Community Detection in Social Networks with an Application to Covid-19 Data

by

Ashani Nuwanthika Wickramasinghe

A Thesis submitted to the Faculty of Graduate Studies of
The University of Manitoba
in partial fulfilment of the requirements of the degree of

Master of Science

Department of Statistics
University of Manitoba
Winnipeg

Abstract

Social network analysis (SNA) is a data analytic field that investigates hidden structures using the baseline of networks and graph theory. It helps to understand the nature of creating connections between the objects. Within a network, there can be multiple sub-networks which are called as 'communities', and there are various algorithms to find communities within a network. In this thesis, we analyze an epidemic spread using social network analysis, based on the data from the COVID-19 outbreak across the world and in Canada. We assess the nature of the spread of this virus by detecting communities using different community detection methods which can be applied on directed networks; Louvain, Label propagation, Infomap, and Spinglass algorithms. We then evaluate the performance of the community detection algorithms using simulation studies. We also assess the impact of the density and sparsity of the network on community detection by introducing a novel random partition graph generator using a mixture of two Gaussian distributions.

keywords: Social Network Analysis, Community detection, Similarity measures, Random partition graphs generator, Mixture of Gaussian distributions, COVID-19

Acknowledgment

First and foremost I would like to express my sincere gratitude to my supervisor, Dr. Saman Muthukumarana for his invaluable advice, continuous support, and patience during my Master's study. It was a privilege to work under his supervision.

I would also like to thank my advisory committee members, Dr. Liqun Wang and Dr. Cuneyt Akcora, for allocating their time to review and giving suggestions and comments to improve my research.

My gratitude extends to the staff, supporting staff, and my colleagues in the Department of Statistics, University of Manitoba. Also, I would like to thank the Department of Statistics, University of Manitoba, and Mitacs Accelerate Program for the financial support provided over the last two years.

Last but not least, I would like to thank my family who believed in me and encouraged me to achieve my dreams. My special thank goes to my dear husband who always motivated and supported me during the last two years with patience.

Dedication Page

This work is dedicated to my father, mother, sister, brother, and loving husband, whose unconditional love and endless support have enriched my soul to pursue and complete this research.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Social network analysis (SNA) has been used as an essential tool in multi-disciplinary fields. If we can observe a relationship between two objects, a network can be created based on those objects. The process of investigating social structures through the use of these networks is defined as Social Network Analysis (Otte and Rousseau, 2002). SNA works as a great visualization tool that helps understand the nature of connections between objects and identify communities within a network. It also helps to identify factors that significantly influence a link to be created between two nodes. Within a network, there can be multiple communities such that nodes inside a community are densely connected, and there are various algorithms to find communities within a network. One of the main usages of social network analysis is describing disease transmission during epidemics. Many empirical studies in humans (Rohani et al., 2010; Stehlé et al., 2011) and animals (Craft, 2015; White et al., 2018) have found the importance of social network structure on epidemiology. Eames and team (Eames et al., 2012) found that during the 2009 H1N1v influenza epidemic, changes in contact patterns explain changes in disease incidence.

Around the world, many researchers in research centers are working tirelessly to find better ways to understand and stop the spread of COVID-19. Governments are trying to control the spread by imposing travel restrictions, and multiple research works are being done to identify the importance of having travel restrictions. Yilmazkuday (Yilmazkuday, 2020), Jacob Burns (Burns et al., 2020), Matteo Chinazzi (Chinazzi et al., 2020) and others have done research related to the travel restriction during the COVID-19 pandemic. But not many people have considered understanding the spread of COVID-19 among countries before the travel restrictions. Hence in this thesis, we conducted a network analysis using COVID-19 mobility data before the travel restrictions.

Identification of sub-networks within a network is essential to understand the functionality of a network. This process is called as 'Community detection'. Communities have different properties than the average properties of the whole network. Hence Only considering the average properties of a network will lose many important features of the network. Some processes, such as disease transmission on a network, considerably affect by existing communities. Hence it is important to detect communities, and we focus on that area in this thesis.

