol{p}$ within the stage. Updating of B is done by using the Metropolis-Hastings algorithm with stochastic affiliation addition/deletion/rewiring.

3.2.2 Cluster Affiliation Model for Big Networks (BigClam)

Yang and Leskovec (2013) also introduced the Cluster Affiliation Model for Big Networks (BigClam), "an overlapping community detection method that scales to large networks of millions of nodes and edges" [54]. The model is largely similar the AGM, though with the introduction of an affiliation-dependent weight and a new energy function for edge formation, BigClam is capable of capturing community affiliation in greater detail, and is easier to be scaled.

Model Definition

The underlying model for BigClam is largely identical to that used in AGM - a bipartite community affiliation, denoted $B(V, C, M)$ (see figure 3.5). Though unlike AGM, BigClam does not feature a intra-community connection probability, but a weight associated to each membership/affiliation. The model assigns a non-negative weight $F_{vc}\,(v \in V,\, c \in C)$ to each edge of the bipartite graph (see figure 3.6) to denote the strength of membership/ affiliation between an individual and a community - the higher the weight the stronger the membership/ affiliation. The given individual and community are unaffiliated if the edge between has a weight of zero.

We can represent affiliation weights between every individual and community by non-negative matrices and vectors as defined below.

Definition 3.9. (Community affiliation weights matrix and vector)

A non-negative matrix $F = (F_{vc}) \in \mathbb{R}^{|V|\times|C|}$ representing affiliation weights between individuals and communities is referred to as the community affiliation weights matrix (*affiliation weights matrix* in short).

A non-negative *row* vector $F_u = (F_{uc}) \in \mathbb{R}^{|C|}$ representing affiliation weights between the individual u and all communities is referred to as the community affiliation weights vector associated with individual node u. If u is unspecified, F_u is referred to as a node-associated community affiliation weights vector (*affiliation weights vector (for u)* in short).

Graph Generation with BigClam

Given a bipartite community affiliation model $B(V, C, M)$ with its associated non-negative affiliation weights matrix F, we can generate a network $G(V, E)$ using the following definition.

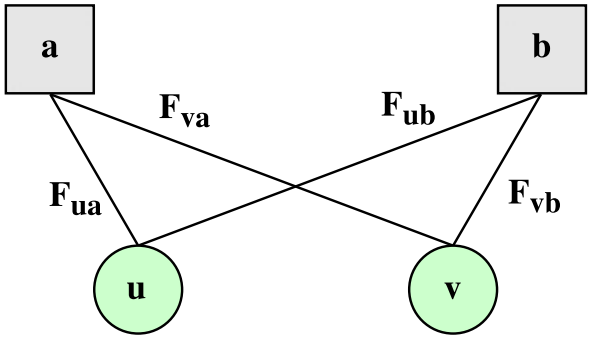


Figure 3.6: Every edge between an individual $v \in V$ and a community $c \in C$ is given a weight $F_{vc} \geq 0$ under the community affiliation model

Definition 3.10. Given a bipartite community affiliation model $B(V, C, M)$, an affiliation weights matrix $F = (F_{vc}), v \in V, c \in C$, and the affiliation weights vector for $u, v \in V$, $F_u = (F_{uc_i}), c_i \in C$ and $F_v = (F_{vc_i}), c_i \in C$ respectively. BigClam generates an edge $(u, v) \in E$ between two individual nodes $u, v \in V$ with probability:

$$p(u, v) = 1 - \exp\left(-F_u {F_v}^T\right) = 1 - \exp\left(-\sum_{c \in C} F_{uc} F_{vc}\right) . \tag{3.7}$$

This is based on the assumption that each mutual community membership between two individuals will gain the individuals an independent, non-zero chance of a connection. Hence the connection probability is strictly increasing with respect to the increasing number of mutual communities. Moreover, stronger affiliations between said individuals and their mutual community(ies) (higher edge weight between) result in a higher probability of the individuals connecting.

Similar to AGM, BigClam preserves the possibility of two individuals connecting at random without sharing any communities by creating an ϵ-community, where edges connect with probability ϵ.

Community Detection with BigClam

In the presence of a network $G(V, E)$ and absence of knowledge on true community affiliations, we are interested in finding the most likely community affiliations, represented by a non-negative affiliation weights matrix $\hat{F}$. Yang and Leskovec (2013) proposed the following non-negative matrix factorisation-like approach in obtaining $\hat{F}$ [54]. Note the sections below only provide a general overview and cover key mathematical derivations of the community detection algorithm (*the algorithm* in short) proposed by Yang and Leskovec - further implementation details can be found in [54].

Log-likelihood maximisation Given a network $G(V, E)$, $\hat{F}$ is the matrix which maximises the likelihood/ log-likelihood function defined as follows.

Definition 3.11. (Likelihood and log-likelihood of an affiliation weights matrix under a known network)

Let $G = G(V, E)$ be a network with all vertices and edges known, and $F = (F_{vc}) \in \mathbb{R}^{|V| \times K}$ $(v \in V,\, c \in C)$ be an affiliation weights matrix (where K is the number of communities we attempt to detect).

The likelihood for F under G is

$$\begin{aligned} L(F) &= Pr(G|F) = \tfrac{1}{2} \prod_{u,v \in V,\, u \neq v} \begin{cases} p(u,v) & \text{if } (u,v) \in E \\ 1 - p(u,v) & \text{otherwise} \end{cases} \\ &= \prod_{(u,v) \in E} p(u,v) \prod_{(u,v) \notin E} (1 - p(u,v)) \\ &= \prod_{(u,v) \in E} (1 - \exp(-F_u {F_v}^T)) \prod_{(u,v) \notin E} \exp(-F_u {F_v}^T) \,. \end{aligned} \tag{3.8}$$

The log-likelihood for F under G is

$$l(F) = \log L(F) = \sum_{(u,v) \in E} \log(1 - \exp(-F_u {F_v}^T)) - \sum_{(u,v) \notin E} (F_u {F_v}^T) \,. \tag{3.9}$$

In other words, the most likely community affiliation is given by

$$\hat{F} = \arg\max_{F \geq 0} L(F) = \arg\max_{F \geq 0} l(F) \,. \tag{3.10}$$

We generally work with log-likelihood functions as they involve sums of smaller components, instead of a large product.

Maximisation as convex optimisation problem The general optimisation problem (equation 3.10) can be broken down into one sub-problem per $u \in V$ - finding the most likely affiliation weights vector for $u \in V$, while fixing all other affiliation weights vectors $F_v \in V$, $v \neq u$:

$$\hat{F}_u = \arg\max_{F_{uc} \geq 0,\, c \in C} l(F_u) \,. \tag{3.11}$$

Definition 3.12. (Log-likelihood of an affiliation weights vector under a known network and its gradient)

Let $G = G(V, E)$ be a network with all vertices and edges known, and $F_u = (F_{uc}) \in \mathbb{R}^K$ $(u \in V,\, c \in C)$ be an affiliation weights vector for $u \in V$ (where K is the number of communities we

attempt to detect).

The log-likelihood for F_u under G is

$$l(F_u) = \sum_{v \in \mathcal{N}(u)} \log\left(1 - \exp(-F_u {F_v}^T)\right) - \sum_{v \notin \mathcal{N}(u)} F_u {F_v}^T \ , \tag{3.12}$$

where $\mathcal{N}(u)$ is the set of neighbours of u under G (see definition 2.4).

The gradient of log-likelihood for F_u is

$$\nabla l(F_u) = \sum_{v \in \mathcal{N}(u)} F_v \frac{\exp(-F_u {F_v}^T)}{1 - \exp(-F_u {F_v}^T)} - \sum_{v \notin \mathcal{N}(u)} F_v \ . \tag{3.13}$$

The sub-problem (equation 3.11) can then be tackled using a gradient ascent approach, with the gradient defined above and a learning-rate parameter α:

$$l'(F_u) = l(F_u) + \alpha \nabla l(F_u) \ . \tag{3.14}$$

It can be shown $l(F_u)$ is a convex function, and hence the sub-problem is a convex optimisation problem: any local maximum found will be the global maximum.

Complexity reduction Naive implementations of equation 3.13 will scale along $O(|V|)$, as each computation of $\nabla l(F_u)$ requires obtaining $F_v \ \forall v \in V$. Though it is observed that:

$$\sum_{v \in \mathcal{N}(u)} F_v = \left(\sum_v F_v\right) - F_u - \left(\sum_{v \in \mathcal{N}(u)} F_v\right) \ . \tag{3.15}$$

By pre-fetching $\sum_v F_v \ \forall v \in V$ once and retaining its value, and only calculating $\sum_{v \in \mathcal{N}(u)} F_v$ at run-time, the time complexity is reduced to $O(|\mathcal{N}(u)|)$ (with a $O(|V||C|)$ storage overhead). The sparsity of a real network implies $|\mathcal{N}(u)| \ll |V|$, and hence optimisations based on equation 3.15 would give a massive performance speed boost in theory.

Miscellaneous Implementation Details

Initial condition The initial affiliation weights matrix is determined with a conductance test on all nodes to locate locally minimal neighbourhoods.

Definition 3.13. (Initialisation of affiliation weights matrix)

Let F be an affiliation weights matrix. The algorithm initialises F as follows:

$$F_{(u')(N(u))} = \begin{cases} 1 & \text{if } u' \in N(u) \text{ and } N(u) \text{ is a locally minimal neighbourhood of } u \\ 0 & \text{otherwise} \end{cases} , \quad (3.16)$$

where $N(u)$ is that defined in definition 2.4.

In other words, the algorithm utilises the set of locally minimal neighbourhoods as the seed set of communities. Yang and Leskovec did not specify in their paper which locally minimal neighbourhoods are included in the seed set if the size of the set of locally minimal neighbourhoods differs from the number of communities we would like to detect.

We may want to modify the initial condition to detect members of known communities, and/or pivot on known member(s) of a community.

Stopping condition The algorithm iteratively updates $F_u \; \forall u \in V$, and terminates by the condition described as follows.

Definition 3.14. (Termination of the community detection algorithm)

Let $l(F_u)$ be the log-likelihood of the affiliation weights vector for u, and $l'(F_u)$ be the log-likelihood of the affiliation weights vector for u after the update by gradient ascent (equation 3.14). The iterative gradient ascent process terminates if:

$$\frac{l'(F_u) - l(F_u)}{l(F_u)} < 0.0001 \quad \forall u \in V \; , \quad (3.17)$$

i.e. the log-likelihood increases by less than 0.01% $\forall u \in V$.

Remark. As the log-likelihood function $l(F_u)$ is convex, the gradient ascent approach is guaranteed to reach the global optimum. The terminating condition would lead to a solution sufficiently close to the global optimum within a reasonable time.

Determining community affiliations Upon termination, the algorithm determines the existence of an individual's affiliation/ a community membership as follows.

Definition 3.15. (Determining existence of affiliation/ membership)

Let $G = G(V, E)$ be a network, F be the most likely affiliation weights matrix under G, and $\epsilon = \frac{2|E|}{|V|(|V|-1)}$ be the background probability for a random edge to form in G.

The algorithm regards $u \in V$ as a member of $c \in C$ if $F_{uc} \geq \delta = \sqrt{-\log(1-\epsilon)}$.

Any two affiliation weights higher than δ between two individuals $u, v \in V$ and a community will result in the probability of forming an edge (u, v) being higher than the background probability, a possibility introduced by the ϵ-community.

Determining number of communities In section 3.2.2, we implicitly assumed the number of communities K is given. Yang and Leskovec outlined the following method to obtain the most likely number of communities $\hat{K}$.

1. Select 20% of node pairs at random as hold out set H
2. For every K as a candidate on number of communities:
   (a) Run the community detection algorithm, with K as the number of communities we intend to detect, on the remaining 80% of node pairs
   (b) Evaluate the likelihood of affiliation weights with respect to H, $l_K(F_H)$
3. Obtain the most likely number of communities $\hat{K}$ as the candidate K which has achieved the highest likelihood with respect to H (i.e. $\hat{K} = \arg\max_K l_K(F_H)$).

A full, working software implementation on BigClam can be found in the Stanford Network Analysis Project (SNAP) library, created and maintained by Leskovec and Sosič (2014) [34].

3.2.3 Other Community Detection Methods

We also list and briefly discuss some other community detection methods available. Interested readers can also refer to the survey paper by Fortunato (2010), where a large number of community detection algorithms of different genres are discussed [17].

Non-negative Matrix Factorisation (NMF) Non-negative Matrix Factorisation (NMF) is a class of methods which aims to reduce the dimension of a given dataset by performing a matrix decomposition. Given a non-negative data matrix A, NMF finds two non-negative matrices B and C such that $A \approx BC$. Hoyer (2004) has discussed methods to factorise a sparse matrix [26], opening up the opportunity for one to detect communities by factorising the adjacency matrix into a lower dimension matrix, which may be interpreted as the community affiliation matrix. Yang and Leskovec's (2013) work on BigClam is heavily motivated by this class of methods.

Louvain Method Proposed by Blondel et al. (2008), the Louvain Method obtain a set of hierarchical communities (disjoint across the same level) by optimising the modularity of a

partition of the network [7]. The modularity of a partition is defined as

$$\frac{\text{\# Edges in the partition}}{\text{\# Edges in the network}} - \frac{\text{Expected \# edges in the partition if randomly distributed}}{\text{\# Edges in the network}} . \quad (3.18)$$

The method starts from node level and finds groups of nodes ("local communities") that optimises the modularity locally. It then finds groups of local communities that further optimises the modularity, and so on until the global optimum in modularity is reached. This results in a hierarchical community structure being detected, though the communities at the same level are not overlapping. With the Louvain method, Blondel et al. (2008) managed to solve a 118 million-node network in less than 3 hours [7].

Chapter 4

Graph Generative Process for Overlapping Communities

The ability to generate real-like graphs is powerful: it allows us to discover latent features of a social network (e.g. community structure), and hence understand social dynamics and inherent properties of society as discussed in section 1.1.1. It also provides the opportunity to quantitatively evaluate network-related algorithms. However, current graph generative models struggle to create graphs with all the properties of real networks, including overlapping community structure and power-law distributed node degrees. This means the graphs generated by those models carry less value for further study.

In this chapter, we propose a new generative model for large interaction networks with overlapping communities and power-law distributed node degrees. The model consists of two submodels: an overlapping community generative model, and a graph generative model based on overlapping communities, which features a measurement of *influence*, a possible explanation of why individuals connect with each other. We show our generative model is capable of generating communities with power-law distributed membership counts, individuals with affiliation count following a distribution with exponential decay, and graphs with power-law distributed node degrees under some common conditions. The work also opens up discussion on the number of communities a network can support, and motivates evaluation of overlapping community detection algorithms, a study of which can be found in section 5.1.

We organise the chapter as follows. Section 4.1 covers a more in-depth exploration on real networks and the need for a graph generative process for overlapping communities and power-law distributed degrees. We proceed to outline the specification for our synthesised (overlapping) communities and graphs in section 4.2, and detail our community and graph generative processes in sections 4.3 and 4.5 respectively. The result of our generative processes is available in section 4.6, with our evaluation and discussion in section 4.7 and possible work the generative processes has opened up in section 4.8.

4.1 Motivation

We begin by outlining our observations on real networks, and why it motivates the need of a new graph generative process which takes an overlapping community structure and power-law distributed node degrees into account.

4.1.1 Power Law in Node Degree Distribution

Interaction networks with node degrees following a power law distribution, also known as scale-free networks, were studied since as early as 1965, when Price (1965) published his findings on citation networks [43].

A large number of networks are scale-free, with node degree following a power-law distribution. Newman (2005) has included the node degree distribution for the AT&T telephone call network, amongst others, as examples which power law is evident [38]. Moreover, we observe node degree distribution in networks maintained by Leskovec and Krevl (2014) [33] appears asymptotically linear on a log-log plot (see figure 4.1), suggesting the node degrees for the networks are power-law distributed.

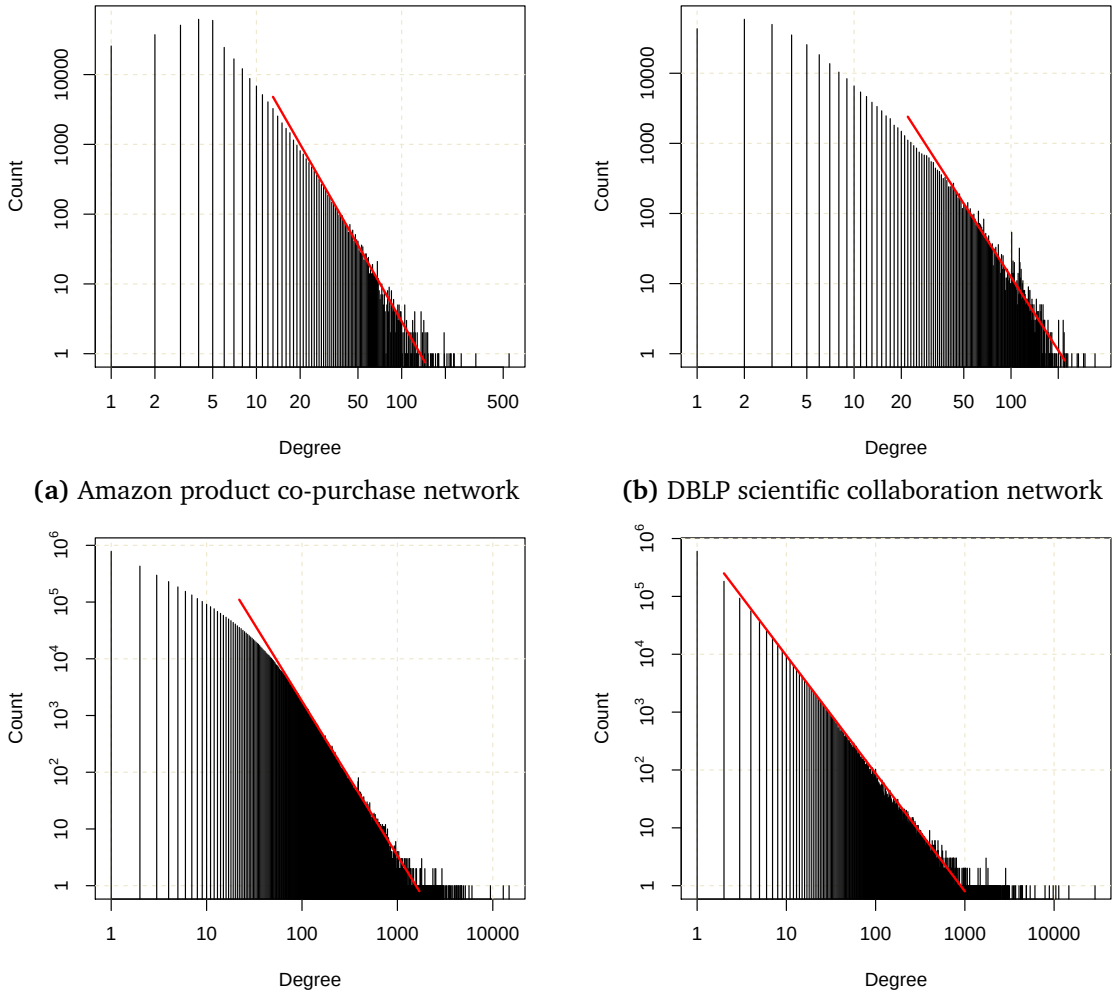
4.1.2 Distributional Result in Membership/Affiliation Counts

We also observe the number of members in a communities appears to follow the power law. This is shown in figures 4.2a and 4.2b, where the membership distribution appears linear on the log-log plot. The phenomenon is also discussed by Newman (2005), who investigated power laws in the number of species in a genus, city size, and book sales [38]. If we treat genera, cities and individuals who have bought the same books as communities, then their size (i.e. number of membership) is power-law distributed.

On the other hand, the number of affiliations for an individual does not follow the power law - the associated probability density function decays exponentially. Broderick et al. (2012) have proved the number of features demonstrated by a feature point under a Beta Process do not follow the power law, but decays exponentially [8]. By the equivalence of features and overlapping communities as discussed in section 1.1.2, we can adapt the result from Broderick et al. to establish the said statement on community affiliation distribution. The exponential decay can be visualised in figures 4.3a and 4.3b, where the affiliation distributions do not appear linear on the log-log plot.

4.1.3 Need for An Overlapping Community-Based Graph Generative Process

Based on the observations and related work discussed above, it is clear a graph generative process that involves overlapping communities and is capable of generating power-law distributed



(a) Amazon product co-purchase network

(b) DBLP scientific collaboration network

(c) LiveJournal blogging community

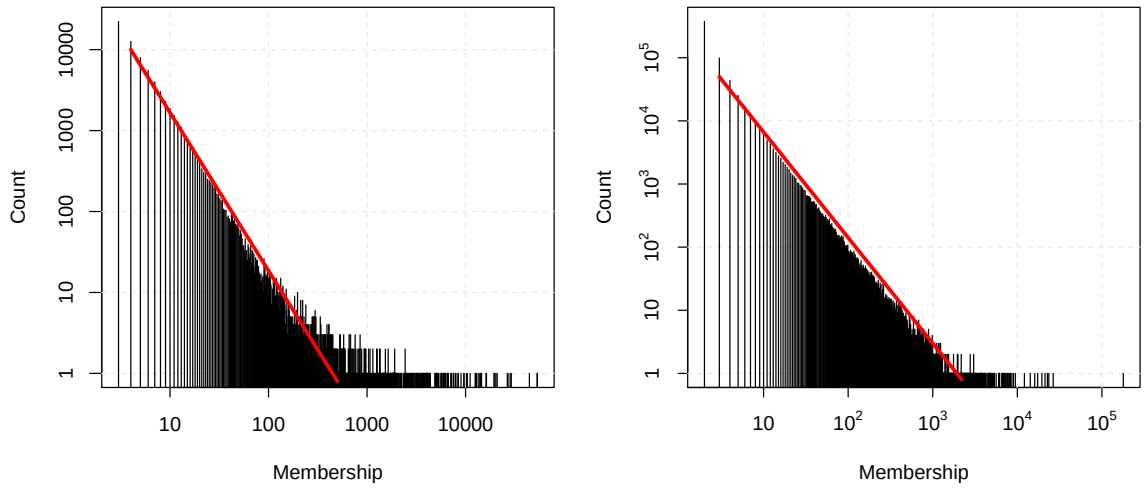
(d) Youtube social network

Figure 4.1: Log-log plots on node degree distribution for the networks with ground-truth communities. The plots suggest node degree are asymptotically power law-distributed, as marked by the straight line in each plot.

node degrees is required.

Current state-of-the-art methods usually satisfy some but not all of the requirements: The Erdős–Rényi model (see section 3.1.1) and the Watts–Strogatz model (see section 3.1.2) do not generate scale free networks (i.e. with power-law distributed node degrees), the Barabási–Albert model (see section 3.1.3) does not involve any community structure[1], and the Stochastic Block Model (see section 3.1.4) assumes non-overlapping communities. This motivates our work in developing a generative model which resultant graphs satisfy all the said properties.

[1]Though the essence of the model - "the rich gets richer" - inspired our development on the influence measure (see section 4.5.1).



(a) Amazon **(b)** LiveJournal

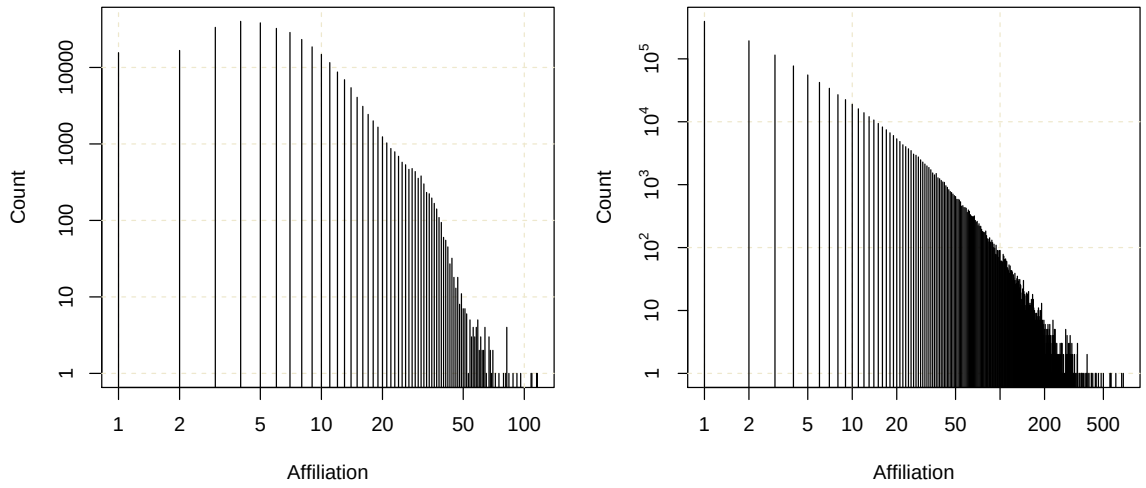
Figure 4.2: Log-log plots for distribution on number of community memberships in the Amazon product co-purchase network (left) and the LiveJournal social network (right). The distributions appears asymptotically linear (denoted by the red lines), suggesting they follow the power law.

4.2 Specification

Our new community and graph generative process will produce networks with the following characteristics:

- Containing $|V|$ individuals
- Containing $|C|$ communities
- Having in total $|M| = |V| \times r$ affiliations (where r denotes the average number of affiliations per individual, and measures the degree of overlap between communities - see figure 4.4)
- Having the number of members in a community following a power law distribution with shape parameter α_c.
- Having the minimum number of members in a community, $m_{\min}$, as three (see section 4.2.1)
- Having in total $|E| \approx |V| \times \frac{|\hat{E}|}{|V|}$ edges (where $\frac{|\hat{E}|}{|V|}$ denotes the target average node degree - number of connections an individual has in the network)

All parameters above, apart from $|M|$ (which could be easily inferred from $|V|$ and r), are specified.



(a) Amazon **(b)** LiveJournal

Figure 4.3: Log-log plots for distribution on number of individual affiliations in the Amazon product co-purchase network (left) and the LiveJournal social network (right). The distributions does not appear to follow the power law.

4.2.1 Minimum number of members for a community

We also assume a community has at least three members. The assumption excludes the possibility of one-person communities, which is equivalent to the individual node itself, and two-person communities from existing. The latter motivates a circular definition - individuals connect with each other because they are affiliated with the same community, which (in two-person community cases) arises because the two individuals are connected to each other - and should be avoided. The assumption is also used in BigClam's implementation by Yang and Leskovec (2012) [54].

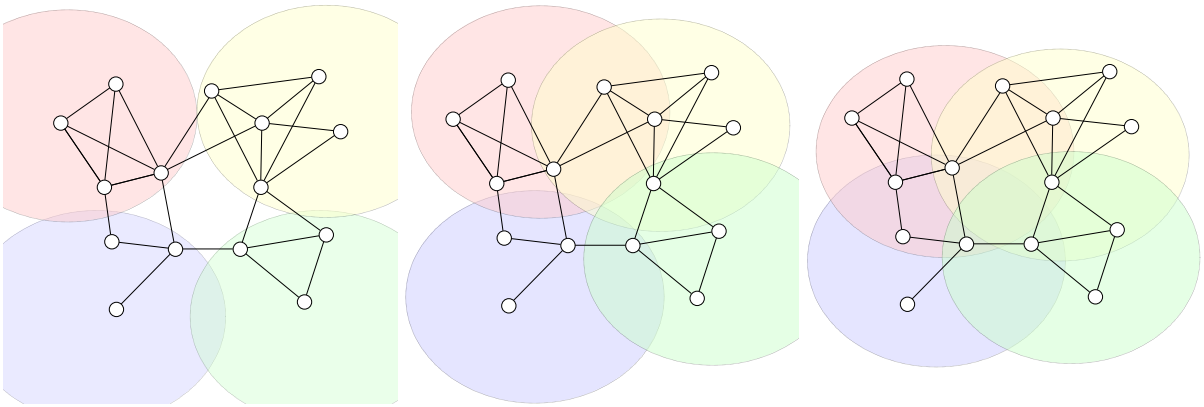
The corresponding constant (minimum number of members per community) is denoted:

$$m_{\min} = \min_{C,M} \left\{ \frac{|M|}{|C|} \right\} = 3 \,, \tag{4.1}$$

where C is the set of all communities, M is the set of memberships/affiliations, and $|\cdot|$ is the cardinality function. Note $m_{\min}$ could be an integer greater than three if we are interested in larger communities, and hence ignoring communities with few members.

4.3 Community Generation

Our community generation method assumes the existence of $|V|$ individuals and $|C|$ (empty) communities within a network, and generates the membership/ affiliation link between individuals and communities. Note we do not generate the edges between individuals in this method.



(a) Disjoint communities (i.e. clusters) ($r = 1$)

(b) Lightly overlapping communities ($r = 1.33$)

(c) More densely-overlapping communities ($r = 1.87$)

Figure 4.4: Three 15-node networks with different underlying 4-community (denoted in red, yellow, green and purple ellipses) arrangements. Each community arrangement features a different degree of overlapping, measured by average number of community affiliations for each individual node (r).

4.3.1 Obtaining number of members for each community

We first recall the requirements on number of members for each community. Let $C_i \in C$, $i \in \{1, ..., |C|\}$ be community i within the network, and $|C_i|$ be the size (number of members) of the community. The sequence $\{|C_i|\}_{i \in \{1,...,|C|\}}$ should satisfy the following properties:

1. The value of each element is greater or equal to $m_{\min}$ (see section 4.2.1)
2. The values asymptotically follow a power law distribution with shape parameter α_c (see section 4.1.2)
3. The values sum up to the total number of memberships/affiliations $|M|$

We notice any random sequence y_i following Type II Pareto distribution with location parameter $\mu = m_{\min}$ and shape parameter α_c satisfies properties 1 and 2 after rounding. Choosing an appropriate σ, which may vary from one sequence to another, will also satisfy property 3. We start with a random sequence:

$$x_i \overset{\text{iid}}{\sim} Pareto(I)(\sigma = 1, \alpha_c) \ , \tag{4.2}$$

which follows Type I Pareto distribution, and perform the transformation $y_i = \sigma'(x_i - 1) + m_{\min}$ to obtain:

$$y_i \overset{\text{iid}}{\sim} Pareto(II)(\mu = m_{\min}, \sigma', \alpha_c) \ , \tag{4.3}$$

a Type II Pareto distribution (see theorem 2.12). We tentatively set $\sigma' = \frac{|M|}{\sum_{i=1}^{|C|}(x_i - 1)}$ to enable development of our method, recognising it does not satisfy property 3 at this stage - the transformed sequence sums to $|M| + |C| \times m_{\min}$.

4.3.2 Membership/ affiliation generation

Having obtained the number of members for each community, we generate the membership/ affiliation (links between individuals and communities) via the following two-stage process, which is also illustrated in figure 4.5:

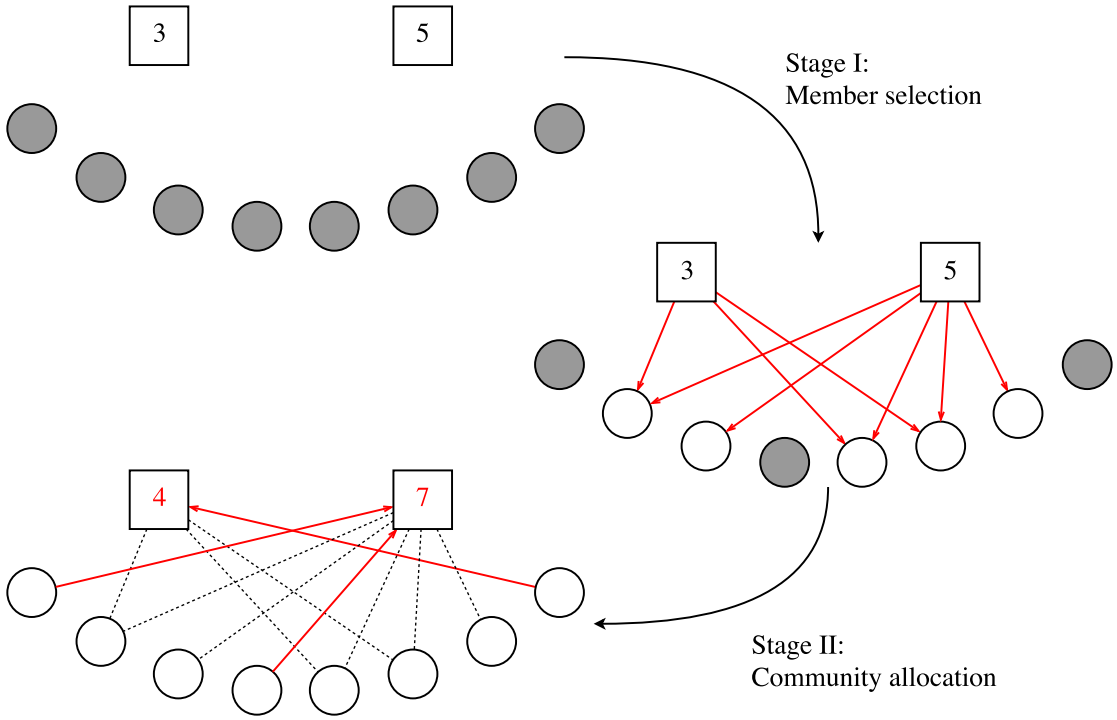


Figure 4.5: Illustration on generating membership/ affiliation for a network with two communities and eight individuals. Communities are denoted in squares, with the number within being the number of members. Individuals are denoted in circles in grey for individuals without an affiliation, and in white for individuals with affiliation(s). Updates at each stage (including individual-community link generation and changes in member count) are marked with solid red arrows, with the direction denoting from whose perspective the membership/ affiliation generation is done.

Stage I: Member selection For each community i, the number of members in the community is given as y_i, and we uniformly choose y_i individuals to be members of the community.

Stage II: Community Allocation We then randomly allocate individuals who failed to gain an affiliation in stage I to community i with probability proportional to the community's size after stage I (i.e. $y_i / \sum y_i$). By doing so, we scale up the community sizes proportionally, maintaining the power law distribution.

4.3.3 Discount factor

Recall we did not maintain property three. Ideally, we should downside the number of members admitted for all communities in stage I such that the total number of links satisfies the property. We calculate the discount factor $d \in (0, 1]$ based on the observations that our previous choice of σ' gives $|C| \times m_{\min}$ memberships in excess, and on average $|V|(1 - 1/|V|)^{|M|+|C|\times m_{\min}}$ individuals are not affiliated to a community after stage I and require an affiliation.[2] Hence we find d which satisfies the following equation:

$$\underbrace{|M| \times d + |C| \times m_{\min}}_{\text{Stage I}} + \underbrace{|V| \left(1 - \frac{1}{|V|}\right)^{|M|\times d+|C|\times m_{\min}}}_{\text{Stage II}} = |M| \,, \tag{4.4}$$

which can be approximated quickly with Newton's method as there are no analytical solutions.

We finally assemble the pieces to obtain the method which generates a set of communities satisfying all three properties:

1. Generate a random sequence $\{x_i\} \sim Pareto(I)(1, \alpha_c)$
2. Perform an affine transformation $y_i = \sigma_c(x_i - 1) + m_{\min}$, where $\sigma_c = \sigma' d = \frac{|M|d}{\sum(x_i - 1)}$, and round the result to obtain the number of members for each community (in stage I)
3. Uniformly select y_i members from all individuals into community i
4. Allocate individuals without an affiliation to community i with probability $y_i / \sum_{i \in C} y_i$

Note step 4 increases the number of members of all communities by $\frac{1}{1-q}$ times, where $q = \left(1 - \frac{1}{|V|}\right)^{|M|\times d+|C|\times m_{\min}}$ is the expected proportion of individuals who did not secure an affiliation in state I of membership/affiliation generation.

4.4 Graph Generation: First Attempt (AGM)

Having obtained the set of community memberships/affiliations, we can generate a network graph based on a community-based graph generative algorithm. We attempted to combine our overlapping community generative model with Yang and Leskovec's (2012) Community-Affiliation Graph Model (AGM), yet find it generate graphs with undesirable properties.

In this section we will show why the AGM is not an ideal candidate for generating real networks based on communities obtained in section 4.3. We observe an AGM graph is a mixture of Erdős–Rényi graphs, and show the node degree distribution of an AGM graph follows a mixed

[2] This is an overestimate: it assumes drawing with replacement in every stage I membership draw, yet the process we propose draws without replacement within a community, and with replacement between communities. The estimation error is small for large $|V|$ and small $|M|/|C|$. Better estimates may become infeasible with the introduction of d.

Poisson distribution, with a finite number of samples from the underlying distribution for mixture component parameters. The node degree distribution demonstrates exponential decay towards infinity, multimodality, and low probability mass near zero, which does not resemble the node degree distribution of real networks (e.g. that in figure 4.1). We also discuss the limiting behaviour for the mixed Poisson distribution with a Type IV Pareto distribution as its underlying mixing distribution on an AGM graph with an infinite number of communities.

4.4.1 Erdős–Rényi graph mixture and mixed Poisson distribution

To begin, we observe the AGM is a collection of Erdős–Rényi graphs. As discussed in section 3.2.1, individuals within the same community $C_i \in C$ connect with probability $p_{C_i} \in [0, 1]$ under the AGM, and each pair of individuals gains an independent chance to connect for each of their shared communities. This is equivalent to saying the AGM creates an independent Erdős–Rényi graph - $G_{ER}(|C_i|, p_{C_i})$ - for each community $C_i \in C$, where $|C_i|$ is the number of nodes, and p_{C_i} is the connection probability.

The node degree of Erdős–Rényi graphs follow the binomial distribution by lemma 4.1, which could be approximated by a Poisson distribution for large, sparse networks (i.e. large number of nodes $n \to \infty$ and small connection probability $p \to 0$) by Poisson limit theorem.

Lemma 4.1. *The node degree of a Erdős–Rényi graph $G_{ER}(n, p)$, where n is the number of nodes and p is the probability for two nodes to connect (independent of other connections), follows $Bin(n-1, p)$ - the binomial distribution with $n-1$ (independent) trials and probability of success p.*

Proof. Fix a node in $G_{ER}(n, p)$, the node has $n-1$ potential neighbours in the graph. For each potential neighbour, the node has probability p to connect independent of other connections/non-connections.

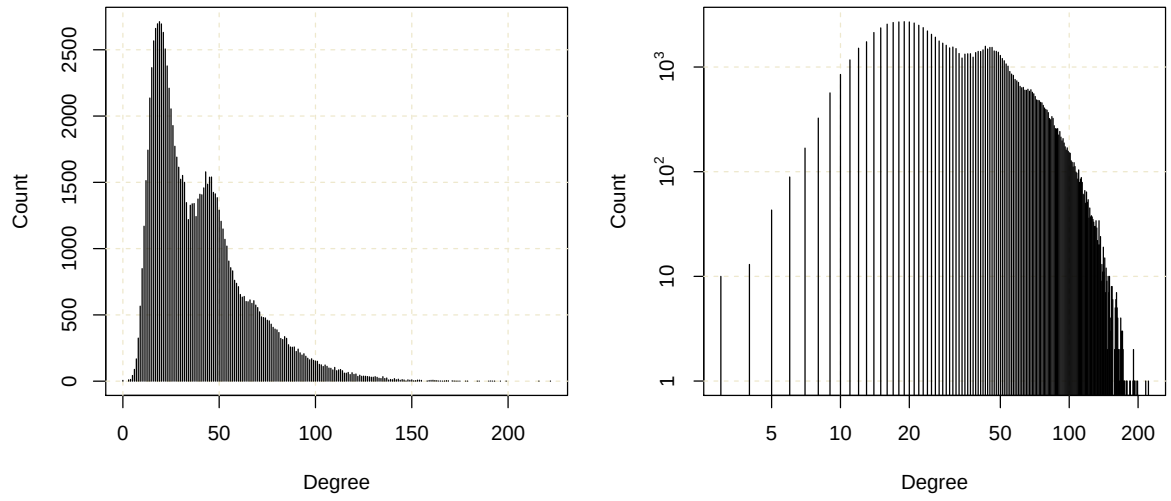
Hence the node will have $\binom{n-1}{k}$ ways to obtain k connections, each with probability $p^k(1-p)^k$. In other words,

$$\mathbb{P}(\text{Node degree} = k) = \binom{n-1}{k} p^k(1-p)^k \ , \ k \in \{0, 1, ..., n-1\} \tag{4.5}$$

This is by definition the binomial distribution $Bin(n-1, p)$. □

The observation leads to the conclusion that the node degree of an AGM graph follows a mixed Poisson distribution. Since the AGM is a mixture of overlapping Erdős–Rényi graphs, the node distribution of an AGM graph can be approximated by a mixed Poisson distribution: the normalised sum of Poisson distributions (with parameters λ_i following a mixing distribution) arising from different community-induced Erdős–Rényi graphs. This is evident in figure 4.6a, where each hump in the histogram represents a mixture component.

We will use the terms “node degree distribution of an AGM graph”, and “mixed Poisson Distribution” interchangeably throughout the section.



(a) Normal plot **(b)** Log-log plot

Figure 4.6: Degree distribution of a network generated by the AGM, in which underlying community affiliations are generated via our community generative model. The normal plot demonstrates “humps” in the degree distribution, which indicates the AGM generates networks with node degree following a mixture of Poisson distributions. The log-log plot shows the lack of individuals with a low degree, and an exponential decay in the number of individuals along increasing degree.

4.4.2 Properties of mixed Poisson distribution under finite mixture component parameters

Theorems 4.2 and 4.3 show a mixed Poisson distribution with finite mixture components is bounded by (a multiple of) the rightmost mixture component. Note theorem 4.3 makes stronger assumptions on the value of mixture component parameters, yet gives a stronger upper bound on the tail end of the mixed Poisson distribution.