1.1 Motivation

Starting from the early 2000, many community detection algorithms have been introduced and studied in multidisciplinary areas. The main types of applications of community detection in networks are recommendation systems (Zanin et al., 2008), link predictions (Tan et al., 2014), and anomaly detection in online social networks (Savage et al., 2014). However, evaluating the results of a community detection algorithm is a difficult task when we don't have information about the true communities (unsupervised). Different algorithms can generate significantly different communities based on their own algorithms, and it is not guaranteed that the output communities are the real sub-groups of a complex network. In the literature, the most common way to evaluate the community detection algorithm is comparing the algorithm's result with the ground truth communities of that network (Jebabli et al., 2018; Rossetti et al., 2016; Yang and Leskovec, 2015). The main similarity metrics which generally use to assess the similarity are Adjusted Rank Index (ARI) (Rand, 1971), and Normalized Mutual Index (NMI) (Zhang, 2015).

In this thesis, we propose a novel method to evaluate and compare the results of community detection algorithms using topological features of the community graphs. In this method, we evaluate the algorithms by generating synthetic networks with known communities, using the same topological features of the real world network. Then we assess the similarity between known communities and the communities produced by the algorithm. We observe the effect of this similarity score with the change of different topological features and generate a heat map to understand the variation of similarity with the change of two features.

The main challenge in developing the new method is the generation of synthetic networks with known communities that follow the network structure of a disease transmission network. The other thing that we had to consider is generating directed networks. One of the existing methods, Gaussian random partition (Brandes et al., 2003) was a better option to satisfy our requirements. This algorithm generates networks based on the number of nodes (n), the mean number of nodes per community (s), shape parameter (v), the probability of connecting nodes within a community (P_{in}), and the probability of connecting nodes between communities (P_{out}). The parameter s draws from a normal distribution. Hence, one of the main drawbacks of this network generator is the number of nodes that belong to a community is almost similar to all other communities. To overcome this, we introduce a new random partition graph generator with a mixture of Gaussians. Instead of drawing parameter s from a single Gaussian distribution, we draw s from a mixture of two Gaussian distributions.

1.2 Thesis Overview

In Chapter 2, we explore the theories of social network analysis and random network generators. The novel random network generator with a mixture of two Gaussian distributions is well explained in 2.5.2. We discuss the theory behind the community detection algorithms and similarity score metrics, which we used throughout the thesis, in Chapter 3.

We then present our case study using COVID-19 data in Chapter 4. In Chapter 5, we focus on the simulation study. In section 5.2, and 5.3 we explain the simulation process for dense and sparse networks respectively. We explain the creation of a heat map in section 5.4. Finally, we conclude the thesis with a conclusion of our results.

Chapter 2

Social Network Analysis

2.1 Introduction

Social Network Analysis (SNA) helps to understand the social structure, through graph theory and network modeling techniques. SNA is considered as the application of network science on social networks. Understanding the basic concepts and terms of SNA would be helpful to investigate the results of network analysis. Hence, throughout this section, we discuss the basic concepts and models of Social network analysis.

Gaussian random partition graph is one of the existing synthetic network generators with known communities (Brandes et al., 2003). This algorithm uses a single Gaussian distribution to draw the number of nodes per community. Due to this process, we will typically end up with communities with a similar number of nodes. To overcome this issue, we have introduced a new random graph generator which uses a mixture of two Gaussian distributions to draw the number of nodes per community.

2.2 Social Networks

A social network is defined as a set of n social ‘actors’ and a social relationship between each pair of actors. These actors do not always need to be people. To build a network graph, two key components are required: actor and relationship. Hence if we can observe a relationship between two objects, a network can be created based on those objects such as road networks, country networks, internet networks, etc...

A social network graph is developed using points and lines which connect those nodes. The points and lines represent the actors and relationships respectively. Mathematically, a network can be represented as 2.1, and two actors are indicated by i and j. By generating Y_{ij} values for the whole network with n actors, a **sociomatrix** can be developed: $Y \equiv \lfloor Y_{ij} \rfloor_{n\times n}$.

$$Y_{ij} = \begin{cases} 1 & \text{if there is a relationship between } i \text{ and } j. \\ 0 & \text{otherwise.} \end{cases} \tag{2.1}$$

2.2.1 Nodes and Edges

Nodes and Edges are a key concept in SNA. In a network, points and lines indicate actors and relationships. But in network science, those points are referred to as ‘Nodes’, while the lines are referred to as ‘Edges’.

Based on the type of network, the node can represent different types of ‘actors’. Nodes in a road network can represent ‘Bus stops’ or ‘Junctions’ while nodes in a travel network can represent ‘countries’. Edges also can represent various types of relationships based on the type of network. In a road network, edges can represent roads that connect ‘Bus stops’ or ‘Junctions’, while in a travel network, edges can represent passengers’ travel information between those ‘countries’.