Theorem 4.2. *Let X be a random variable under a mixed Poisson distribution with finite mixture components, each with parameter $0 < \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$. Then the probability density function (PDF) of X is bounded by the unnormalised probability density function of the rightmost mixture component, i.e.:*

$$f_X(x) = \mathbb{P}(X = x) \leq \frac{{\lambda_n}^x}{x!}, \quad \forall\, x \in \mathbb{N}\,. \tag{4.6}$$

Proof. The PDF of X is bounded by:

$$f_X(x) = \mathbb{P}(X = x)$$

$$
\begin{aligned}
&= \frac{1}{n}\left[\frac{e^{-\lambda_1}{\lambda_1}^x}{x!} + \frac{e^{-\lambda_2}{\lambda_2}^x}{x!} + ... + \frac{e^{-\lambda_n}{\lambda_n}^x}{x!}\right] \\
&= \frac{1}{nx!}\left[e^{-\lambda_1}e^{x\log\lambda_1} + e^{-\lambda_2}e^{x\log\lambda_2} + ... + e^{-\lambda_n}e^{x\log\lambda_n}\right] \\
&\leq \frac{1}{nx!}\left[e^{0}e^{x\log\lambda_1} + e^{0}e^{x\log\lambda_2} + ... + e^{0}e^{x\log\lambda_n}\right] \qquad (\text{as } e^0 \geq e^{-x}\ \forall x \geq 0) \\
&\leq \frac{e^0}{nx!}\left[e^{x\log\lambda_n} + e^{x\log\lambda_n} + ... + e^{x\log\lambda_n}\right] \\
&= \frac{n}{nx!}\left[e^{x\log\lambda_n}\right] \\
&= \frac{{\lambda_n}^x}{x!}\ . \qquad (4.7)
\end{aligned}
$$

□

Theorem 4.3. *Let X be a random variable under a mixed Poisson distribution with finite mixture components, each with parameter $0 < \lambda_1 \leq \lambda_2 \leq ... < \lambda_n$. Then for $x > \lambda_n$, the probability density function (PDF) of X is strictly bounded by the (normalised) probability density function of the rightmost mixture component, i.e.:*

$$f_X(x) = \mathbb{P}(X = x) < \frac{e^{-\lambda_n}{\lambda_n}^x}{x!}, \quad \forall\, x > \lambda_n,\ x \in \mathbb{N}\ . \qquad (4.8)$$

We first prove the following lemma:

Lemma 4.4. *Let $\lambda^* > 0$, then $x > \lambda^* \implies x\log\lambda^* - \lambda^* > x\log\lambda - \lambda \quad \forall\, 0 < \lambda < \lambda^*$.*

Proof. We first let

$$f(\lambda) = \frac{\lambda^* - \lambda}{\log\lambda^* - \log\lambda} \qquad (4.9)$$

$$\implies f'(\lambda) = \frac{1}{(\log\lambda^* - \log\lambda)^2}\left[1 + \frac{\lambda^*}{\lambda} - \log\frac{\lambda^*}{\lambda}\right] \geq 0 \quad \forall\, 0 < \lambda < \lambda^*\ . \qquad (4.10)$$

This means f is increasing for $0 < \lambda < \lambda^*$, i.e. $f(\lambda) < \text{“}f(\lambda^*)\text{”}\ \forall\, 0 < \lambda < \lambda^*$. Since $f(\lambda^*)$ is undefined, we take $\lim_{\delta\to 0} f(\lambda^* + \delta)$ as the upper bound instead:

$$
\begin{aligned}
\lim_{\delta\to 0} f(\lambda^* + \delta) &= \lim_{\delta\to 0} \frac{\lambda^* - (\lambda^* + \delta)}{\log(\frac{\lambda^*}{\lambda^*+\delta})} = \lim_{\delta\to 0} \frac{-\delta}{\log(1 - \frac{\delta}{\lambda^*+\delta})} \\
&= \lim_{\delta\to 0} \frac{-\delta}{-\frac{\delta}{\lambda^*+\delta} - O\left((\frac{\delta}{\lambda^*+\delta})^2\right)} \\
&= \frac{1}{\frac{1}{\lambda^*}} = \lambda^*\ . \qquad (4.11)
\end{aligned}
$$

Hence $\lambda^* > \dfrac{\lambda^* - \lambda}{\log \lambda^* - \log \lambda} \; \forall \, 0 < \lambda < \lambda^*$, and:

$$x > \lambda^* \implies x > \frac{\lambda^* - \lambda}{\log \lambda^* - \log \lambda} \iff x \log \lambda^* - \lambda^* > x \log \lambda - \lambda \quad \forall \, 0 < \lambda < \lambda^* \tag{4.12}$$

□

Proof of theorem 4.3. The PDF of X for $x > \lambda_n, x \in \mathbb{N}$ is given as:

$$\begin{aligned} f_X(x) &= \mathbb{P}(X = x) \\ &= \frac{1}{n}\left[\frac{e^{-\lambda_1}{\lambda_1}^x}{x!} + \frac{e^{-\lambda_2}{\lambda_2}^x}{x!} + ... + \frac{e^{-\lambda_n}{\lambda_n}^x}{x!}\right] \\ &= \frac{1}{nx!}\left[e^{x \log \lambda_1 - \lambda_1} + e^{x \log \lambda_2 - \lambda_2} + ... + e^{x \log \lambda_n - \lambda_n}\right] \\ &< \frac{1}{nx!}\left[e^{x \log \lambda_n - \lambda_n} + e^{x \log \lambda_n - \lambda_n} + ... + e^{x \log \lambda_n - \lambda_n}\right] \quad \text{(by lemma 4.4 , } x > \lambda_n\text{)} \\ &= \frac{1}{x!}\left[e^{x \log \lambda_n - \lambda_n}\right] \\ &= \frac{e^{-\lambda_n}{\lambda_n}^x}{x!} \end{aligned} \tag{4.13}$$

□

Since a Poisson distribution decays exponentially at the tail end, theorems 4.2 and 4.3 imply the mixed Poisson distribution also decays exponentially at the tail end (by the sandwich theorem). The exponential decay can also be observed in figure 4.6b.

A mixed Poisson distribution with finite mixture components may also demonstrate multimodality. Multimodality arises when successive elements in a sorted mixture parameter sequence are sufficiently apart, as illustrated in figure 4.7. This is in contravention of the degree distributions in figure 4.1, which appear smooth.

We also investigate properties of a mixed Poisson distribution near zero with the following results:

Theorem 4.5. *Let X be a random variable under a mixed Poisson distribution with finite mixture components, each with parameter* $0 < \lambda_1, \lambda_2, ..., \lambda_n$*. A sufficient condition for the probability mass function to be increasing from zero to* $k \in \mathbb{N} \setminus \{0\}$ *is* $\lambda_i > k \; \forall \, i \in \{1, ..., n\}$.

We begin by proving a lemma:

Lemma 4.6. *If* $\lambda > n, \;\; n \in \mathbb{N} \setminus \{0\}$*, then* $\dfrac{\lambda^n}{n!} - \dfrac{\lambda^{n-1}}{(n-1)!} > 0$.

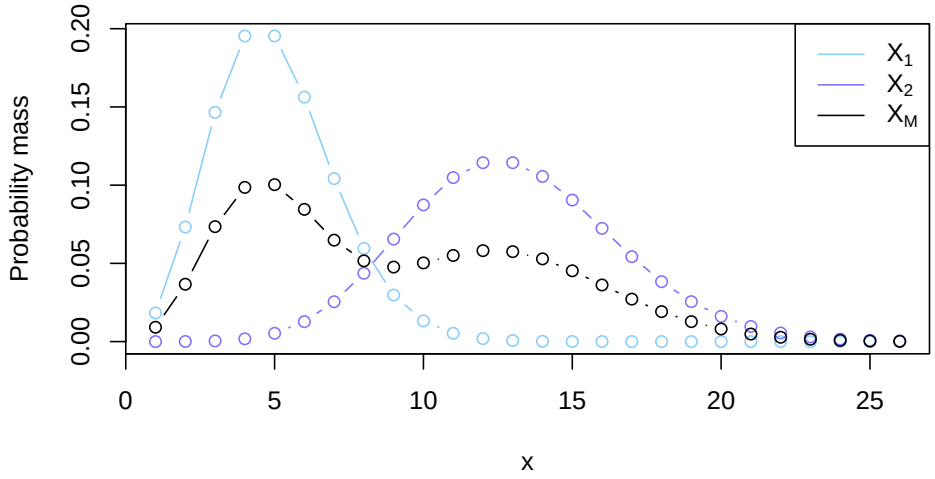


Figure 4.7: Plot of probability mass functions (PMF) for $X_1 \sim Po(4)$ (in light blue), $X_2 \sim Po(12)$ (in light purple), and $X_M = (X_1 + X_2)/2$ (in black), a mixed Poisson distribution with two components. PMF for X_M demonstrates multimodality, with one mode at $x = 5$ and another at $x = 12$.

Proof. Assume $\lambda > n \iff \dfrac{\lambda}{n} > 1$,

$$\frac{\lambda^n}{n!} - \frac{\lambda^{n-1}}{(n-1)!} = \frac{\lambda(\lambda^{n-1})}{n(n-1)!} - \frac{\lambda^{n-1}}{(n-1)!} = \frac{\lambda^{n-1}}{(n-1)!}\left[\frac{\lambda}{n} - 1\right] > 0 \tag{4.14}$$

□

Proof of theorem 4.5. Assume $\lambda_i > k \; \forall \, i \in \{1, ..., n\}$, consider the difference between $\mathbb{P}(X = k)$ and $\mathbb{P}(X = k - 1)$:

$$\begin{aligned}
&\mathbb{P}(X = k) - \mathbb{P}(X = k-1) \\
&= \frac{1}{n}\left[\frac{e^{-\lambda_1}{\lambda_1}^k}{k!} + \frac{e^{-\lambda_2}{\lambda_2}^k}{k!} + ... + \frac{e^{-\lambda_n}{\lambda_n}^k}{k!}\right] - \frac{1}{n}\left[\frac{e^{-\lambda_1}{\lambda_1}^{k-1}}{(k-1)!} + \frac{e^{-\lambda_2}{\lambda_2}^{k-1}}{(k-1)!} + ... + \frac{e^{-\lambda_n}{\lambda_n}^{k-1}}{(k-1)!}\right] \\
&= \frac{1}{n}\left[e^{-\lambda_1}\left(\frac{{\lambda_1}^k}{k!} - \frac{{\lambda_1}^{k-1}}{(k-1)!}\right) + e^{-\lambda_2}\left(\frac{{\lambda_2}^k}{k!} - \frac{{\lambda_2}^{k-1}}{(k-1)!}\right) + ... + e^{-\lambda_n}\left(\frac{{\lambda_n}^k}{k!} - \frac{{\lambda_n}^{k-1}}{(k-1)!}\right)\right]
\end{aligned} \tag{4.15}$$

$$> 0 \quad \text{(using lemma 4.6)} \tag{4.16}$$

Equivalently, $\mathbb{P}(X = k) > \mathbb{P}(X = k - 1)$. In general, for $j \in \{0, 1, ..., k - 1\}$:

$$\begin{aligned}
&\mathbb{P}(X = k-j) - \mathbb{P}(X = k-(j+1)) \\
&= \frac{1}{n}\left[e^{-\lambda_1}\left(\frac{{\lambda_1}^{k-j}}{(k-j)!} - \frac{{\lambda_1}^{k-(j+1)}}{(k-(j+1))!}\right) + e^{-\lambda_2}\left(\frac{{\lambda_2}^{k-j}}{(k-j)!} - \frac{{\lambda_2}^{k-(j+1)}}{(k-(j+1))!}\right) + ... + \right. \\
&\qquad \left. e^{-\lambda_n}\left(\frac{{\lambda_n}^{k-j}}{(k-j)!} - \frac{{\lambda_n}^{k-(j+1)}}{(k-(j+1))!}\right)\right]
\end{aligned} \tag{4.17}$$

$$> 0 \quad \text{(as } \lambda_i > k - j \ \ \forall\, i \in \{1, ..., n\}\text{, result follows by using lemma 4.6)} \tag{4.18}$$

Hence $\mathbb{P}(X = k - j) > \mathbb{P}(X = k - (j + 1))\ \forall\, j \in \{0, 1, ..., k - 1\}$, and we conclude $f_X(x) = \mathbb{P}(X = x)$ is a increasing function from zero to k. □

Corollary 4.7. *Let X be that in theorem 4.5, then the probability mass function (PMF) of X is increasing from zero to $\lfloor \min\{\lambda_1, \lambda_2, ..., \lambda_n\} \rfloor$, where $\lfloor . \rfloor$ denotes the floor function.*

Proof. This follows from the fact that $\lambda_i > \lfloor \min\{\lambda_1, \lambda_2, ..., \lambda_n\} \rfloor\ \forall\, i \in \{1, ..., n\}$, and the result of theorem 4.5. □

Remark. The PMF of X may still be increasing beyond $\lfloor \min\{\lambda_1, \lambda_2, ..., \lambda_n\} \rfloor$, though it is not necessarily the case.

Theorem 4.5 and its corollary 4.7 show the node degree distribution of an AGM graph starts from the bottom-left corner and has an initial increasing phase if $\lfloor \min\{\lambda_1, \lambda_2, ..., \lambda_n\} \rfloor \geq 1$. The condition is usually satisfied as an individual can expect to gain, on average, one or more connections from each community affiliation. This is in contravention of the degree distributions in figure 4.1, which starts from the top-left corner and appears decreasing.

4.4.3 Mixing distribution and limiting behaviour

To evaluate the full potential of the AGM as a generator of real networks, we also discuss the underlying distribution for the parameters λ_i, and the limiting behaviour of the mixed Poisson distribution (as if infinite samples are drawn from the mixing distribution).

We focus on the construction used in the implementation by Leskovec and Sosič (2014) [34], where

$$p_{C_i} = \sigma_A |C_i|^{-\alpha_A}, \quad \sigma_A, \alpha_A > 0 \tag{4.19}$$

$$\implies \lambda_i = |C_i|\, \sigma_A |C_i|^{-\alpha_A} = \sigma_A |C_i|^{(1-\alpha_A)}, \tag{4.20}$$

and show λ_i approximately follows a Type IV Pareto distribution:

Theorem 4.8. *Under some regularity conditions, λ_i in equation 4.20 approximately follows a Type IV Pareto Distribution.*

We begin by establishing the following two lemmas:

Lemma 4.9. *Let $X \sim Pareto(II)(\mu, \sigma, \alpha)$, then $kX \sim Pareto(II)(k\mu, k\sigma, \alpha)$ for constant $k > 0$.*

Proof. Consider the cumulative distribution function (CDF) for kX:

$$\mathbb{P}(kX \leq x) = \mathbb{P}\left(X \leq \frac{x}{k}\right) = 1 - \left(1 + \frac{\frac{x}{k} - \mu}{\sigma}\right)^{-\alpha} = 1 - \left(1 + \frac{x - k\mu}{k\sigma}\right)^{-\alpha}, \tag{4.21}$$

which is the CDF for $Pareto(II)(k\mu, k\sigma, \alpha)$. □

Remark. This shows type II Pareto distribution is scale-invariant (see theorem 2.7).

Lemma 4.10. *Let $Y \sim Pareto(II)(\mu, \sigma, \alpha)$, then $Y^{1-\beta}$ approximately follows a Type IV Pareto distribution under some regularity conditions.*

Proof. Consider the cumulative distribution function (CDF) for $Y^{1-\beta}$:

$$\begin{aligned}
\mathbb{P}\left(Y^{(1-\beta)} \leq y\right) = \mathbb{P}\left(Y \leq y^{\frac{1}{1-\beta}}\right) &= 1 - \left(1 + \frac{y^{\frac{1}{1-\beta}} - \mu}{\sigma}\right)^{-\alpha} \\
&= 1 - \left(1 + \frac{y^{\frac{1}{1-\beta}} - \left(\mu^{1-\beta}\right)^{\frac{1}{1-\beta}}}{\left(\sigma^{1-\beta}\right)^{\frac{1}{1-\beta}}}\right)^{-\alpha} \qquad (4.22) \\
&\approx 1 - \left(1 + \frac{\left(y - \mu^{1-\beta}\right)^{\frac{1}{1-\beta}}}{\left(\sigma^{1-\beta}\right)^{\frac{1}{1-\beta}}}\right)^{-\alpha} \\
&= 1 - \left(1 + \left(\frac{y - \mu^{1-\beta}}{\sigma^{1-\beta}}\right)^{\frac{1}{1-\beta}}\right)^{-\alpha}, \qquad (4.23)
\end{aligned}$$

which is the CDF for $Pareto(IV)(\mu^{1-\beta},\, \sigma^{1-\beta},\, \gamma = 1-\beta,\, \alpha)$. □

Remark. The approximation in equation 4.23 works well if $\mu \leq 1$ and β is not too close to one (a rule of thumb is $\beta < 0.85$).

Proof of theorem 4.8. Notice equation 4.20 can be rewritten as:

$$\lambda_i = \left(\sigma_A^{\frac{1}{1-\alpha_A}} |C_i|\right)^{(1-\alpha_A)}. \tag{4.24}$$

By construction, $|C_i| \sim Pareto(II)(\mu_c, \sigma_c, \alpha_c)$.

$\overset{\text{lemma 4.9}}{\Longrightarrow}$ $\sigma_A^{\frac{1}{1-\alpha_A}} |C_i| \sim Pareto(II)\left(\mu_c\, \sigma_A^{\frac{1}{1-\alpha_A}},\, \sigma_c\, \sigma_A^{\frac{1}{1-\alpha_A}},\, \alpha_c\right)$.

$\overset{\text{lemma 4.10}}{\Longrightarrow}$ $\lambda_i \overset{approx}{\sim} Pareto(IV)\left(\left(\mu_c\, \sigma_A^{\frac{1}{1-\alpha_A}}\right)^{(1-\alpha_A)},\, \left(\sigma_c\, \sigma_A^{\frac{1}{1-\alpha_A}}\right)^{(1-\alpha_A)},\, 1-\alpha_A,\, \alpha_c\right)$

$\iff$ $\lambda_i \overset{approx}{\sim} Pareto(IV)\left(\mu_c^{(1-\alpha_A)}\, \sigma_A,\, \sigma_c^{(1-\alpha_A)}\, \sigma_A,\, 1-\alpha_A,\, \alpha_c\right)$. □

Remark. The approximation works well if $\left(\mu_c \sigma_A^{\frac{1}{1-\alpha_A}}\right) \leq 1$, and α_A is not too close to one.

Theorem 4.8 implies the node degree distribution of an AGM graph demonstrates power-law behaviour as the number of communities tends to infinity. Willmot (1990, 1993) has shown a mixed Poisson distribution with mixture component parameters following a generalised Pareto distribution (the mixing distribution) follows the power law at the tail end [51][52]. The result follows from our ability to approximate the mixture component parameter as a Type IV Pareto distribution, a member of the generalised Pareto family of distributions.

Furthermore, the node degree distribution of an AGM graph does not demonstrate multimodality as the number of communities tends to infinity. Holgate (1970) has shown that a mixed Poisson distribution with mixture component parameters following a unimodal function is unimodal [24]. Moreover, Al-Zahrani and Sagor (2014) have shown a mixed Poisson distribution with mixture component parameters following a Lomax distribution (a variant of the Type II Pareto distribution) is either unimodal or decreasing, depending on the parameter values used for the Lomax distribution [3]. We expect the result to be similar with mixture component parameters approximately following a Type IV Pareto distribution, though we are unable to outline the exact conditions on distribution parameters that lead to a strictly decreasing probability mass function.

4.5 Graph Generation: An Influence-based Model

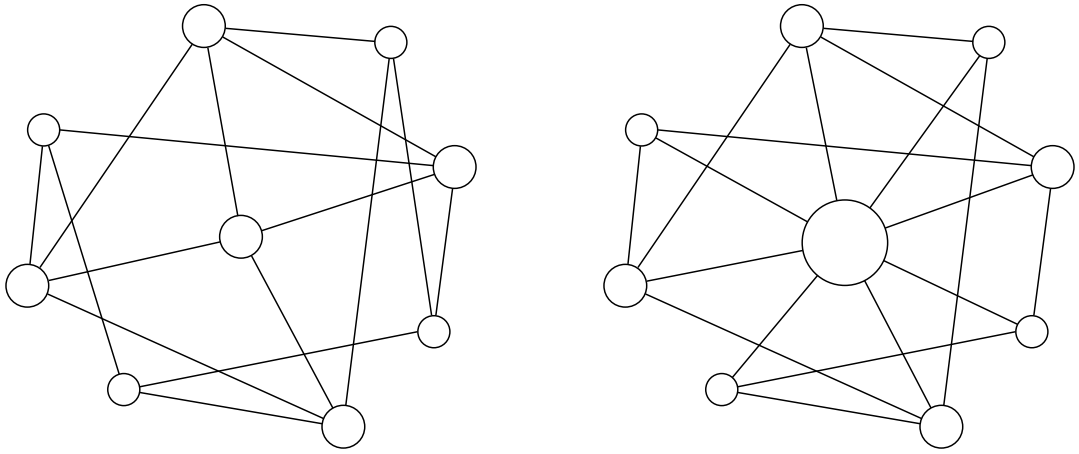
We then propose our own graph generative model to overcome deficiencies in network graphs generated by the AGM.