2.2.2 Directed and Undirected Graphs

There are two types of edges, which are called as directed and undirected. When developing a network graph or network model, it is important to understand the type of edge based on your data set.

Directed graphs have edges with directions, which indicates by an arrowhead. These edges have a starting node and an ending node, which indicate a one-way relationship. In there, each edge can only be traversed in a single direction. In a travel network, when a passenger traveled from a country to another, that relationship is directed. The departure country is considered as the starting node and the arrival country is considered as the ending node. The right graph of Figure 2.1 shows an example of a directed graph.

Undirected graphs have edges with no direction, which indicates a two-way relationship. In there, each edge can be traversed in both directions. Edges of these graphs have no arrowheads. In a road network, when there is a two-way road that connects two ‘Bus stops’, it is an undirected relationship because buses can travel in both directions, from bus stop A to B and from B to A. The left graph of Figure 2.1 shows an example for an undirected graph.

Figure 2.1: Undirected and directed

2.2.3 Edge Weight

When there are multiple connections between the same two nodes, it can be represented by the weight of an edge. A network, which has edges with weights are referred to as weighted networks. For example, per day many people may travel from one country to another. Hence, if your data set has 100 people who have traveled from country A to country B, the edge which connects these two countries will have a weight of 100.

2.3 Centrality Measures

In network analysis, centrality measures help to identify the most important nodes within a graph. These are the most widely used indices based on network data. Hence centrality measures play an important role in social network analysis. There are several measures, but here we will explain degree centrality.

2.3.1 Degree Centrality

Degree centrality is the most elementary measure of node connectivity. The degree of a node is the number of edges connected to that node. The most central node has the highest degree. The degree centrality can be calculated by the Equation 2.2. For an undirected network, we can measure only the degree centrality. But in a directed network, there are three different degree measures: in-degree, out-degree, and degree.

$$C_D(i) = \sum_{j=1}^{n} Y_{ij} \tag{2.2}$$

In-degree and Out-degree Centrality

In directed graphs, we can calculate 'In-degree' and 'Out-degree' measures. In-degree is the number of connections that point inward at a vertex, and Out-degree is the number of connections that originate at a vertex and point outward to other vertices. In the left graph of Figure 2.2, node C has two edges towards it, while nodes A and B have 0 and 1 inward edges, respectively. Hence Node C has the highest in-degree. In the graph of the right panel, node A has two outward edges while B and C have only 1 and 0 outward edges, respectively. Because of that, Node A has the highest out-degree.

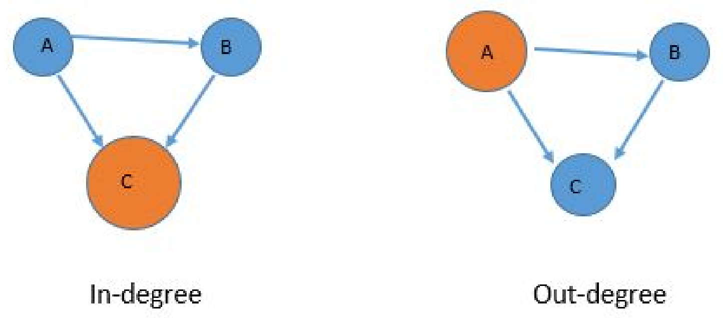


Figure 2.2: Indegree and outdegree graphs

2.4 Exponential Random Graph Models

Exponential Random Graph Models (ERGM) identify the variables that influence the link creation between nodes (Hunter et al., 2008). ERGMs are edge-based models that model the probability or weight of each edge as a function of network structure and the characteristics of individuals (nodes) within the network in network analysis (Pol, 2019). In 2018 Goeyvaerts and others (Goeyvaerts et al., 2018) used ERGM for infectious disease modeling within household members and found that fathers are less likely to be infected. The research work of Chris Groendyke and team (Groendyke et al., 2012) showed that the ERGM network model better fits the epidemic data than a Bernoulli network model previously used.

The interpretation of an ERGM is similar to the interpretation of a binary logistic regression model. ERGM predicts the probability of creating a link between a pair of nodes in a network. The basic ERGM model is fitted using edges. This is similar to fitting a regression model with only an intercept. However, since the links between nodes are not independent, it violates the basic assumption of independence of regression. The basic principle underlying the method is the comparison of an observed network to Exponential Random Graphs. A generic ERGM can be written as:

$$P_{\theta,y}(Y = y) = \frac{exp(\theta^T g(y))}{k(\theta)}. \tag{2.3}$$

Here Y is a $n \times n$ matrix, where $y_{ij} = 1$ if nodes i and j have a tie, y is the space of possible graphs, and θ is a vector of coefficients. Note that g(y) shows a vector of sufficient statistics, and $k(\theta)$ is the normalizing constant. Sufficient statistics can be node-based or network-based attributes.

2.4.1 Model Selection and Evaluation

Model selection can be done using the Akaike information criterion (AIC) (Akaike, 1973), and Bayesian information criterion (BIC) (Schwarz, 1978) values and the performance of an ERGM model can be evaluated using Goodness of Fit and MCMC diagnosis. This section will explain these model selection and evaluation methods.

For the ERGM model selection, the AIC and BIC values can be used. The model with the lowest AIC and BIC values is considered as the best model. The AIC estimates the relative distance between the unknown true likelihood function of the data and the fitted likelihood function of the model. The BIC is an estimate of a function of the posterior probability of a model being true. The following equations 2.4 and 2.5 are used to estimate the AIC and BIC of a model:

$$AIC = -2 * ln(L) + 2 * k \tag{2.4}$$

$$BIC = -2 * ln(L) + 2 * ln(N) * k \tag{2.5}$$

where L is the value of the likelihood, N is the number of recorded measurements, and k is the number of estimated parameters.

Goodness of Fit

To study the coefficients, and to extract information from the marginal plots, we need to assess the goodness-of-fit of the model. This step is essential because we can not determine how as-expected your model is behaving without it.

The goodness-of-fit procedure simulates a large number of networks using the estimated coefficients (Brandenberger and Martínez, 2019). In every one of the graphs, the bold lines in the frequency plots represent the values from the original network, while the box plots are the values from the simulated networks. Ideally, the mean of the simulated networks should overlap with the observed value. If they do not, you must return to your model and check which kinds of endogenous network terms your theory of the world might be missing.

MCMC Diagnosis

Degeneracy is one of the main problems researchers find when dealing with ERG models. Degeneracy occurs when the model places disproportionate mass on only a few of the possible graph configurations. This means that the methods used to find convergence do not work as intended, and the resulting simulations are not adequate examples of the observed graph.

Once you assess the degeneracy, all values from your sampling procedure should form a bell curve. If you observe multiple peaks for a variable, this variable is not converged, and the variable makes the model degenerate. We then have to remove it and rerun the model.

2.5 Random Graph Generation

In this section, we discuss two methods to generate random directed graphs with random partitions. Those methods are Gaussian random partition graph (Brandes et al., 2003) and our newly developed method. Gaussian random partition graph method generates random graphs based on the number of nodes, mean, and variance of cluster size. Here each community of the generated networks has a similar number of nodes. It happens because we use single Gaussian distribution to draw the size of a community. Hence we introduced a new method that can draw the size of a community using the mixture of two Gaussian distributions.

2.5.1 Gaussian Random Partition Graph

A Gaussian random partition graph (Brandes et al., 2003) is created by creating k clusters each with a size drawn from a normal distribution with mean (s) and standard deviation (s/v). Here v is the shape parameter. Nodes are connected within clusters with probability P_{in} and between clusters with probability P_{out}. The number of clusters depends on the number of nodes (n), mean (s), and standard deviation (s/v) values and the size of the last cluster is possibly significantly smaller than the others.

Let $G = (V, E)$ is a connected, undirected graph; $|V| = n, |E| = m$ and $C = (C_1, ..., Ck)$ is clustering of G and C_is are clusters. The graph $G[C_i] := (C_i, E(C_i))$, where $E(C_i) := \{\{u, w\} \in E : u, w \in C_i\}$. Then $E(C) := \bigcup_{i=1}^{k} E(C_i)$ is the set of intra-cluster edges and $E(\bar{C}) := E \backslash E(C)$ the set of inter-cluster edges. For a given n,s,v and P_{in} and P_{out} a uniformly random clustered graph (G, C) is generated by inserting intra-cluster edges with probability P_{in} and inter-cluster edges with probability P_{out}. A clustered graph is defined as (G, C), where G indicates graph and C indicates clusters. The number of edges in graph G is indicated by m, while the number of intra-cluster edges and inter-cluster edges are indicated by *m(C)* and $\bar{m}(C)$ respectively. For a generated clustered graph, the expected values of *m(C)*, and $\bar{m}(C)$ can be obtained as follows

$$E[\bar{m}(C)] = \frac{P_{out}}{2}(n(n-s)) \ \ and \ \ E[m(C)] = \frac{P_{in}}{2}(n(s-1)). \tag{2.6}$$