4.5.1 Affiliation Weights Matrix Generation

We base our model on the concept of *influence*, a relative measurement to gauge an individual's influence, engagement and participation within a network. We believe the higher an individual's influence is on a community, the more likely it is for the individual to connect with other individuals in the community. In the case of multiple affiliations, an individual can "split" their influence over their affiliations. This reflects the observation that an individual can have different degrees of participation in different affiliated communities: A researcher can work in the intersection of Computer Science and Social Science, with a slant to the former.

Similar to the number of connections an individual has, the influence of an individual across all their affiliation(s) resembles the 80-20 rule. Few individuals (e.g. Barack Obama, David Beckham, Justin Bieber) have great influence overall, and the absolute majority have little influence. Note a community can have zero or more individuals with high overall influence, and high overall influence is not a prerequisite for an individual to gain connections within a

community - the measure only governs how individual-centric the graph within the community will be. To illustrate (see figure 4.8), connections between a group of schoolchildren with low but comparable influence may resemble a random graph, whereas the existence of a "sports star" or "popular kid" in school may draw attention and connections towards him/her. The person may not have overall influence comparable to that of Barack Obama's, though it would be higher than other schoolchildren.



(a) Comparable influence between individuals **(b)** Unbalanced influence between individuals

Figure 4.8: Possible connections between a community of nine individuals under different influence distributions. The size of the node is proportional to the influence of an individual, which controls how many connections the individual obtains within the community. Note in the unbalanced influence distribution case, the centre node (with extremely high influence) gains a connection with all other nodes.

We then link our concept of influence to Yang and Leskovec's (2013) concept of affiliation weights in BigClam by initialising the affiliation weights matrix $F = (F_{uc})$ used in BigClam as follows. For each individual in the network, their total affiliation weight is sampled from a different Pareto Type II distribution (i.e. $\sum_c F_{uc} \overset{\text{iid}}{\sim} Pareto(II)(\mu_w, \sigma_w, \alpha_w)$ for specified μ_w, σ_w and α_w). For each individual with multiple, say k, affiliations, we randomly partition their total affiliation weight into k parts (of different sizes) and assign them to each affiliation (see figure 4.9). This is done by generating $k-1$ ordered standard uniform ($U(0,1)$) samples, and can be made efficient by using the uniform spacing algorithm proposed by Schucany (1971) [44].

4.5.2 Edge Formation

We finally proceed with a variant of Yang and Leskovec's (2013) BigClam generative model, which states two nodes u, v connect with probability

$$p(u,v) = \sigma_e(1 - \exp(-{F_u}^T F_v)) + \epsilon \,, \tag{4.25}$$

where $F_u = (F_{uc})$, $F_v = (F_{vc})$ are the affiliation weights (*column*) vectors associated with

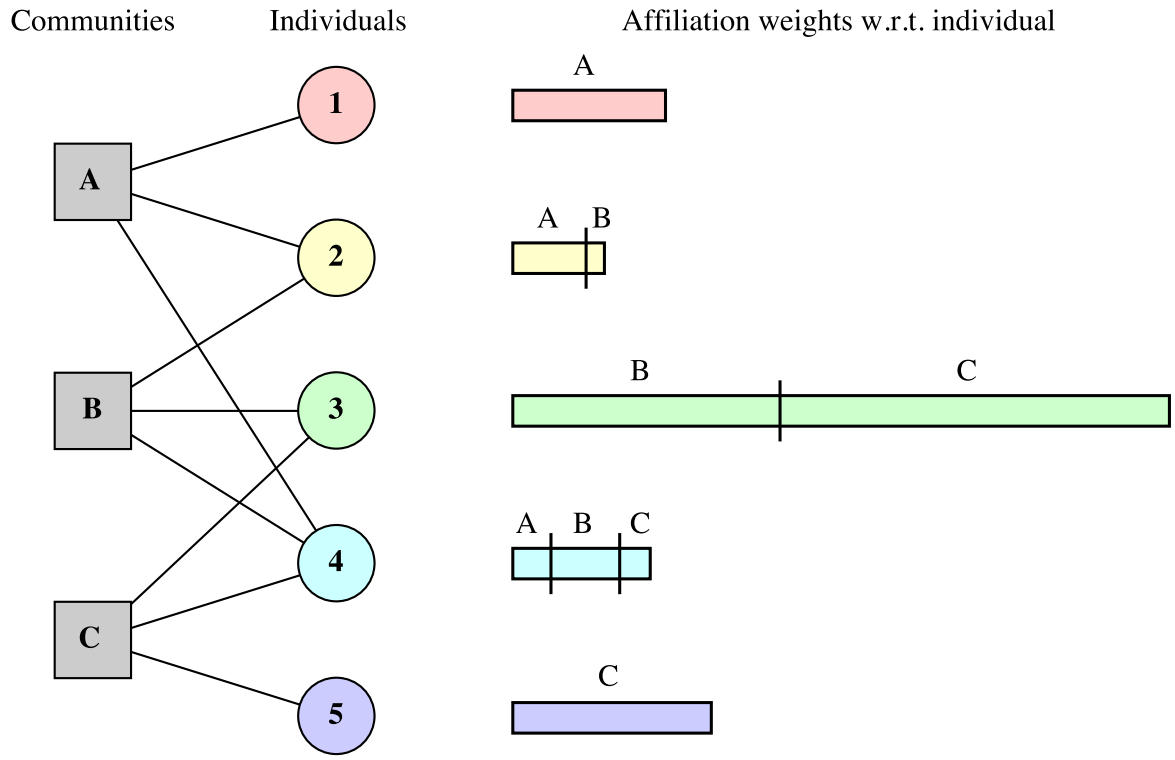


Figure 4.9: A network with five individuals and three communities, featuring each individual's total affiliation weights (with magnitude denoted by length of the bars on the right) and how it is split between their affiliation(s) (denoted by position of the markers).

individual u and v respectively, and σ_e is the scale factor governing the total number of edges being generated. There is a small chance of two nodes connecting "at random", reflected by the $\epsilon > 0$ in equation 4.25.

Yang and Leskovec (2013) have also offered an alternate construction, stating a pair of nodes (u, v) have multiple independent chances $(1 - \exp(-F_{uc} \times F_{vc}))$ for each common community c) of connecting if both nodes are affiliated to more than one community [54]. In other words:

$$p(u,v) = \sigma_e \Bigg(\underbrace{1 - \prod_{c \in C^*} (1 - \underbrace{(1 - \exp(-F_{uc} \times F_{vc}))}_{\text{Prob. } u,v \text{ connects via } c})}_{\text{Prob. } u,v \text{ does not connect } \forall c} \Bigg) + \epsilon \,, \tag{4.26}$$

where C^* is the set of communities with both u and v as members, and ϵ is a small quantity.

This leads to an implementation which need not consider all possible node pairs, but only node pairs within each community. All node pairs without common communities can only connect "at random" from the construction above, and hence can be ignored when we generate community-motivated connections. It can be shown that the alternate construction (equation 4.26) will achieve a $\frac{|C|^2}{r^2}$ times speed-up compared to the initial construction (equation 4.25), where $|C|$ is the number of communities and r is the average number of affiliations per individual.

4.6 Analysis and Experimental Results

We show our community and community-based graph generation processes are capable of producing networks with desirable properties. This is done by proving our community generation process generates communities with the number of members following the power law, and individuals with exponential decay in number of affiliations. Moreover, we present experimental results which demonstrate our community-based graph generation process can generate graphs with node degree distribution following the power law.

4.6.1 Distribution of Number of Community Membership

By construction, the number of memberships follows a Type II Pareto distribution:

Theorem 4.11. *Under our community generative model, the number of memberships for community i, $|C_i|$, is distributed as:*

$$|C_i| \sim Pareto(II)\left(\frac{1}{1-q}m_{\min},\ \frac{1}{1-q}\sigma_c,\ \alpha_c\right), \tag{4.27}$$

where $q = \left(1 - \frac{1}{|V|}\right)^{|M|\times d+|C|\times m_{\min}}$ is the proportion of individuals who did not secure an affiliation in stage I of membership/affiliation generation.

Proof. Recall the community generative model determines the number of memberships by first generating $x_i \sim Pareto(I)(1, \alpha_c)$.

By theorem 2.12, $y_i = \sigma_c(x_i - 1) + m_{\min} \implies y_i \sim Pareto(II)(m_{\min},\ \sigma_c,\ \alpha_c)$.

By lemma 4.9, $|C_i| = \frac{1}{1-q}\, y_i \implies |C_i| \sim Pareto(II)\left(\frac{1}{1-q}\, m_{\min},\ \frac{1}{1-q}\, \sigma_c,\ \alpha_c\right)$. □

Together with the result of theorem 2.11, we conclude the number of membership asymptotically follows the power law. This can be visualised in figure 4.10b, where the number of communities with a certain size shows a linear relationship with the size on a log-log plot.

4.6.2 Distribution of Number of Individual Affiliations

The number of affiliations of an individual follows a Modified Poisson's Binomial distribution defined as follows:

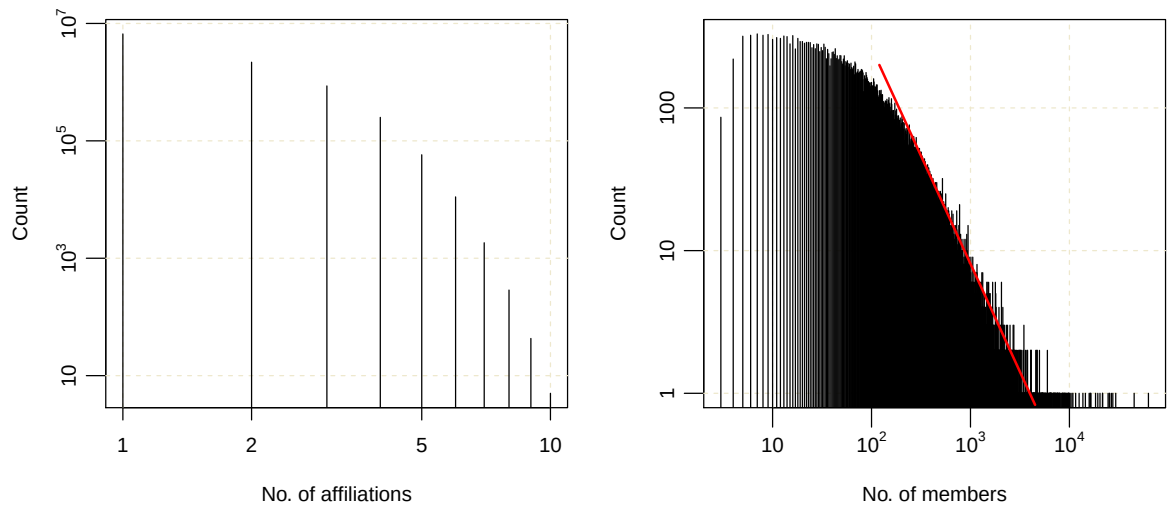
Definition 4.12. (Modified Poisson's Binomial Distribution)

Let X be a Poisson's Binomial random variable as defined in section 2.3.3, then Y is a Modified Poisson's Binomial Distribution (with support $\{1, 2, ...\}$) if:

$$\mathbb{P}(Y = y) = \begin{cases} \mathbb{P}(X = 0) + \mathbb{P}(X = 1) & y = 1 \\ \mathbb{P}(X = y) & y \in \{2, 3, ...\} \\ 0 & \text{otherwise} \end{cases} . \tag{4.28}$$

The distribution arises from how individuals are affiliated in our community generation process (see section 4.3.2). In our membership/affiliation generation process, each community $C_i \in C$ independently, uniformly selects $|C_i|$ individuals from V (the set of all individuals) as its members. From an individual's perspective, the process means the individual has a independent probability of $|C_i|/|V|$ of being affiliated to community i. If we treat forming an affiliation as a "success" for the individual, the total number of "successes" (i.e. number of affiliations) for the individual follows the standard Poisson's Binomial distribution. Providing individuals without any affiliations an affiliation results in the modified Poisson's Binomial distribution.

Poisson's Binomial distribution decays exponentially towards the tail end. This results from Wang's (1993) work on Poisson's Binomial distribution being bounded above by a Poisson distribution [47], and Poisson distribution decays exponentially at the tail end. The exponential decay is shown in figure 4.10a, which we are unable to establish a linear relationship in the log-log plot.



(a) Number of affiliations per individual

(b) Number of memberships per community

Figure 4.10: Aggregated affiliation/ membership distribution for 100 synthesised networks with 100,000 individuals and 500 communities. The log-log plots shows an exponential decay in distribution of number of affiliations, and number of members being asymptotically power-law distributed (denoted by the red line on the right).

4.6.3 Node Degree Distribution

We also study the node degree distribution of the graphs generated by our graph generative process, which shows promising results - most of the degree distribution appears asymptotically linear under a log-log plot, suggesting it follows the power law.

We further conduct an in-depth study on the effect of varying numbers of communities on node degree distribution. We generate medium-sized networks with 500,000 individuals with varying numbers of communities, in which communities have a moderate degree of overlapping (with r, the average number of affiliations per individual, as 1.5) and size disparity (with α_c, the shape parameter of membership distribution, as 3.0). The node degree distribution for each synthesised network is shown in figure 4.11.

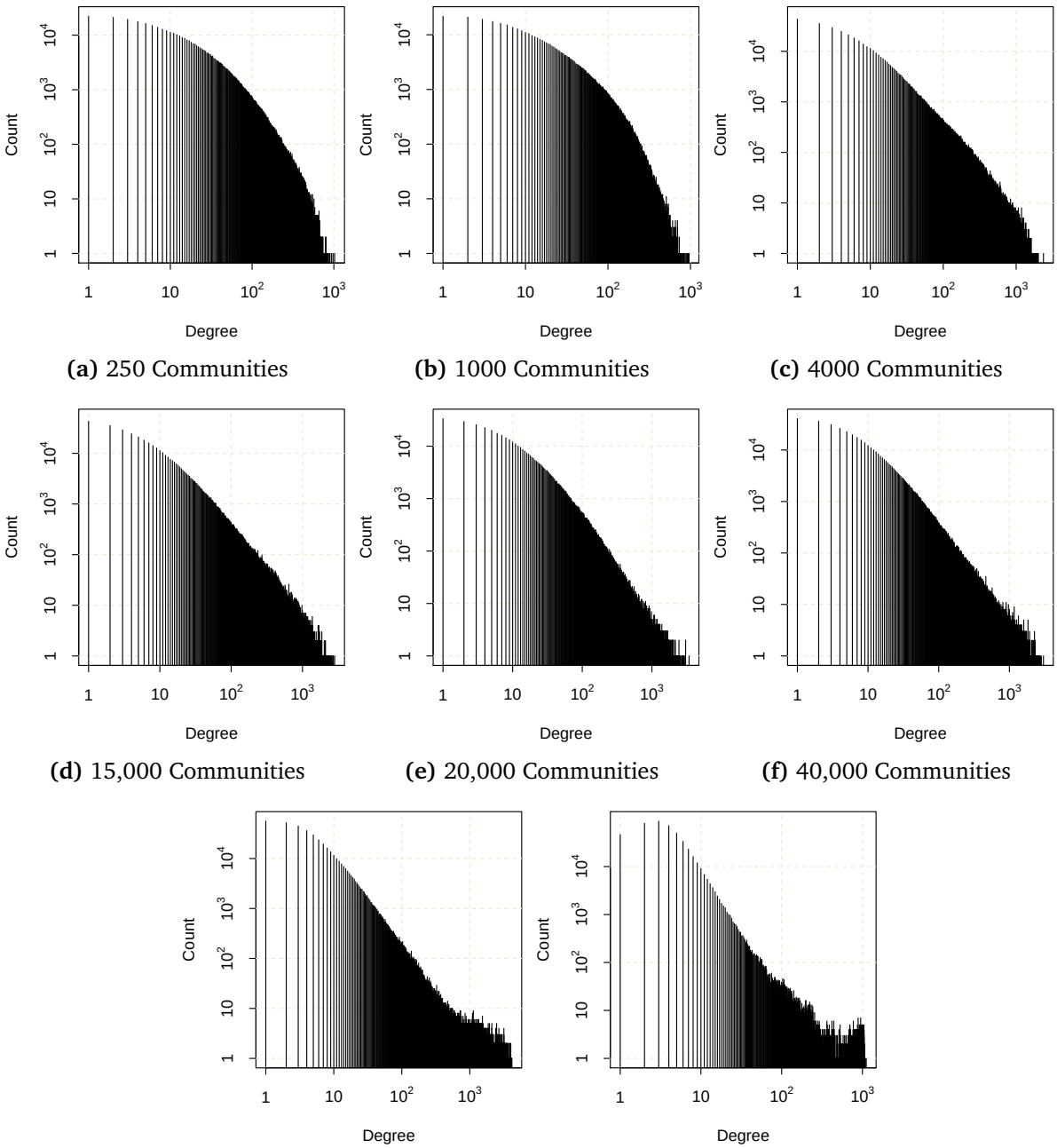
The node degree distributions demonstrate different characteristics under different numbers of communities, suggesting there might be an optimal range on the number of communities a network can support. For a low number of communities, the node degree distribution (figures 4.11a, 4.11b) appears to decay exponentially. The "concavity" of the node degree distribution on a log-log plot decreases with an increasing number of communities (figure 4.11c), and demonstrates asymptotic linearity when the number of communities is between 15,000 and 40,000 (figures 4.11d, 4.11e, 4.11f). Further increasing the number of communities leads to irregular behaviour in node degree distribution under the mentioned parameters. On a log-log plot the distribution appears linear in the middle part, with a "bulge" towards the (right) tail of the distribution (figures 4.11g, 4.11h). The irregular behaviour is further discussed in section 4.7.3.

4.7 Evaluation and Discussion

In this section we provide an in-depth evaluation on the strengths and weaknesses of our graph generative process. We also discuss some implications arising from our model.

4.7.1 Contribution and Impact

Our overlapping community-based graph generative process is one of the first models which accommodates both overlapping community structures and power-law distributed node degrees - Todeschini and Caron (2016) have independently come up with the idea of generating graphs with power-law distributed node degrees during the course of this project, using the Generalised Gamma Process to generate the affiliation weights matrix [45]. Graphs generated under our graph generative process carry a higher value in being studied compared to other models due to their higher resemblance in node degree distribution and overlapping community structures to real networks. Table 4.1 summarises the capabilities of different models.



(a) 250 Communities **(b)** 1000 Communities **(c)** 4000 Communities

(d) 15,000 Communities **(e)** 20,000 Communities **(f)** 40,000 Communities

(g) 75,000 Communities **(h)** 150,000 Communities

Figure 4.11: Node degree distribution for networks synthesised by our overlapping community-based generative process. The networks have 500,000 individuals and varying numbers of communities, with moderate community overlap ($r = 1.5$) and community size disparity ($\alpha = 3$).

We also offered an explanation of why individuals connect with each other throughout our model construction. Our observation - influential and engaged individuals tend to obtain more connections within their communities - brings a flavour of the Barabási–Albert Model to an overlapping community setting. This results in our model demonstrating both an overlapping community structure and the scale-free property in generated graphs.

	Community structure	*Overlapping* community structure	Scale-free
Erdős–Rényi Model	✗	✗	✗
Watts–Strogatz Model	✗	✗	✗
Barabási–Albert Model	✗	✗	✓
Stochastic Block Model	✓	✗	?
Our overlapping community + AGM graph gen. model	✓	✓	✗
Our overlapping community + influence-based graph gen. model	✓	✓	✓

Table 4.1: Comparison of our overlapping community-based graph generative model with models studied in section 3.1, based on their capability of accommodating a(n) (overlapping) community structure and the scale-free property (i.e. node degree following a power law distribution).

Furthermore, the ability of our graph generative model to generate real network-like graphs opens up the opportunity for us to quantitatively evaluate overlapping community detection algorithms. Researchers no longer need to depend solely on real datasets, which are limited in number and require significant effort in compiling community affiliations, to tell the quality of their algorithm(s). We took advantage of such abilities to evaluate, and ultimately enhance, Yang and Leskovec's (2013) BigClam community detection algorithm in chapter 5.1.

4.7.2 Limitations

Our graph generative model is originally designed to evaluate the accuracy and scalability of overlapping community detection algorithms (e.g. BigClam, AGM). As a result, we made a few assumptions and design choices that is now considered restrictive. These include the assumption that every individual should have at least one community affiliation, the "requirement" that the actual number of edges and membership/affiliation generated should be as close as possible to that specified by the parameters.

The assumptions and design choices lead to two classes of restrictions - a relatively narrow, albeit common, range of parameter values for our graph generative process to work properly, and the difficulty in carrying out inferences based on our graph generative process.

Some parameter combinations will create graphs that do not match our specifications in section 4.2. For example if $|M| = |V| \times r < |C| \times m_{\min}$ [3], then the algorithm will generate $|C| \times m_{\min}$, instead of $|M|$ memberships/affiliations. This happens as the rule on the minimum number of members in a community takes precedence in implementation. Likewise, it may not be possible to generate networks with $|E|$ edges if $|C|$ is too large, where the network becomes too sparse for a sufficient number of edges to form (see section 4.7.3 for further discussion).

[3] Where $|M|$ is the number of memberships, $|V|$ is the number of individuals, r is the average number of individual affiliations, $|C|$ is the number of communities, and $m_{\min}$ is the minimum number of members in a community.

The design choices motivate the need to scale quantities based on samples drawn from a distribution, which makes inference difficult. Recall our scaling parameter in overlapping community generation (see section 4.3), $\sigma_c = \frac{|M|d}{\sum(x_i - 1)}$ (where $|M|$ is the (specified) number of memberships, and d is the discount factor introduced in section 4.3.3), depends on x_i, samples generated from a Type I Pareto distribution. As the samples are random in nature, but not random variables following some probability distributions themselves, the learning problem "Given the community memberships/affiliations, what should the specified parameters (e.g. σ_c) be" will not have tractable solutions - we can only obtain crude estimates for the parameters.

Further work is required to remove said restrictions. We discuss some of the possible approaches in section 4.8.1. This may lead to us losing control of fine structural details, but a simpler and more flexible model.

4.7.3 Implied Limit on Community Structure

From section 4.6.3, we see for large numbers of communities, the node degree distribution gives a "bulge" at its tail end. We also observe the graph generative process is unable to generate the specified number of edges if the number of communities is too large.

We believe this implies a limit on real networks' community structure, more specifically the number of communities a network can support given the degree of overlap. Further increasing the number of communities in a network will lead to smaller communities - a greater number of communities leads to a greater sum of Type I Pareto samples introduced in equation 4.2 ($\sum_{i=1}^{|C|} x_i$), which leads to a smaller scale parameter for the distribution of the number of community memberships $\sigma_c = \frac{|M|d}{\sum(x_i - 1)}$. Hence individuals, regardless of their influence value, have a limited number of potential neighbours to connect with within a large number of communities, as an individual is only likely to connect with individuals who share the same community affiliation(s) under our graph generative process. This means the number of possible edges is limited, and even a graph with fully connected communities may have its edge count not meeting the specified target $|E|$ (see figure 4.12).

We also provide the missing link between excessively large number of communities and the irregularities in node degree distribution (see section 4.6.3). The graph generative process attempts to increase the connection probability if the number of edges present is less than the target count. This is done by increasing the scale parameter for edge generation σ_g, expecting more edges to be formed. For $\sigma_g \gg 1$, the probability of any individual connecting with individuals who share the same community affiliation will be one or close to one, and hence the node degree distribution now depends on the size of each community. Notice in this degenerate case, larger communities have more members, and each member has a larger chance of connecting with a large number of individuals compared to their counterparts affiliated to smaller communities. This results in a surge in the node degree distribution at the tail end, explaining the "bulge" seen in figure 4.11h.

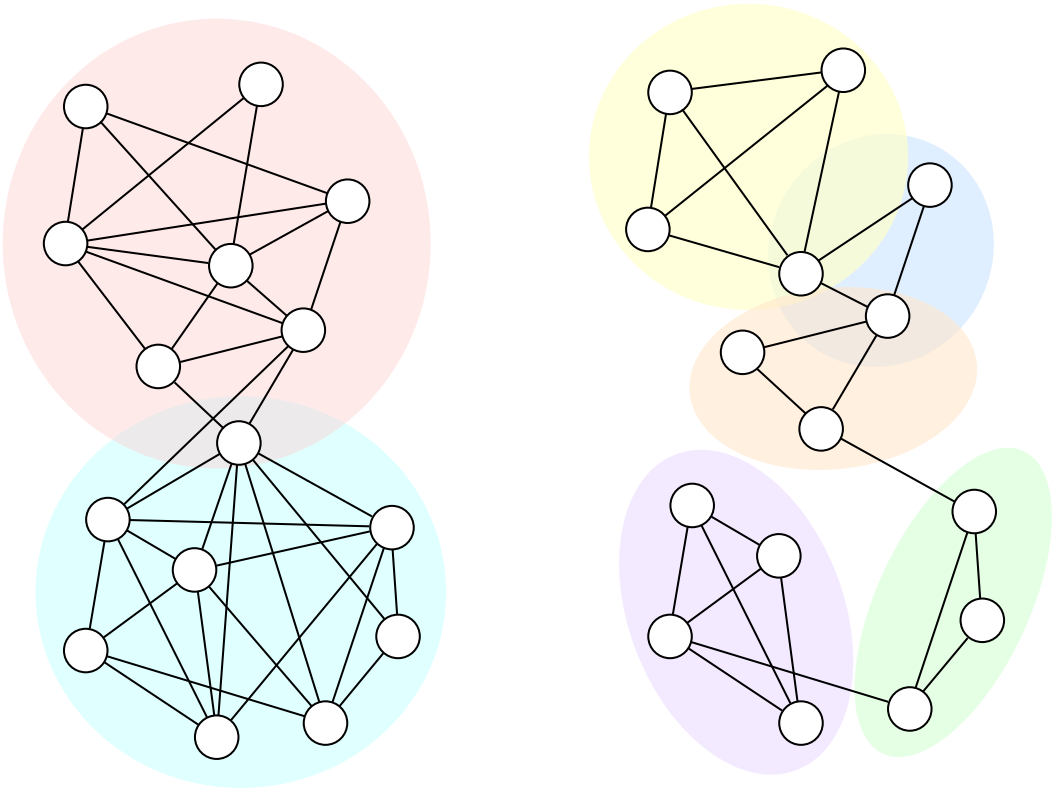


Figure 4.12: Two 15-node graphs with different numbers of underlying communities (in coloured ellipses). The network on the left has two larger communities, and hence allows more edges (37 in this graph) to be formed under our graph generative process. The network on the right has more (five) but smaller communities, and while it is maximal within each community, it only features 23 edges.

A simple workaround to the problem is to increase the degree of overlap (indicated by the average number of affiliations per individual, r) between communities. Increasing the degree of overlapping allows us to admit more communities, and further explore hierarchical community structures (sub-communities within a community) within social networks. On the other hand, it also means more complex edge generation and community detection (due to a more complex community structure), and a balance should be struck in between when deciding how many communities we would like to admit in our model.

4.8 Future work

We conclude by outlining possible work opened up by our overlapping community-based graph generative process.

4.8.1 Relaxing Assumptions on Distributions and Their Parameters

Our overlapping community-based graph generative model involves specifying ten distinct parameters (including the five specified in section 4.2, and the location, scale and shape parameters used in sections 4.3 and 4.5), and can be simplified.

We look at the possibility of relaxing assumptions on distributions and their parameters via the use of non-parametric models and inference methods. Non-parametric models learn the parameters from data, and hence less parameters are required to be specified. The models also remove the need for sample-dependent discount factor(s), which prevent one from carrying out inference. Possible candidates for generating the affiliation weights matrix include the Indian Buffet Process (IBP) outlined by Griffiths and Ghahramani (2011) [23], which is capable of generating binary random matrices representing existence of memberships/affiliations between individuals and communities.

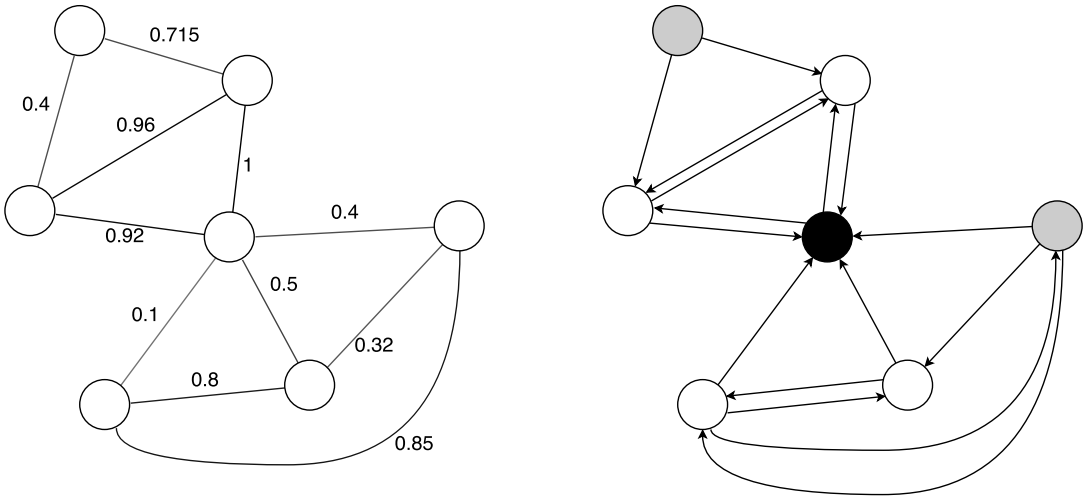
The Generalised Gamma Process (GGP) outlined by Caron and Fox (2014) [10] has been used by Todeschini and Caron (2016) to generate random matrices for affiliation weights [45]. Further work is required to verify if the properties of GGP is sound.

4.8.2 Generating Other Types of Networks

We also seek the possibility of extending our graph generative process to accommodate other types of graphs. Examples include, in increasing perceived difficulty, weighted, directed, and temporal graphs.

Weighted networks (see figure 4.13a) allow one to specify the interaction strength between individuals. A possible approach to generating weighted networks is to modify the edge generation process - instead of generating an edge with probability p, we generate the edge with weight p if p is higher than some threshold. We have to be careful as this process may result in dense graphs (graphs having too many edges), which is considered an undesirable property in graph-generative models by Todeschini and Caron (2016) [45].

Directed networks (see figure 4.13b) support asymmetrical connections. For example, in the Twitter social network, following another account and being followed by another account are two distinct concepts. Similarly, in a telephone call network, an outgoing call would almost certainly not be matched by an incoming call; yet most of the methods continue to model individual-individual connections as an undirected graph (e.g. Blondel et al. (2008) [7], Yang and Leskovec (2012, 2013) [53] [54]). A model that can explain individuals with a large number of incoming and/or outgoing edges is preferred. This may be done by introducing two sets of influence/participation measurements, one for incoming connections (highly influential individuals attract connections from others) and the other for outgoing connections (heavily engaged individuals connect with a large number of other individuals).



(a) Weighted graph **(b)** Directed graph

Figure 4.13: Illustration of a weighted graph (left) and a directed graph (right). The number beside a weighted graph edge shows the weight of the edge, which can be interpreted as the interaction strength between two individuals. The arrows in the directed graph show the edge directions, which can be interpreted as asymmetrical connections. Nodes with significantly more incoming edges in the directed graph are coloured in black, and nodes with significantly more outgoing edges in the directed graph are coloured in grey.

Temporal networks take dynamics between individuals across time into account. It is almost certain that any influential individuals will have a diminished influence measurement given sufficient time, due to reasons such end-of-office, retirement, and death. Individuals can also experience changes in affiliations (e.g. interest, country/area of residence, school), and their influence, hence connections may also change accordingly. Hence the ability to generate temporal networks will enable the study of changes to an individual's surroundings and social interactions. An interesting application would be the detection of social withdrawal, which with appropriate intervention could potentially save lives. Further work is required to study temporal dynamics within social networks before any feasible models can be proposed.

Chapter 5

BigClam: Verification Framework Extension and Runtime Reduction

We turn our attention to the Cluster Affiliation Model for Big Networks (BigClam), a popular overlapping community detection algorithm proposed by Yang and Leskovec (2013) [54].

In their paper, Yang and Leskovec (2013) conducted some experiments on BigClam's prediction accuracy and runtime [54], though the technical details were largely omitted, presumably for brevity. We are interested in verifying the experiments, and further expanding the verification framework for BigClam to explore the community detection algorithm's behaviour under different network characteristics. As such, we learn from experiment results that BigClam achieves a higher prediction accuracy on networks with less overlap and dispersion in community sizes. Furthermore, we determine the runtime complexity of the BigClam community detection algorithm, which is divided into three stages: the initialisation/conductance test stage, the gradient ascent stage, and the community association stage.

We also notice from experiment results that the implementation of the BigClam community detection algorithm spends the majority of its runtime on the community association stage when solving networks with a large number of communities, and yet computation is not parallelised across CPU threads. This results in lengthened runtime and waste available hardware resources. We parallelise the community association stage with OpenMP, and show the parallelisation achieves as much as 5.3 times speed up and saves as much as 12.8 hours in the community association stage when solving networks by Leskovec and Krevl (2014) [33].

We organise the chapter as follows. Section 5.1 covers our work on extending the verification framework for overlapping community detection algorithms, and analysing BigClam's accuracy and runtime under different network parameters. Section 5.2 outlines our work on speeding up BigClam, and comparison of our work with Leskovec and Sosič's (2014) implementation on the Stanford Network Analysis Platform (SNAP) [34]. We finally evaluate and discuss the impact of our work and future work this has opened up in section 5.3.

5.1 Verification Framework Extension

5.1.1 Motivation

Yang and Leskovec (2013) conducted a series of experiments on BigClam's predictive ability in their paper [54], though they did not explain the details of the experiments. One of the experiments involved generating synthetic networks under the AGM generative model (see section 3.2.1), fitting the generated networks with the BigClam community detection algorithm to obtain predicted networks, and comparing the predicted networks with the generated networks to infer BigClam's ability in recovering community affiliations: the more similarities between the predicted and generated networks, the better BigClam is. We observed Yang and Leskovec did not specify any details on the synthetic networks, or their underlying community structures used in their experiment, to reach the conclusion that BigClam community detection algorithm achieves high prediction accuracy.

A similar lack of details can be observed when Yang and Leskovec evaluate the runtime of BigClam. Yang and Leskovec compared runtime for BigClam against runtime for other community detection algorithms on synthetic networks of different size, yet details of the synthetic networks, part from number of nodes in the networks are not reported. Moreover, we observed when Yang and Leskovec commented on applying BigClam on real ground-truth networks, they focused on its improvement in prediction accuracy compared with other community detection methods, with very little detail on BigClam's runtime on the mentioned networks.

The observations above open up the following opportunities:

Verification of Experiments by Yang and Leskovec (2013) [54] We would like to verify if BigClam is *the* overlapping community detection algorithm we are looking for as claimed by Yang and Leskovec (2013) [54] - if it produces accurate community predictions and scales. By attempting to reproduce the experiments (filling in assumptions left blank by Yang and Leskovec), we may be able to verify Yang and Leskovec's claim or identify opportunities to further improve current work.

A Verification Framework for Overlapping Community Detection Algorithms An extended verification framework, which takes a few network characteristics into account, can be used to evaluate BigClam's performace under various networks. It allows us to explore the effect of overlap between communities and dispersion between community sizes on BigClam's prediction accuracy, and the number of nodes, edges and communities present in the network on BigClam's runtime. Ideally the verification framework can be used to evaluate any overlapping community detection algorithms as is.

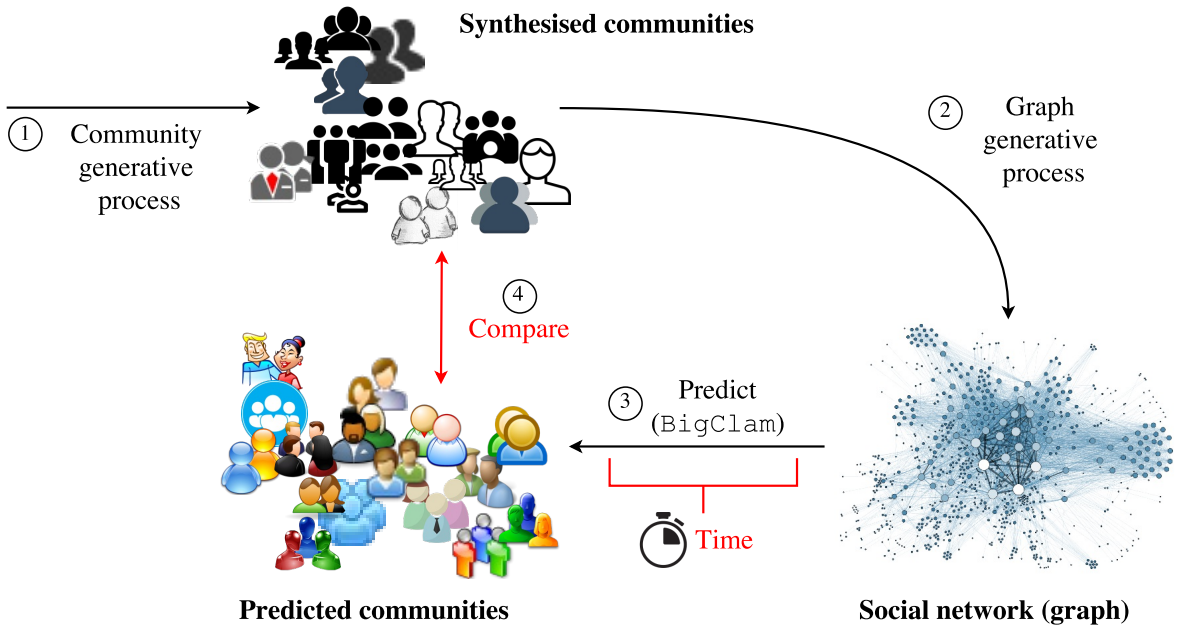


Figure 5.1: Method in verifying Yang and Leskovec's experiment on BigClam: (a) accuracy in detecting overlapping communities and (b) scalability on large networks.

5.1.2 Methodology

Our verification of BigClam first synthesise a set of (overlapping) communities with our community generative process and a graph with our graph generative process outlined in chapter 4. It then obtain a predicted set of communities with BigClam community detection algorithm (see section 3.2.2) from the graph.

We obtain the following metrics throughout. We measure the accuracy of the BigClam community detection algorithm by various similarity measures between the synthesised and predicted communities, and infer the scalability of the method by retrieving the time taken to obtain the predicted communities from the graph. Figure 5.1 illustrates our verification process, details of which are available in sections below.

Test Parameters

We are interested in the effect of the parameters below on the accuracy and scalability of BigClam. All parameters below were featured and used in chapter 4.

- **Number of individuals** ($|V|$):
  It is used by Yang and Leskovec as a controlled parameter to compare runtime between BigClam and other community detection methods. We are interested if this is the sole factor on runtime.
- **Number of communities** ($|C|$):

It determines the size of the underlying affiliation weights matrix under a BigClam model (jointly with $|V|$). We suspect a large $|C|$ might lead to increased, possibly a prohibitively large, runtime.

- **Average number of edges per individual** ($\frac{|\hat{E}|}{|V|}$):
  Also known as average degree per node on a graph setting, it approximates the number of edges required to be processed by BigClam community detection method. We are interested if large $\frac{|\hat{E}|}{|V|}$ would lead to prohibitively large runtime.
- **Average number of community affiliation for each individual** (r):
  It approximates the degree of overlap between communities - a small r yields a set of near-disjoint (non-overlapping) communities, whereas a large r yields a set of densely overlapped communities (see figure 4.4). We are interested if increased overlap between communities would affect the community detection method's accuracy.
- **Power law distribution parameter on community size** (α):
  It determines the distribution on number of membership for each community - a small α would lead to a wider range of numbers of memberships, whereas a large α would lead to a more consistent range of membership number between communities. We are interested in BigClam's ability to detect both massive and tiny communities.

Run Procedure

We run the following procedures for each combination of test parameters listed above. The test parameters is sampled from the space of possible values.

1. **Synthetic communities generation**:

   We first generate a set of communities using our community generative process as outlined in section 4.3. The communities are represented in a SNAP library-compatible [34] format, such that it can be used by the library functions.

2. **Graph generation**:

   We generate a graph based on the set of communities obtained in step 1 using our influence-based graph generative process in section 4.5.[1]

3. **Community detection with BigClam**:

   We then run the SNAP implementation of BigClam community detection method (`BigClam`) on the generated network to obtain a set of predicted communities. We also obtain some time metrics throughout stages of the method to infer its scalability, which details can be found below.

[1] While Yang and Leskovec generated a number of synthetic networks with the Affiliation Graph Model (AGM) generative method, we have shown in section 4.4 that the method yields undesirable properties - the node degree distribution of an AGM graph shows exponential decay at the tail end, multimodality, and low probability mass at low values, and hence results based on AGM should be treated cautiously.

4. **Synthetic/ predicted communities comparison**:

   Finally we compare the synthetic communities generated in step 1 the with the predicted communities obtained in step 3, and compute similarity measures as appropriate. We also collect some communities statistics of the two networks, which details, together with similarity measures of the networks, are discussed below.

Metrics of Interest

Throughout every run with procedures outlined above, we collect the following metrics to determine the community detection method (hereafter *the method*)'s accuracy and infer its scalability:

- **Communities statistics**:

  We are interested in some general properties of the generated and predicted networks: number of individuals and communities in each network, as well as extremity and centrality of number of membership/ affiliation for each community/individual.

  Statistics for synthetic communities ensure our pre-defined parameters remain accurate throughout each run, while statistics for predicted communities allow us to uncover implicit assumptions of the method on communities structure, membership distribution, etc.

- **Graph statistics**:

  As edge generation is probabilistic, we cannot guarantee the number of edges generated by our graph generation process is exactly $|\hat{E}| = \frac{|\hat{E}|}{|V|} \times |V|$. We collect the number of nodes and edges in the graph obtained from our graph generation method, so we can better explore number of edges' effect on the method's accuracy and scalability.