We develop a simulation study to obtain the empirical understanding of the *m(C)*, and $\bar{m}(C)$. As the first step, we generated a graph using the Gaussian random partition graph and calculated the number intra edges by identifying the number of edges that connect the nodes in the same cluster. The number of inter edges was calculated by subtracting the number of intra edges from the total number of edges of that network. Then the number of inter-edges and intra-edges in two different vectors were stored and repeated the same procedure 5000 times. This will end up with a distribution of inter-edges and intra-edges with 5000 data points for each. The expected value of a variable is known as the mean of that variable (Papoulis, 1984). Using trace plots and density plots of inter and intra edges, we can identify the expected values of *m(C)* and $\bar{m}(C)$.

We generated random networks by setting n=70, s=10, v=10, $P_{in} = 0.5$, and $P_{out} = 0.2$. According to the equation 2.6 the calculated values of $E[m(C)]$ and $E[\bar{m}(C)]$ should be equal to 157.5 and 420. According to the trace plot in Figure 2.3, it shows that number of inter edges varies around 420, while the number of intra edges varies around 150. The mean values of the distributions of *m(C)* and $\bar{m}(C)$ are 149.025 and 423.124 respectively. In order to have a better understanding we plotted density plots for both $m(C)$ and $\bar{m}(C)$ and Figure 2.4 shows those plots. Based on the density plots, it is clear that both distributions are normally distributed and mean values are around the expected values we calculated from equation 2.6.

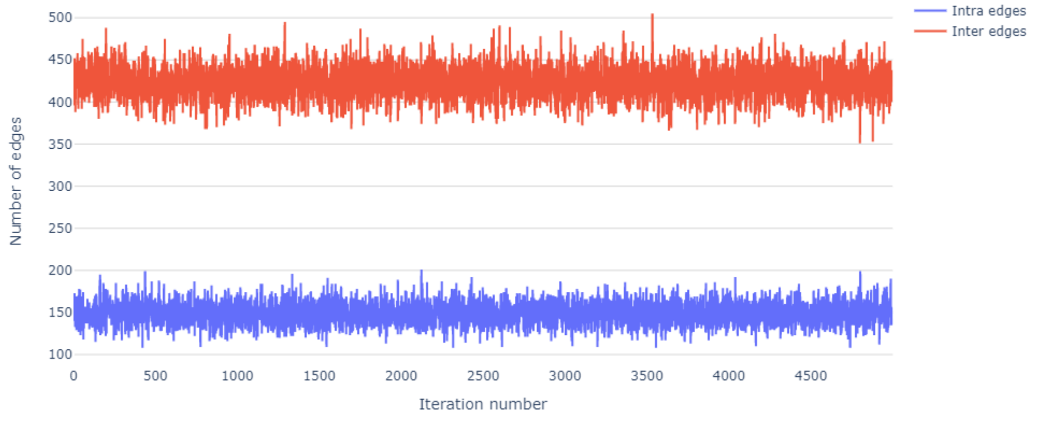
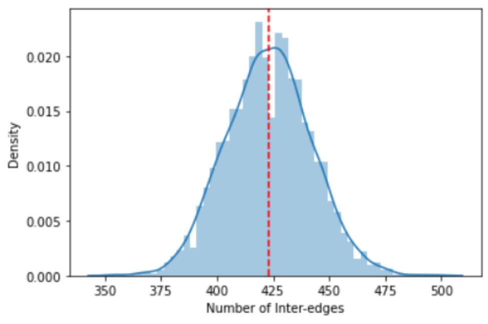
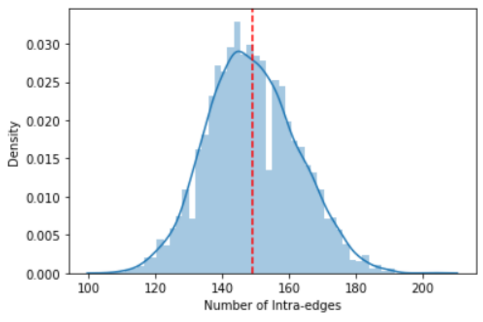


Figure 2.3: Trace plot of number of edges over iterations. Red line indicates the number of inter edges and blue color indicates the number of intra edges.