- **Similarity measures**:

  A higher similarity between the networks indicates a higher accuracy for the method. The measures currently consists two versions of F1-score between the synthesised and predicted communities: an unweighted average and a weighted-by-community-size average of F1-score for all synthesised communities. We will formally define the similarity measures and their interpretation in section 5.1.3.

  While each F1-score is by itself indicative of accuracy of the method, by comparing two versions of F1-score we may gain an understanding on the strengths and weaknesses of the method on detecting communities with different size.

- **Run time**:

  Used to infer the scalability of the method, we are interested in the time taken for the method to initialise (i.e. perform a conductance test), run the gradient ascent iterations to obtain a good approximation of edge weights matrix, and determining community affiliations from the edge weight matrix respectively (see section 5.1.4 for an overview of

different stages in BigClam community detection algorithm) under different test parameters.

We take the time to carry out the operations above in both wall clock time and CPU time - the later offers a much higher resolution on time taken to run a procedure, yet CPU timers suffer from integer overflow within 32 minutes under 32-bit architectures, making it only useful for programs with short runtime.

5.1.3 Accuracy of BigClam in Synthetic Communities and Networks

We evaluate if BigClam is capable of producing accurate community predictions, and if changes in network characteristics (e.g. dispersion in community size, degree of overlap for communities) will affect BigClam's ability to give accurate predictions.

Similarity Measures

We use two variants of F1 scores throughout our evaluation of BigClam's accuracy. The F1 scores range between zero and one, and a higher F1 score indicates a better performance in prediction. F1 scores are chosen as they are featured in Yang and Leskovec's (2013) paper and likely to be heard of by readers. Other measures featured in Yang and Leskovec (2013) include Omega-index and Normalised Mutual Information.

To enable discussion in the section, we formally define the F1 scores used in our experiments.

Definition 5.1. (F1 score between a community and a community prediction)

Let $C_i \in C$ be a community and $\hat{C}_j$ be a community prediction. Treating a community as a set of individuals, the F1 score between C_i abd $\hat{C}_j$ is defined as:

$$F_1(C_i, \hat{C}_j) = \frac{2 \times TP}{2 \times TP + FN + FP} , \tag{5.1}$$

where $TP = |C_i \cap \hat{C}_j|$, $FN = |C_i \setminus \hat{C}_j|$, and $FP = |\hat{C}_j \setminus C_i|$.

Definition 5.2. (F1 score between a community and predicted set of communities)

Let $C_i \in C$ be a community and $\hat{C}$ be a predicted set of communities. The F1 score between C_i and $\hat{C}$ is defined as:

$$F_1(C_i, \hat{C}) = \max_{\hat{C}_j \in \hat{C}} F_1(C_i, \hat{C}_j) , \tag{5.2}$$

the F1 score for between C_i its best match community in $\hat{C}$ as defined in definition 5.1.

Definition 5.3. (Average F1 score)

Let C be a set of communities, and $\hat{C}$ be a predicted set of communities. The F1 score between C and $\hat{C}$ is defined as:

$$F_1(C, \hat{C}) = \frac{1}{|C|} \sum_{C_i \in C} F_1(C_i, \hat{C}) \ , \tag{5.3}$$

the average value of the F1 score between each community and its best match in the predicted set of communities.

Definition 5.4. (F1 score weighted by community size)

Let C be a set of communities, and $\hat{C}$ be a predicted set of communities. The F1 score between C and $\hat{C}$, weighted by community size, is defined as:

$$F_1^*(C, \hat{C}) = \frac{1}{\sum_{C_i \in C} |C_i|} \sum_{C_i \in C} |C_i| F_1(C_i, \hat{C}) \ . \tag{5.4}$$

Degree of community overlap and dispersion of community sizes

We first investigate the relationship between the degree of overlap in communities and dispersion of community sizes on BigClam's prediction accuracy. Figure 5.2 clearly shows a positive correlation between α (the shape parameter in the power-law distribution on community sizes) and the F1 score. As small α leads to a large dispersion in distribution (due to a heavier tail end in power-law distribution), the experiment data suggests BigClam performs better under networks with less dispersion in community size. We did not collect data for $\alpha > 5$ as we believe it is not practical to model communities with little or no dispersion in sizes.

Figure 5.2 also shows a negative correlation between r (average number of affiliations per individual) and the F1 score. It is worth mentioning that for nearly disjoint communities, BigClam can almost retrieve all community affiliations, with (unweighted) average F1 score greater than 95%. However, for networks with increasing community overlap, BigClam's prediction performance drops accordingly. We believe the missed predictions originate from the heavily overlapped regions on the graph - it is extremely difficult to identify all of one's affiliations, unless they are distinct enough, and attract connections that are sufficiently apart from each other.

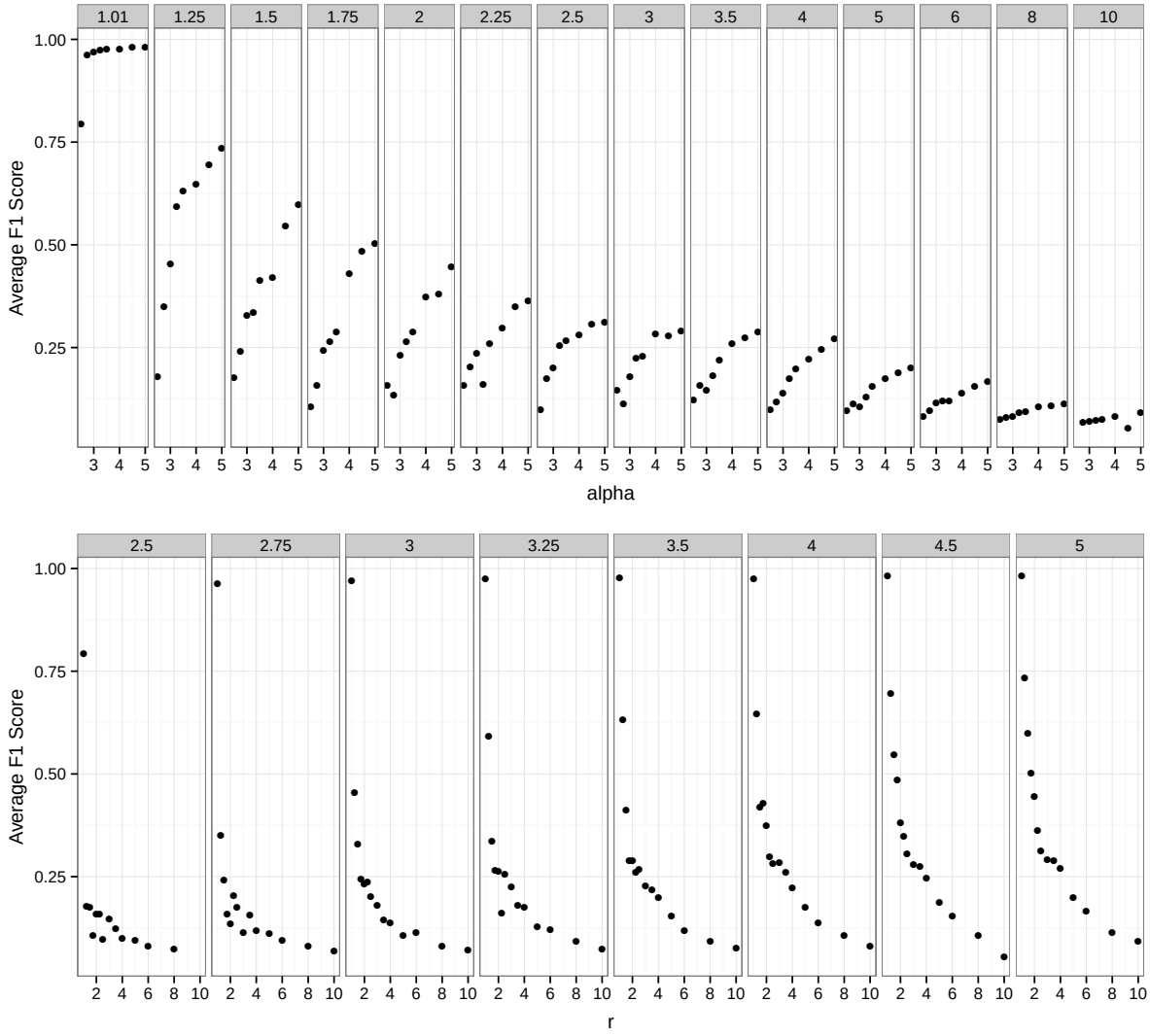


Figure 5.2: (Top) Plot of average F1 score achieved by BigClam against α under a fixed r indicated at the top of each sub-plot. (Bottom) Plot of average F1 score achieved by BigClam against r under a fixed α indicated at the top of each sub-plot. The plots are based on graphs generated by our graph generative process, with 500,000 individuals, 20,000 communities, and on average 20 edges per individual.

5.1.4 Scalability of BigClam in Synthetic and Real Networks

We evaluate if the BigClam community detection algorithm is scalable by identifying the three main stages of the algorithm and giving their runtime complexity in terms of the test parameters in section 5.1.2. This allows us to identify cases where the parameters lead to a prohibitively large runtime.

Three Stages of The BigClam Community Detection Algorithm

We observe the BigClam community detection algorithm has three prominent stages: the initialisation/conductance test (CT) stage, the gradient ascent (GA) stage, and the community association (CA) stage. Recall the operations carried out in the three stages, as stated by the definitions in section 3.2.2:

- The initialisation/conductance test stage:

  Definition 3.13. (Initialisation of affiliation weights matrix)

  Let F be an affiliation weights matrix. The algorithm initialises F as follows:

  $$F_{(u')(N(u))} = \begin{cases} 1 & \text{if } u' \in N(u) \text{ and } N(u) \text{ is a locally minimal neighbourhood of } u \\ 0 & \text{otherwise} \end{cases} , \tag{3.16}$$

  where $N(u)$ is that defined in definition 2.4.

- The gradient ascent stage, which terminates if the following condition is met:

  Definition 3.14. (Termination of the community detection algorithm)

  Let $l(F_u)$ be the log-likelihood of the affiliation weights vector for u, and $l'(F_u)$ be the log-likelihood of the affiliation weights vector for u after the update by gradient ascent (equation 3.14). The iterative gradient ascent process terminates if:

  $$\frac{l'(F_u) - l(F_u)}{l(F_u)} < 0.0001 \quad \forall u \in V , \tag{3.17}$$

  i.e. the log-likelihood increases by less than 0.01% $\forall u \in V$.

- The community association stage:

  Definition 3.15. (Determining existence of affiliation/ membership)

  Let $G = G(V, E)$ be a network, F be the most likely affiliation weights matrix under G, and $\epsilon = \frac{2|E|}{|V|(|V|-1)}$ be the background probability for a random edge to form in G.

  The algorithm regards $u \in V$ as a member of $c \in C$ if $F_{uc} \geq \delta = \sqrt{-\log(1-\epsilon)}$.

Runtime Complexity

Due to the variety of operations involved in each of the stages, it is difficult to obtain the runtime complexity via a program analysis approach. We instead perform a series of tests, each with different numbers of individuals ($|V|$), communities ($|C|$) and average node degrees ($\frac{|\hat{E}|}{|V|}$) (keeping other parameters fixed), and record the time taken to run each stage. The runtime complexity is then obtained by fitting three normal linear models of the form:

$$\log_{10}(T_s) = \beta_0 + \beta_1 \log_{10}(|V|) + \beta_2 \log_{10}(|C|) + \beta_3 \log_{10}\left(\frac{|\hat{E}|}{|V|}\right) , \tag{5.5}$$

where T_s is the time taken to run stage $s \in \{\text{CT}, \text{GA}, \text{CA}\}$, and β_i is the (unknown) linear model parameters. Notice equation 5.5 is equivalent to the following:

$$T_s = 10^{\beta_0}|V|^{\beta_1}|C|^{\beta_2}\left(\frac{|\hat{E}|}{|V|}\right)^{\beta_3} , \tag{5.6}$$

which is not linear on β_i, and hence forbids the formulation of a linear model.

Conductance Test Stage

We estimate the runtime complexity for the initialisation/conductance test stage to be $O\left(|V|\frac{|E|}{|V|}^2\right)$. The stage calculates the conductance for the neighbourhood of all nodes (an $O(|V|)$ operation). For each node u, the algorithm visits all nodes in the neighbourhood of u (an $O\left(\frac{|E|}{|V|}\right)$ operation), and in each neighbour visit the algorithm has to determine if each of the neighbour's neighbours is within the neighbourhood of u (another $O\left(\frac{|E|}{|V|}\right)$ operation) to calculate the conductance.

The conductance test stage appears to have runtime complexity $O\left(|V|^{1.20}\left(\frac{|E|}{|V|}\right)^{1.85}\right)$ according to our experiment data. While it is statistically significant for the exponents to differ (by the t-test motivated by theorem 2.25), we believe we are on the right track - the discrepancy is likely to be caused by the extra operations done per node, as well as compiler optimisations which prevent redundant operations.

We also considered including the number of communities $|C|$ in the model, but did not proceed under this direction due to lack of statistical evidence. The inclusion of $|C|$ in the linear model does not bring a statistically significant improvement to its fitness (by the F-test motivated by theorem 2.26), and hence by the "simple is beautiful" principle we do not include $|C|$ in the model.

Gradient Ascent Stage

It is difficult to determine the runtime complexity for the gradient ascent stage, as the runtime is highly dependent on whether the convergence condition is met. It could take 30 iterations to reach an optimum, or a few thousand iterations before being cut off. The difference in runtime is shown in figure 5.3, in which runtime data fitted under the best fit linear model still demonstrate a dispersion of 1.1 units in the vertical scale, equivalent to 12.6 times difference in runtime.

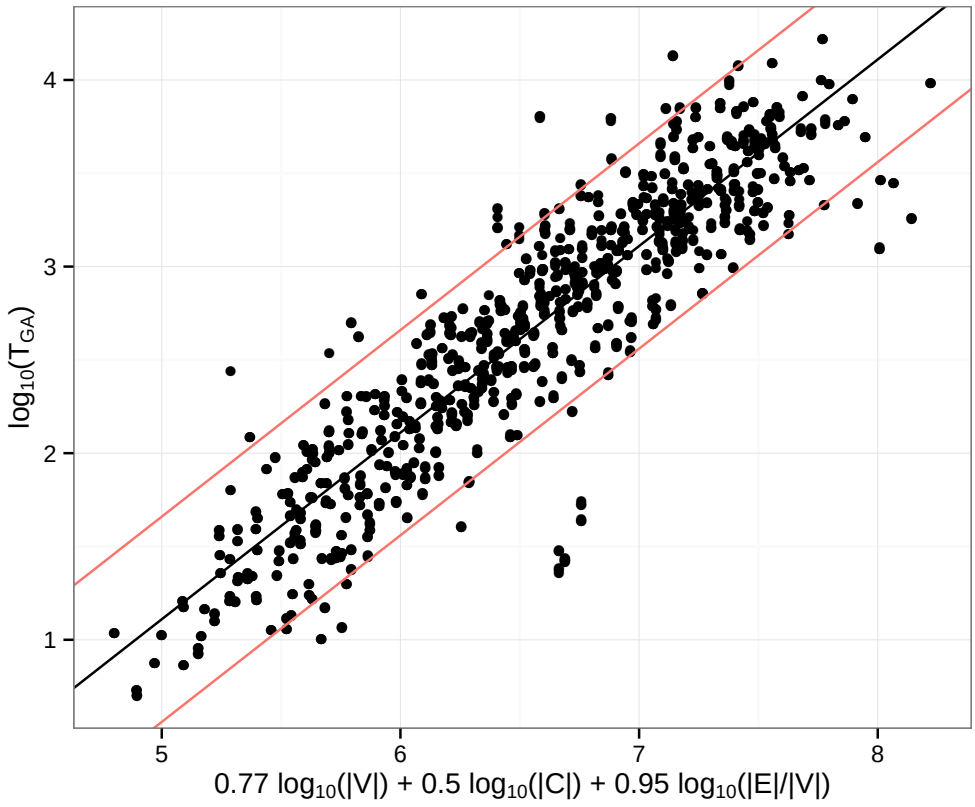


Figure 5.3: Runtime data for the gradient ascent stage fitted under the best fit linear model - $\log_{10}(T_{GA}) = -3.89 + 0.77\log_{10}(|V|) + 0.50\log_{10}(|C|) + 0.95\log_{10}(|E|/|V|)$. The diagonal line in black indicates the best line of fit, and the diagonal lines in red provides a guide on how dispersed the data points are around the best line of fit. The vertical distance between the two red lines is 1.1 units, equivalent to 12.6 times difference in runtime given similar parameters.

Our best fit in runtime data gives an estimated runtime complexity of $O\left(|V|^{0.77}|C|^{0.50}\left(\frac{|E|}{|V|}\right)^{0.95}\right)$. We believe the exponent for $\frac{|E|}{|V|}$ found in our experiments roughly matches what is claimed by Yang and Leskovec - by caching the column sum of the affiliation weights matrix, gradient ascent for a node will have a runtime complexity of $O(|\mathcal{N}(u)|) \approx O(\frac{|E|}{|V|})$. The approximation arises from the fact that the size of the neighbourhood of node u is simply the degree for node u plus one.

Further work is required to provide a plausible explanation for the estimated value of $|V|$ and

$|C|$'s exponents.

Community Association Stage

We estimate the runtime complexity for the community association stage to be $O(|V||C|)$. The estimate stems from the fact that the stage mainly involve scanning the affiliation weights matrix, with size $|V| \times |C|$.

Our data indicates the community association stage appears to have a runtime complexity $O(|V|^{1.27}|C|^{0.95})$. The data estimate indicates a number of operations, other than scanning the $|V|$ elements, are in place during a column scan. The average number of edges per node ($\frac{|E|}{|V|}$) is not included in the model, as it does not bring statistically significant (at 5% level) improvements in model fitness under a F-test motivated by theorem 2.26.

5.2 Speeding Up BigClam

We then discuss how to reduce the runtime for BigClam by parallelising the computation for the community association stage, a dominating stage in runtime (for networks with a large number of communities) which has yet to be parallelised.

5.2.1 Motivation

Networks with a fairly large number of communities are common, as real networks often demonstrate rich overlapping community structures. To illustrate, nearly all networks with ground-truth communities featured in Leskovec and Krevl (2014) have a large number of communities recorded [33], as shown in table 5.1.

	Number of individuals	Number of communities
LiveJournal online social network	4M	288k
Friendster online social network	66M	957k
Orkut online social network	3M	6M
Youtube online social network	1M	8k
DBLP collaboration network	317k	13k
Amazon product co-purchase network	335k	75k

Table 5.1: Number of communities recorded in the networks by Leskovec and Krevl (2014) [33]. The figures are rounded to the nearest million (denoted M) or thousand (denoted k). Note not every individual has a community affiliation.

We notice the community association stage dominates the runtime for the BigClam community detection algorithm for networks with a large number of communities. Figure 5.4 shows a plot

on the proportion of time spent in different stages of the algorithm on a number of networks with ground-truth communities featured in Leskovec and Krevl (2014) [33]. While the time spent on the conductance test stage is negligible, the time spent on the community association stage generally accounts for more than half of the entire algorithm's runtime.

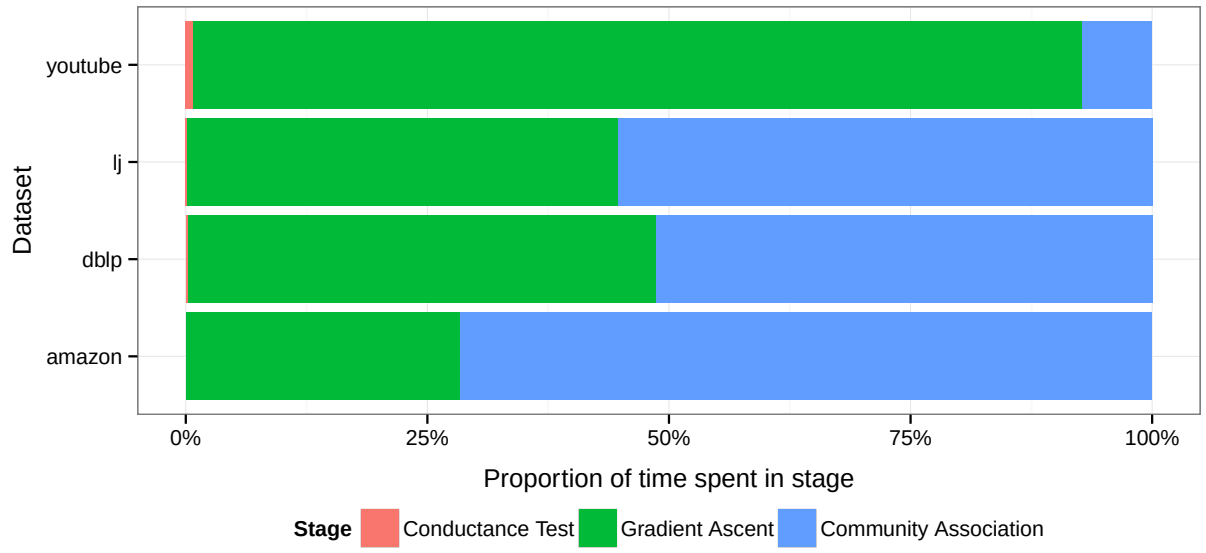


Figure 5.4: Average proportion of time spent on the three stages of the BigClam community detection algorithm (from left to right: Conductance Test, Gradient Ascent, and Community Association) for four networks with ground-truth communities (from top to bottom: Youtube social network, LiveJournal online social network, DBLP collaboration network, Amazon product co-purchase network) in Leskovec and Sosič's (2014) implementation. The implementation is tested on an eight-thread machine, with the number of communities to detect for each dataset set to the number of communities recorded by Leskovec and Krevl (2014) (see table 5.1).

However, unlike the gradient ascent stage, in which computation is parallelised over multiple CPU threads, the community association stage is not parallelised. This means the majority of CPU resources and man-time is wasted on idle "mode". Parallelised computation in the community association stage will better utilise available resources, and hence improve the scalability of the BigClam community detection algorithm.

5.2.2 Methodology

We parallelise the community association stage with OpenMP, a specification for high-level parallelism in C++ programs. More information on OpenMP is available in appendix B.

Overview of Implementation

To enable parallelisation for the computation of the community association stage, we outline the program flow of the C++ method corresponding to the community association stage. The method takes a reference to the collection of community memberships, the lower affiliation

weight threshold for an individual to be considered a member of a community and the minimum size of a community, and overwrites the collection with community memberships/affiliations obtained from the gradient ascent stage. The collection of community memberships is implemented as a vector of vectors - the nested vector with index i records the ID of community i's members.

Recall from section 3.2.2, given an affiliation weights matrix F, we consider an individual u to be a member of community c if F_{uc} is greater than the lower affiliation weight threshold. This is achieved in the implementation by scanning the affiliation weights matrix generated in the gradient ascent stage top-down, left-right. For each scan along the same column (i.e. community), a vector (local in scope) is used to keep track of members already associated to the community, and added to the collection of community memberships when the scan along the column is completed.

The setting above allows us to spread the tasks of scanning a particular column between multiple threads. Figure 5.5 shows an illustration on how parallelised operation can reduce time taken to scan all matrix columns.

Parallelisation with OpenMP

To prevent race conditions yet maintain the highest level of parallelism possible, we declare all operations that involve any objects shared between threads that may be overwritten critical. It is safe to parallelise operations that involve only objects used by that particular thread (i.e. private objects/variables) and read-only objects that are shared between threads. In our case, the only object which is shared between threads, and may involve write operations is the collection of community memberships. All other objects are either shared and read-only, or private to a thread:

- The affiliation weights matrix is read-only by all threads
- The lower affiliation weight threshold and minimum community size are constants, and hence are read-only
- The vector used to keep track of current community memberships is local in scope, and hence is private to a thread by OpenMP construction.

We hence declare the only operation on the collection of community memberships (i.e. the operation to add the vector used to keep track of community memberships processed by the current thread to the collection of community memberships) as critical, and parallelise all other operations whilst scanning a column across multiple threads.

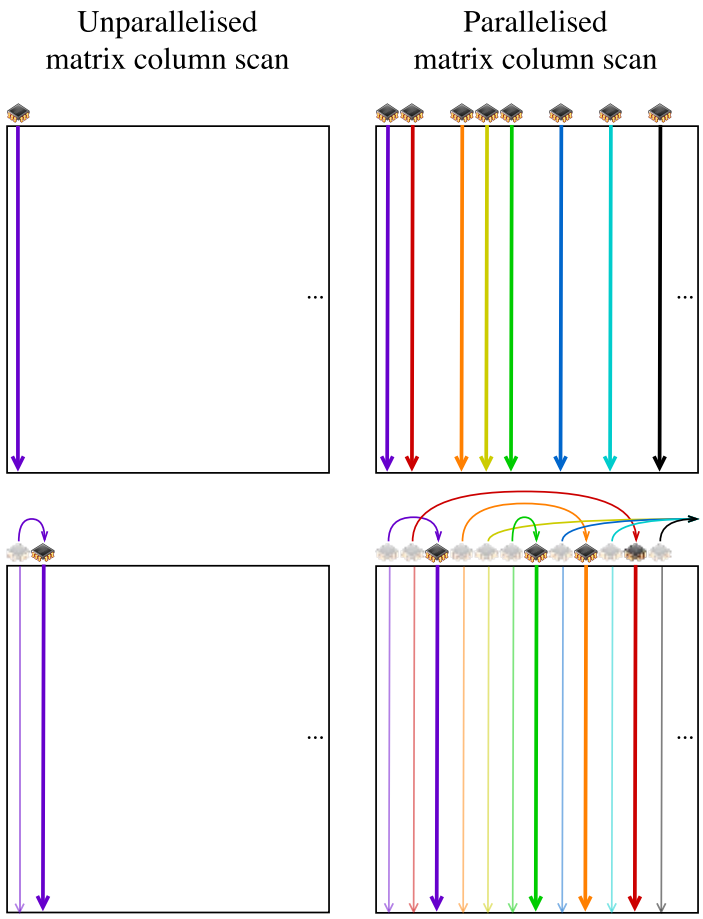


Figure 5.5: Scanning matrix columns under unparallelised (left) and parallelised (right) operations. Each colour represents operations completed (thin-lined arrow) or in progress (thick-lined arrow) by a particular thread. Assume it takes roughly the same time to scan any given column, parallelised operations reduce the time taken to scan the entire matrix.

5.2.3 Result and Evaluation

We proceed to show parallelising the community association stage actually reduces the runtime for both the community association stage and the entire BigClam community detection algorithm, and perhaps more importantly, it produces the same community memberships/affiliations prediction.

Runtime Reduction

We verify our parallelisation achieves runtime reduction with the procedures as follow. We run both the unparallelised and parallelised versions of BigClam implementation on multiple machines with the same specification (Intel i7-4790 CPU @ 3.60 GHz with eight threads). At any given instant the machines will run either the unparellelised or the parallelised version to ensure all CPU threads are utilised for one task. We automate the process by using Condor, which

allows us to submit, and schedule batch jobs to be run on machines within our department's network (see appendix A for more details on Condor).

For each run, the program is instructed to find the communities in one of the following four networks: the Amazon product co-purchase network, the DBLP collaboration network, the LiveJournal online social network, and the Youtube online social network, and is told the number of communities to detect is that listed in table 5.1. Similar to what is done in section 5.1.2, we measure the runtime for each stage of the BigClam community detection algorithm in both the parallelised and unparallelised versions.

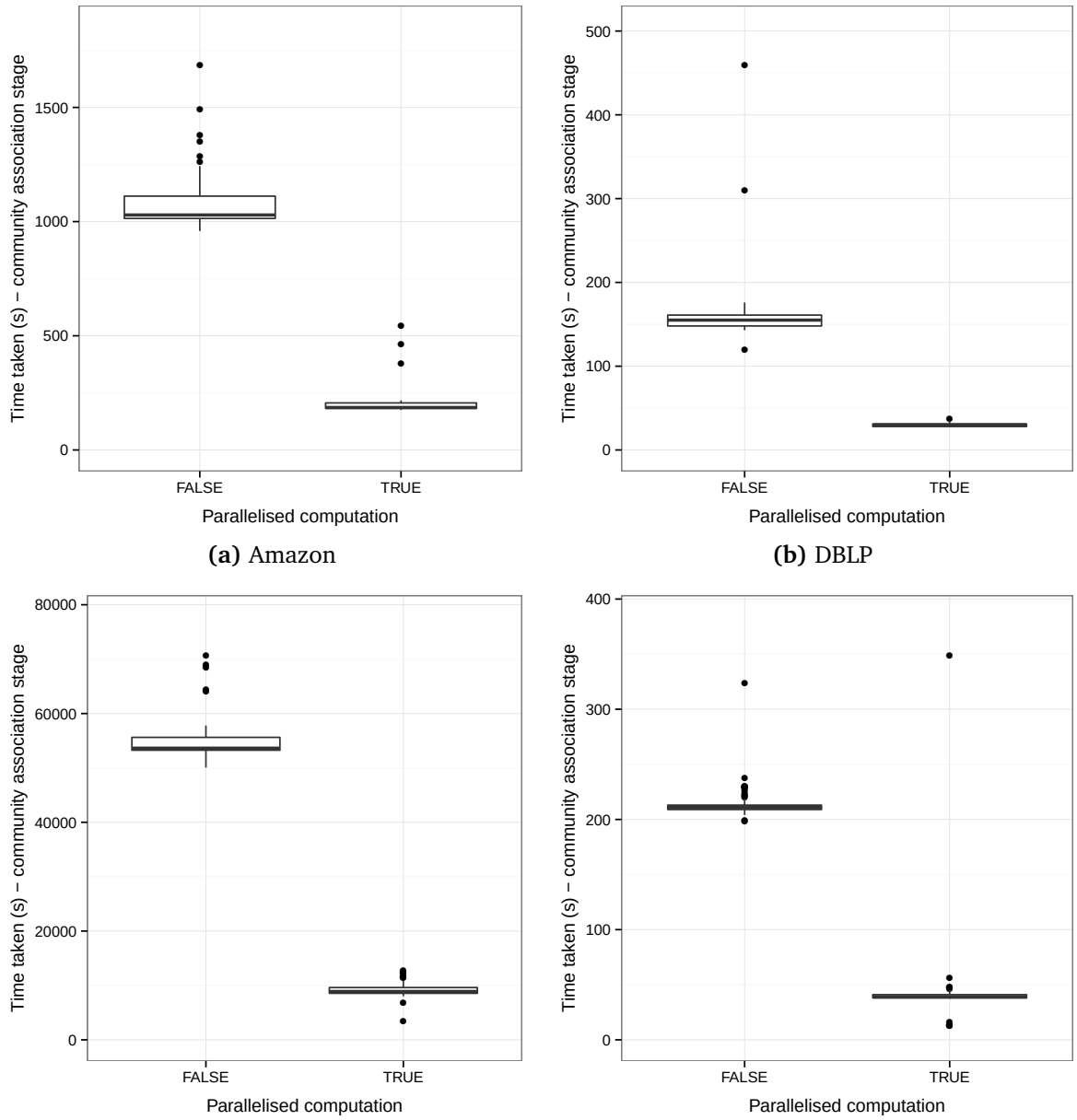
The result on runtime is encouraging - runtime data, the average values of which are listed in table 5.2 and quartiles plotted in figure 5.6, show a runtime reduction in the parallelised community association stage. In fact, we can show the difference in runtime is statistically significant at 5% level by performing a Welch's t-test.

		Average runtime	
		Unparallelised	Parallelised
Amazon product co-purchase network	CA stage	1077.58	203.74
	Overall	1505.38	610.23
DBLP collaboration network	CA stage	160.39	30.00
	Overall	312.47	180.69
LiveJournal online social network	CA stage	55363.22	9233.02
	Overall	100146.81	55259.48
Youtube online social network	CA stage	213.62	43.07
	Overall	2965.12	2717.85

Table 5.2: Average time taken, in seconds, to run **a)** the community association (CA) stage **b)** the entire BigClam community detection algorithm, without and with parallelisation of the community association stage. The implementations are tested on eight-thread machines with the same CPU specifications (Intel i7-4790 CPU @ 3.60 GHz).

We also put the numbers in context. With an eight-thread machine, we achieve a 430% times speed up in the community association stage for the Amazon product co-purchase network, and subsequently achieve a 150% times speed up in the overall BigClam community detection algorithm. Moreover, we manage to, on average, shave off 46,130 seconds (or 12.8 hours) from the runtime of the community association stage, and hence the BigClam community detection algorithm on the LiveJournal online social network. This leads to a significant runtime reduction: Yang and Leskovec (2013) have mentioned in their paper, "[w]ith 20 threads, it takes about one day to fit BigClam to the LiveJournal network" [54] - we manage to do the same with roughly half of their computation power in less than 16 hours.

On the other hand, parallelising the community association stage does not bring significant improvements in runtime on networks with lower numbers of communities. While being statistically significant, we only speed up the overall execution time on the Youtube online social network by 10% despite achieving 395% times speed up on the community association stage.



(a) Amazon **(b)** DBLP

(c) LiveJournal **(d)** Youtube

Figure 5.6: Box plot of the time taken, in seconds, to run the community association stage on different networks with ground-truth communities, without and with parallelisation on the stage.

The speedup is not apparent under such cases as the runtime for the BigClam community detection algorithm on networks with a small number of communities is dominated by the gradient ascent stage, and parallelising the computation in the community association stage would not bring significant improvement.

Program Correctness

We also show the parallelisation produces the same set of predictions in community membership/affiliation. Treating a community as a set of individuals, we consider the set of communities predicted by the unparallelised version to be the same as that of the parallelised version if they are the same set of sets of individuals.

As OpenMP does not necessarily assign tasks to threads in the original order, the output may be shuffled in both versions. The community with index i in the output of the parallelised version may not be equal to the community with the same index, but a community with a different index in the output of the unparallelised version. Likewise, the order which community membership is "confirmed" on an individual may change from a program run to another. This poses a problem as common approaches in comparing program output (e.g. comparing the community membership dump using `diff`, using MD5 check sum) will indicate failure, even if two sets of communities are the same. Thus we ensure the two sets of communities are the same by sorting the individual IDs in a community and the communities in lexicographical order in the program output before finding discrepancies within.

Our utility does not report any discrepancies between the post-processed program outputs, hence we conclude our parallelisation in the community association stage produces the same output in a shorter period.

5.3 Discussion

We conclude the chapter by discussing our contribution to the understanding of BigClam, the impact of speeding up BigClam's implementation, and future work our investigation has opened up.

5.3.1 Contribution and Impact

We devised a new verification framework on the accuracy and scalability of BigClam and any other overlapping community detection algorithms. The framework considers more aspects in a network with overlapping communities (e.g. the degree of overlap between communities, dispersion between community sizes, number of communities, number of nodes and edges), and hence offers a more comprehensive review of overlapping community detection algorithms. We would be unable to understand the relationship between community overlap and prediction accuracy of BigClam, or the runtime complexity of all three stages of the BigClam community detection algorithm without this verification framework.

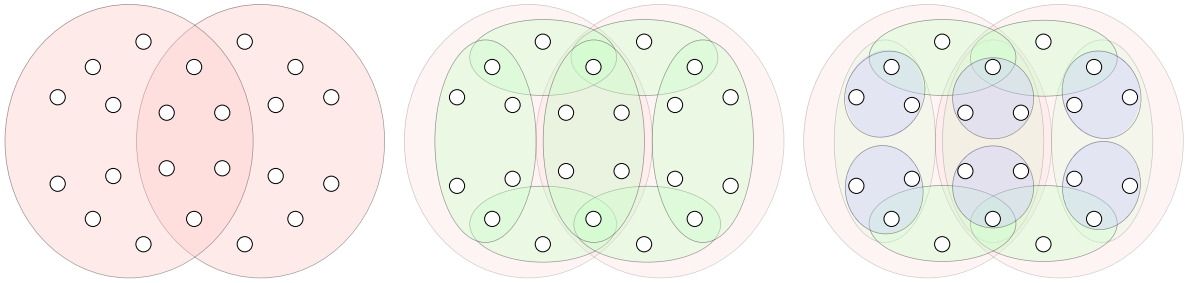
We sped up BigClam, a popular overlapping community detection algorithm, and submitted our code user for the parallelised operations to the Stanford Network Analysis Platform (SNAP)

for review. Parallelising the community association stage easily cut the runtime for BigClam to produce predictions in community memberships/affiliations in half (or more). It allows researchers and analysts to obtain community memberships/affiliations within a shorter time frame. As SNAP is an open source network analysis platform used by a number of researchers and analysts, our code contribution will benefit others for reasons discussed above.

5.3.2 Limitation and Future Work

The verification framework is "limited" in the sense that it is unable to reproduce all experiments conducted by Yang and Leskovec (2013). Further work is required to extend the framework with more prediction accuracy indicators (e.g. Omega-Index, Normalised Mutual Information) to give an overlapping community algorithm an all-rounded evaluation it deserves.

Throughout the chapter, we assumed the number of communities are known, which is not necessarily the case in real-life applications. Yang and Leskovec (2013) have proposed a mechanism to determine the number of communities by performing some sort of cross-validation and locating the most likely number of communities (see the paragraph titled "Determining number of communities" approaching the end of section 3.2.2). Unfortunately we did not have the chance to evaluate if the mechanism is of an acceptable standard - if it can recover the number of communities we have specified while running the community generative process. The ability to verify such mechanisms can allow us to better understand the resolution of an overlapping community detection algorithm (i.e. how deep down in a hierarchical community structure can the community detection algorithm uncover, see figure 5.7 for an illustration).



(a) Top level communities **(b)** Sub-communities **(c)** Sub-sub-communities

Figure 5.7: Three 22-node networks with different "resolution" in overlapping community structure. The network on the left features two communities (in red ellipses), the network on the middle a number of sub-communities within the communities (in green ellipses), and the network on the right features a number of sub-sub-communities within the sub-communities (in blue ellipses). Connections between individual nodes are omitted for clarity.

Chapter 6

Conclusion

We conclude our work by summarising our contributions and outlining a number of open problems in the general field of community detection.

6.1 Contribution

Our work's contribution lies in two domains: graph generation and community detection.

We developed an overlapping community-based graph generative model. The model is one of the first developed which features both an overlapping community structure and the scale-free property. The model is based on our novel observation that individuals who are more influential, engaged and participate more gain more connections within a community. We have also discovered during our first attempts that the Community-Affiliation Graph Model (AGM) proposed by Yang and Leskovec (2012) [53] does not generate scale-free networks, and formalised the distributional result.

Furthermore, we extended the verification framework for the Cluster Affiliation Model for Big Networks (BigClam) proposed by Yang and Leskovec (2013) [54]. The framework accommodates extra network characteristics, and allows us to establish the correlation between the degree of community overlap, the dispersion of community sizes, and BigClam's prediction accuracy. We identified the three stages of BigClam and obtained their apparent runtime complexity.

Our final contribution involves speeding up BigClam. We observed the community association stage for BigClam dominates the runtime under networks with a large number of communities, yet its computation is not parallelised. We sped up the runtime by as much as 430% by parallelising the stage's computation.

6.2 Future Work

On top of the future work proposed in chapters 4 and 5, including:

- The use of non-parametric models and inference methods for graph generation (see section 4.8.1),
- Generation of weighted/directed/temporal graphs (see section 4.8.2), and
- Evaluation of BigClam's mechanism in finding the optimal number of communities (see section 5.3.2),

we believe the following problems in the general domain of community detection in networks are worth looking into further. For each problem, we give a brief motivation, provide the statement of the problem, and outline any existing work done and/or possible approaches to the problem.

6.2.1 Pivoting and Membership Constraints

Usually, we possess knowledge of some individuals within a network. These individuals may provide their affiliations (e.g. hometown, current region of residence, current place of study/work). They may also have their affiliations tagged by others or inferred, in the case of famous stars. We regard such individuals as the gateway to their communities.

We are interested in the answer to the following problem: Given the knowledge of the individuals acting as the gateway, is it possible to "pivot" on them to increase the accuracy and interpretability of a community detection algorithm (say BigClam)?

Current work has proposed seed set expansion approaches to detect overlapping communities, in which knowledge of a small number of community members (e.g. gateway individuals) and their connections is utilised to discover additional members of the community quickly. These approaches are better suited for local community detection problems (given some individuals of interest, identify communities concentrated around the individuals locally (Kloumann and Kleinberg, 2014) [29]). However, we can combine the approaches with BigClam-like algorithms by seeding the initial communities with the gateway individuals and seeing how the communities detected evolve around the individuals.

We can further strengthen our assumptions. Instead of pivoting on certain individuals for a community and end up not having the said individuals in the final community predictions, we seek the possibility of imposing membership constraints. Under membership constraints certain individuals must be affiliated with some pre-determined, labelled communities. Success in doing so could make overlapping community models more interpretable, for example explaining the existence of fan clubs for stars or supporters in different football teams, simply with some extra information on the star or leader node.

6.2.2 Scaling a global community detection algorithm to a massive network

We are interested in developing and/or implementing an overlapping community detection algorithm which could resolve communities from massive networks quickly and accurately. One example would be an overall user graph of Twitter, with at least 320 million monthly active users as nodes (Twitter Inc., 2016) [46].

No experiments on overlapping community detection at the mentioned scale have been done to date. The largest experiments on overlapping community detection involved a network with 4.8 million nodes and 42.8 million edges (Gleich and Seshadhri, 2012) [21]. Blondel et al. (2008) experimented with a 118 million-node, 1 billion-edge network [7], though their study centered around a non-overlapping community detection algorithm.

Given sufficient computing resources, techniques in identifying and resolving potential bottlenecks (see sections 5.1.4 and 5.2), it should be possible to scale a good community detection model and algorithm to the entire Twitter network. Solving for communities in the Twitter network, and possibly other massive networks, could set a new benchmark for community detection in terms of speed and scale.

Bibliography

[1] Adamic, L.A. (2000) *Zipf, power-laws, and pareto-a ranking tutorial*. Availale from: `http://www.hpl.hp.com/research/idl/papers/ranking/ranking.html` [Accessed 8th April 2016]. pages

[2] Albert, R., Barabási, A.L. (2002) Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1), 47-97. pages 24

[3] Al-Zahrani, B., Sagor, H. (2014) The Poisson-Lomax distribution. *Colombian Journal of Statistics*, 37(1), 223-243. pages 52

[4] Arnold, B.C. (2014) Univariate and multivariate Pareto models. *Journal of Statistical Distributions and Applications* 1(1), 1-16. Available from: doi:10.1186/2195-5832-1-11 [Accessed 12th April 2016]. pages 10

[5] Barabási, A.L. and Albert, R. (1999) Emergence of scaling in random networks. *science*, 286(5439), 509-512. pages i, 4

[6] Barney, B. (2015) *OpenMP*. Available from: `https://computing.llnl.gov/tutorials/openMP/` [Accessed 7th April 2016]. pages 97

[7] Blondel, V.D., Guillaume, J.-L., Lambiotte, R., Lefebvre, E. (2008) Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*. 2008 (10), P10008. Available from: doi:10.1088/1742-5468/2008/10/P10008 [Accessed 28th January 2016] pages 35, 62, 87

[8] Broderick, T., Jordan, M.I., Pitman, J. (2012) Beta processes, stick-breaking, and power laws. *Bayesian Analysis*. 7 (2), 439-476. Available from: doi:10.1214/12-BA715 [Accessed 8th April 2016] pages 38

[9] Broderick, T., Pitman, J., Jordan, M.I. (2013) Feature allocations, probability functions, and paintboxed. *Bayesian Analysis*. 8 (4), 801-836. Available from: doi:10.1214/13-BA823 [Accessed 29th January 2016] pages 3

[10] Caron, F., Fox, E.B. (2014) Sparse graphs using exchangeable random measures. *arXiv preprint*. Available from: `http://arxiv.org/abs/1401.1137` [Accessed 6th June 2016] pages 62

[11] Coles, S. (2001) *An Introduction to Statistical Modeling of Extreme Values*. Springer Series in Statistics. London, Springer. pages

[12] Chong, E.K.P., Żak, S.H. (2013) *An Introduction to Optimization*. Wiley Series in Discrete Mathematics and Optimization. 4th Edition. New York, John Wiley & Sons, Inc. pages 18, 19

[13] Cohen, R.F., Eades, P., Lin, T., Ruskey, F. (1997) Three-dimensional graph drawing. *Algorithmica*. 17(2), 199-208. Available from: doi:10.1007/BF02522826 [Accessed 28th January 2016] pages 2

[14] Computing Support Group, Imperial College London. (2016) *CSG: Services - Hpc - Condor*. Available from: `http://www.doc.ic.ac.uk/csg/services/hpc/condor` [Accessed 5th April 2016] pages 95

[15] Edalat, A. (2015) *Random Networks*. [Lecture] Dynamical Systems and Deep Learning. Imperial College London, 13th November. pages 22

[16] Erdős P., Rényi, A. (1959) On random graphs I. *Publicationes Mathematicae Debrecen*, 6, 290-297. pages i, 4, 21

[17] Fortunato, S. (2010) Community detection in graphs. *Physics Reports*. 486, 75-174. Available from: doi:10.1016/j.physrep.2009.11.002 [Accessed 27th January 2016] pages 1, 3, 34

[18] Freeman, L.C. (2004) *The Development of Social Network Analysis: A Study in the Sociology of Science*. Vancouver, B.C., Empirical Press. pages 1

[19] Gilbert, E.N. (1959) Random Graphs. *Annals of Mathematical Statistics*, 30 (4), 1141–1144. pages 21

[20] Girvan, M., Newman, M.E.J. (2002) Community structure in social and biological networks. *Proceedings of the Natural Academy of Sciences of the United States of America*. 99 (12), 7821-7826. Available from: doi:10.1073/pnas.122653799 [Accessed 29th January 2016] pages 1

[21] Gleich, D.F., Seshadhri, C. (2012) Vertex neighborhoods, low conductance cuts, and good seeds for local community methods. In: ACM, *KDD '12 Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. New York, ACM. pp. 597-605. pages 87

[22] Gleiser, P., Danon, L. (2003) Community Structure in Jazz *Advances in Complex Systems*, 6 (4), 565-573. Available from: doi:10.1142/S0219525903001067 [Accessed 27th January 2016] pages 2

[23] Griffiths, T.L., Ghahramani, Z. (2011) The Indian Buffet Process: An Introduction and Review. *Journal of Machine Learning Research*, 12 (Apr), 1185-1224. pages 62

[24] Holgate, P. (1970) The modality of some compound Poisson distributions. *Biometrika*, 57 (3), 666-667. Available from: doi: 10.1093/biomet/57.3.666 [Accessed 31st May 2016] pages 52

[25] Holland, P.W., Laskey, K.B., Leinhardt, S. (1983) Stochastic Blockmodels: First Steps. *Social networks*, 5(2), 109-137. pages i, 4, 24

[26] Hoyer, P.O. (2004) Non-negative matrix factorization with sparseness constraints. *The Journal of Machine Learning Research*, 5, 1457-1469. pages 34

[27] Karlis, D., Xekalaki, E. (2005) Mixed poisson distributions. *International Statistical Review*, 73(1), 35-58. pages 14

[28] Klemm, K., Eguíluz, V.M. (2002) Growing scale-free networks with small-world behavior. *Physical Review E*, 65 (5), 057102. pages 24

[29] Kloumann, I. M., Kleinberg, J.M. (2014) Community membership identification from small seed sets. In: ACM, *KDD '14 Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. New York, ACM. pp. 1366-1375. pages

[30] Lau, D.-H. (2016) *Normal Linear Models*. [Lecture] Statistical Modelling II. Imperial College London, 18th January. pages 15, 16

[31] Leskovec, J. (2015) *Community Detection in Graphs*. [Lecture] Mining Massive Datasets. Coursera (Stanford University), 26th September. pages

[32] Leskovec, J. (2015) *Solving the BigClam*. [Lecture] Mining Massive Datasets. Coursera (Stanford University), 26th September. pages

[33] Leskovec, J., Krevl, A. (2014) *SNAP Datasets: Stanford Large Network Dataset Collection* [Data file] Stanford University, CA, United States of America. Available from: `http://snap.stanford.edu/data` [Accessed 7th December 2015] pages 38, 65, 76, 77

[34] Leskovec, J., Sosič, R. (2014) *SNAP: A general purpose network analysis and graph mining library in C++* (Version 2.4) [Computer code] Stanford University, CA, United States of America. Available from: `http://snap.stanford.edu/snap` pages 34, 50, 65, 68, 97

[35] Magee, J., Kramer, J. (2006) *Concurrency: State Models and Java Programs*. 2nd Edition. New York, John Wiley & Sons, Inc. pages 99

[36] Misener, R. (2016) *One-dimensional Optimisation*. [Lecture] Computing for Optimal Decisions. Imperial College London, 2nd February. pages 19

[37] Newman, M.E.J., Strogatz, S.H., Watts, D.J. (2001) Random graphs with arbitrary degree distributions and their applications. *Physical Review E*, 64 (2), 026118. pages 22

[38] Newman, M.E.J. (2005) Power laws, Pareto distributions and Zipf's law. *Contemporary physics*, 46 (5), 323-351. pages 38

[39] Ondrejcsák, R. (2009) American Foreign and Security Policy under Barack Obama: change and continuity, In: Majer, M. – Ondrejcsák, R. – Tarasovič, V. – Valášek, T. (eds.) *Panorama of global security environment 2009*. Bratislava, Slovakia, CENAA, pp. 147-162. pages

[40] OpenMP ARB Corporation. (2013) *FAQ OPENMP*. Available from: `http://openmp.org/openmp-faq.html` [Accessed 5th April 2016] pages 97

[41] Oxford University Press. (2016) *community - definition of community in English from the Oxford dicationary*. Available from: `http://www.oxforddictionaries.com/definition/english/community`. [Accessed 29th January 2016] pages 1

[42] Parpas, P. (2016) *The Newton-Raphson and Related Methods*. [Lecture] Computing for Optimal Decisions. Imperial College London, 16th February. pages 19

[43] Price, D.J. de S. (1965) Networks of scientific papers. *Science*, 149 (3683), 510-515. pages 38

[44] Schucany, W. R. (1971) Order statistics in simulation. *Journal of Statistical Computation and Simulation*, 1 (3), 281-286. pages 53

[45] Todeschini, A., Caron, F. (2016) Exchangeable Random Measures for Sparse and Modular Graphs with Overlapping Communities. *arXiv preprint*. Available from: `https://arxiv.org/abs/1602.02114`. [Accessed 19th May 2016]. pages 57, 62

[46] Twitter, Inc. (2016) *Company | About*. Available from: `https://about.twitter.com/company`. [Accessed 28th January 2016]. pages 87

[47] Wang, Y.H. (1993) On the number of successes in independent trials. *Statistica Sinica*, 3 (2), 295-312. pages 13, 56

[48] Watts, D.J., Strogatz, S.H. (1998) Collective dynamics of 'small-world' networks. *Nature*, 393, 440-442. pages i, 4, 22, 23

[49] Welch, B.L. (1947) The generalization of "Student's" problem when several different population variances are involved. *Biometrika*, 34 (1–2), 28–35. pages 14, 15

[50] Whang, J.J., Gleich, D.F., Dhillon, I.S. (2013) Overlapping community detection using seed set expansion. In: ACM, *CIKM '13 Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. New York, ACM. pp. 2099-2108. pages 2

[51] Willmot, G.E. (1990) Asymptotic Tail Behaviour of Poisson Mixtures with Applications. *Advances in Applied Probability*, 22 (1), 147-159. pages 52

[52] Willmot, G.E. (1993). On Recursive Evaluation of Mixed Poisson Probabilities and Related Quantities. *Scandinavian Actuarial Journal*, 1993(2), 114-133. pages 52

[53] Yang, J., Leskovec, J. (2012) Community-affiliation graph model for overlapping network community detection. In: IEEE, *Data Mining (ICDM), 2012 IEEE 12th International Con-*

ference. New Jersey, IEEE Conference Publishing Services. pp. 1170-1175 pages i, 2, 4, 5, 26, 62, 85

[54] Yang, J., Leskovec, J. (2013) Overlapping Community Detection at Scale: A Nonnegative Matrix Factorization Approach. In: ACM *WSDM '13 Proceedings of the sixth ACM international conference on Web search and data mining, February 4–8, 2013, Rome, Italy*. New York, ACM. pp. 587-596. pages i, 3, 6, 28, 29, 30, 41, 54, 62, 65, 66, 80, 85, 97

[55] Yang, J., Leskovec, J. (2015) Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems*, 42(1), 181-213. pages

Appendix A

Using Condor for Running Experiments In Parallel

> *"Condor is an open-source, high-throughput batch-processing system developed by the University of Wisconsin[-Madison] . [... It] is useful for running large numbers of independent, CPU-intensive jobs in an automated, reliable fashion." (Computing Support Group, Imperial College London, 2016) [14]*

We observed experiments outlined in sections 5.1 and 5.2 involve executing the same program thousands of times with different parameters. Each executable run is independent of each other, does not require inter-processes communication, is CPU-intensive (usually taking all CPU cores available in a machine), and time-consuming.

As it is time-infeasible to run the executables sequentially on one machine, we take advantage of the existing Condor system to manage and run our experiments on around 360 machines available in Department of Computing, Imperial College London, all running with 8-thread Intel CPUs.

The general workflow with Condor used in our experiments are as follows (also illustrated in figure A.1):

Create Job Specification Condor requires a job specification for each executable run. The job specification contains details on which execution mode to be adopted, path to executable, program arguments, and paths to output and log files (a guide is available from the Computing Support Group, Imperial College London [14]). For this project, the job specification files are generated dynamically with bash scripts `code/condor_*.sh`, and they should remain until their associated executable run completes.

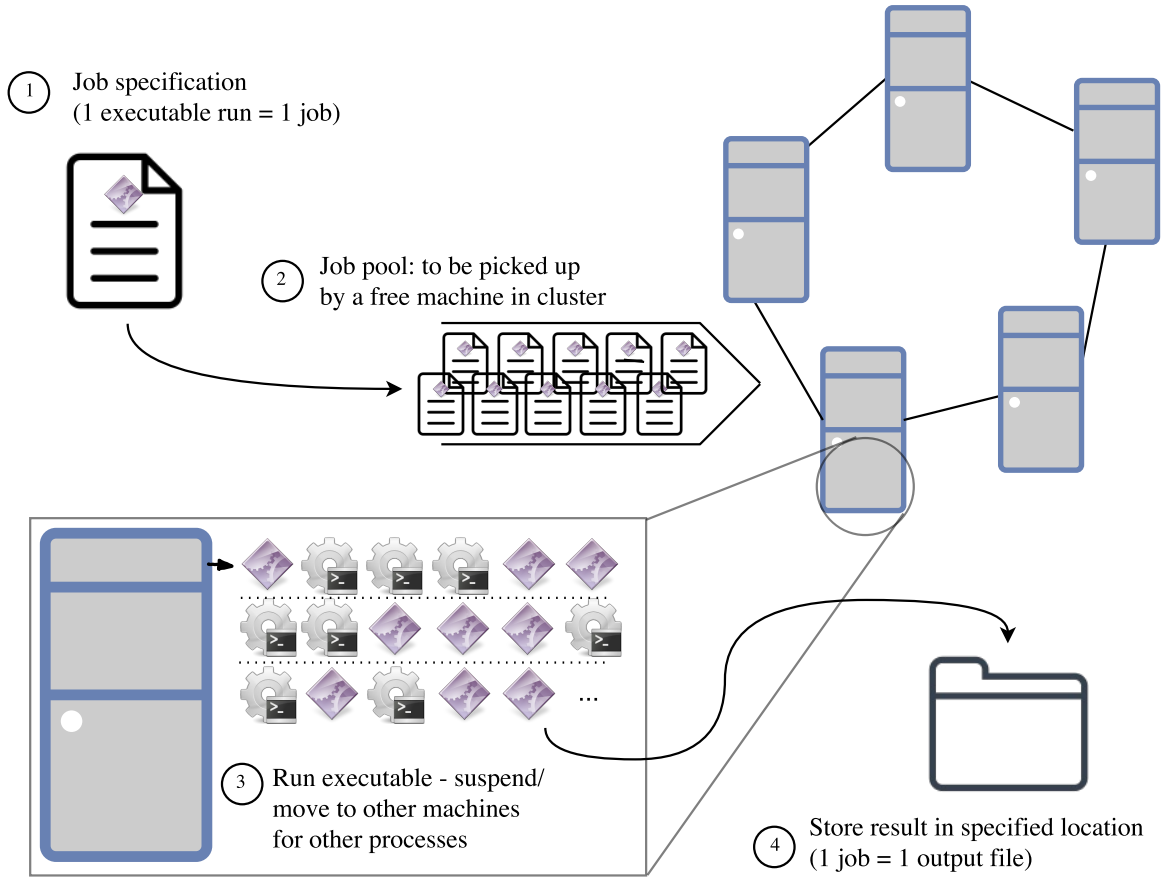


Figure A.1: Illustration of Condor workflow used in experiments.

Submit Job To Condor Job Pool We then submit the completed job specifications to the job pool of Condor (for this project, the procedure is done automatically by the bash scripts mentioned above). Jobs submitted are picked up by an idle machine managed by Condor without the need of user intervention.

Executables Being Run in Machines The machine then automatically runs the executable with specified parameters referenced by the job specification it picked up. Condor at the Department of Computing, Imperial College London is configured under "cycle-stealing" mode: regular snapshots are taken of the machines' state, and executables on one machine may be suspended or moved to another machine if processes with a high priority (e.g. interactive sessions for users physically in front of a machine) require CPU usage. The use of snapshots allows the execution to be accurately timed, as time spent by executable in suspended state is ignored.

Results Stored in Specified Location The programs in the project are written such that executables will write resultant data in specified directories. To prevent race conditions, each executable run will write data to a separate file. We then aggregate data in resultant files using utilities such as `cat`.

Appendix B

Using OpenMP for Parallelised Computation

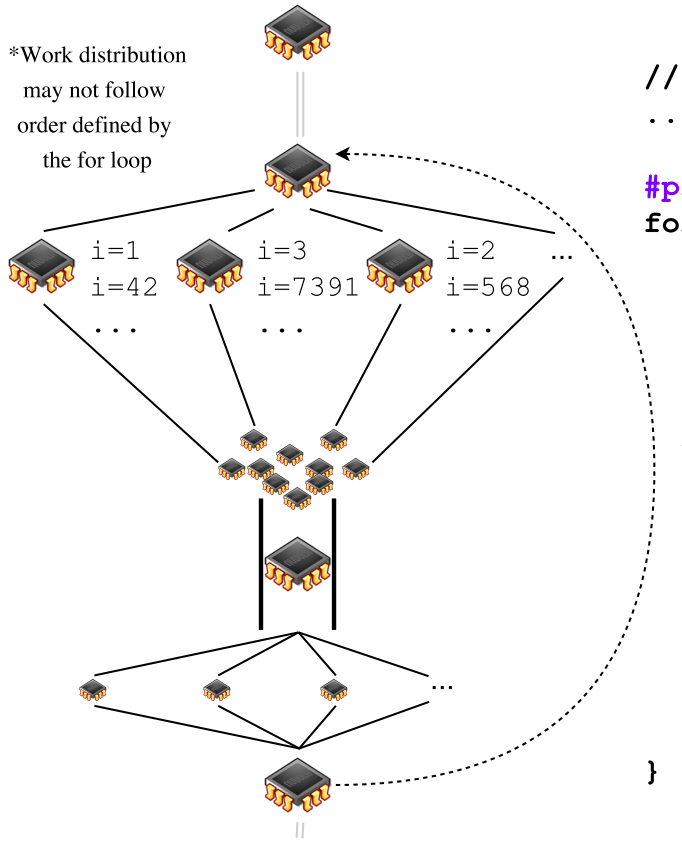
> *"OpenMP is a specification for a set of compiler directives, library routines, and environment variables that can be used to specify high-level parallelism in [..] C++ programs." (OpenMP ARB Corporation, 2013) [40]*

Unlike Condor, which distributes independent program executions on multiple machines, OpenMP allows one to parallelise their program onto multiple CPU cores/threads to achieve execution speedup on a single machine. It is used throughout BigClam's code implementation in the Stanford Network Analysis Project (Yang & Leskovec (2013), Leskovec & Sosič (2014)) [54][34], as well as our work to further speed up BigClam in section 5.2.

B.1 OpenMP constructs

OpenMP features the following constructs (as far as this project requires, further information is available from OpenMP ARB Corporation (2013) [40]), also illustrated in figure B.1:

Parallel Control & Work Sharing OpenMP supports loop-level parallelism, which allows instructions in different iterations of a loop to be executed by multiple threads in parallel. It operates under the "fork-join" model (Barney, 2015) [6]: upon encountering a parallelised loop (e.g. one preceded with declaration `#pragma omp parallel for`), the current thread (master) will spawn a number of new threads (workers), and assign iterations of the loop to the workers. Loop iterations are assigned in a "first come first served" fashion and not necessarily in order: in the case of a for-loop, a worker might be assigned to work with a loop body associated with the 1st, 42nd, 378th, 5691st, 163rd... iterations until all loop iterations are dealt with. Each




```
// Single-threaded code
...

#pragma omp parallel for
for (i=0; i<largeNo; i++) {
  // Code w/ long runtime
  ...
  ...
  ...
  ...
  #pragma omp critical
  {
    // Critical section:
    // Writing to shared obj
    // One thread at a time
  }
  // More (non-critical) code
  ...
  ...
}
```

Figure B.1: Illustration of loop-level parallelism and synchronisation in OpenMP.

thread, upon being assigned work, will work independently on the loop body in parallel, subject to constraints on the data environment and synchronisation (see below).

Data Environment OpenMP introduces two types of variables:

- Private variables: Each thread spawned (including the master thread) will have its own set of private variables. Private variables usually include the loop counter, variables declared within the loop's scope, and other variables which are explicitly declared private. These variables are immune from race conditions, as threads are unable to access private variables owned by another thread.
- Shared variables: All other variables which are in-scope when the loop begins but not private, for example variables declared outside a loop which are not explicitly declared private, are shared between all threads. These variables are vulnerable to race conditions, and require synchronisation constructs to avoid garbled results.

Synchronisation OpenMP offers a number of standards to synchronise work between threads, including the `#pragma omp critical` declarative which marks a critical section, allowing only one thread to execute instructions within the section at a time to prevent race condition.

B.2 OpenMP Use In This Project

Based on the idea on introducing locks on shared objects in concurrent executions discussed by Magee and Kramer (2006) [35], we mark sections involving read/write operations on shared variables which may be altered throughout execution critical. In other words, we parallelise all operations that involve private variables and shared variables which are read-only throughout the entire program. This allows for maximal parallelism while preventing incorrect results or memory corruption due to race conditions between different threads.