The expected values of *coverage(C)* and *performance(C)* of the graph can be obtained as equations 2.7 and 2.8. The *coverage(C)* is the fraction of intra-cluster edges within the complete set of edges. The larger the value of *coverage(C)* the better the quality of a clustering C. The *performance(C)* of a clustering C counts the number of 'correctly interpreted pairs of nodes' in a graph. Calculating the performance of clustering according to this formula would be quadratic in the number of nodes. Hence, it might be more efficient



(a)



(b)

Figure 2.4: (a): Density plot of inter-edges $\bar{m}(C)$, (b): Density plot of intra-edges *m(C)*. Red color dashed line indicates the mean value.

to find expected 'errors' instead (Eq. 2.8). Higher coverage and performance will generate high quality clusters.

$$\begin{aligned} E[coverage(C)] &= E[\frac{m(C)}{m}] = \frac{E[m(C)]}{E[m(C)+\bar{m}(C)]} \\ &= \frac{(s-1)P_{in}}{(s-1)P_{in}+(s-1)P_{out}} \end{aligned} \tag{2.7}$$

$$\begin{aligned} 1-E[performance(C)] &= \frac{2m(1-2coverage(C))+\sum_{i=1}^{k}|C_i|(|C_i|-1)}{n(n-1)} \\ &= \frac{(n-s)P_{out}+(1-P_{in})(s-1)}{n(n-1)} \end{aligned} \tag{2.8}$$

2.5.2 Random Partition Graph with Mixture of Gaussians

In order to induce a more general structure with disease transmission communities, we introduced a new partition algorithm with a mixture of Gaussian distributions. This method is different from 'Gaussian Random Partition Graph', as this uses a mixture of Gaussian distributions to draw the size of a community, instead of a single Gaussian distribution. This will generate network graphs with both bigger and smaller communities.

For a given number of nodes (n), cluster sizes will be drawn from a mixture of two Gaussian distributions with means $s1$ and $s2$, and standard deviations ($s1/v1$) and ($s2/v2$) respectively. Let the probability density function (PDF) for the i^{th} Gaussian distribution is $f_i(x)$. In the mixture of Gaussians, the probability of drawing from the first distribution is p and from the second distribution is $(1-p)$, where p is known as the mixing parameter. Then the probability density function of the mixture of Gaussians can be written as below:

$$\begin{aligned} f_M(x) &= (p)N\left(s_1, \frac{s_1}{v_1}\right) + (1-p)N\left(s_2, \frac{s_2}{v_2}\right) \\ &= (p).f_1(x) + (1-p).f_2(x). \end{aligned} \tag{2.9}$$

The mean of Gaussian mixture can be observed by integrating:

$$\begin{aligned}
\mu_M &= \int_{-\infty}^{\infty} x f_M(x)dx \\
&= \int_{-\infty}^{\infty} xp.f_1(x)dx + \int_{-\infty}^{\infty} x(1-p).f_2(x))dx \\
&= p\int_{-\infty}^{\infty} xf_1(x)dx + (1-p)\int_{-\infty}^{\infty} xf_2(x))dx \\
&= p.\mu_1 + (1-p)\mu_2.
\end{aligned} \tag{2.10}$$

In order to find the variance, first of all we need to find the second moment. Then using that we can find the variance of Gaussian mixture.

$$\begin{aligned}
E[M^2] &= \int_{-\infty}^{\infty} x^2 f_M(x)dx \\
&= \int_{-\infty}^{\infty} x^2p.f_1(x)dx + \int_{-\infty}^{\infty} x^2(1-p).f_2(x))dx \\
&= p\int_{-\infty}^{\infty} x^2f_1(x)dx + (1-p)\int_{-\infty}^{\infty} x^2f_2(x))dx \\
&= p.E(x_1^2) + (1-p).E(x_2^2)
\end{aligned} \tag{2.11}$$

$$\sigma_M^2 = E[M^2] - \mu_M^2 \tag{2.12}$$

As in section 2.5.1, we now do a simulation to assess the behaviour of $E[m(C)]$ and $E[\bar{m}(C)]$ values. Let n=70, $s_1 = 10$, $s_2 = 6$, $\sigma_1 = 1$, $\sigma_2 = 1$, p=0.5, $P_{in} = 0.5$, and $P_{out} = 0.2$. Once we calculated Mean and standard deviance based on equations 2.10 and 2.12, we got $\mu_M = 8$ and $\sigma_M^2 = 5$. The probability of connecting nodes within clusters is P_{in} and the probability of connecting nodes among clusters is P_{out}. We generate 5000 networks using new algorithm and stored number of intra edges and inter edges in each network.

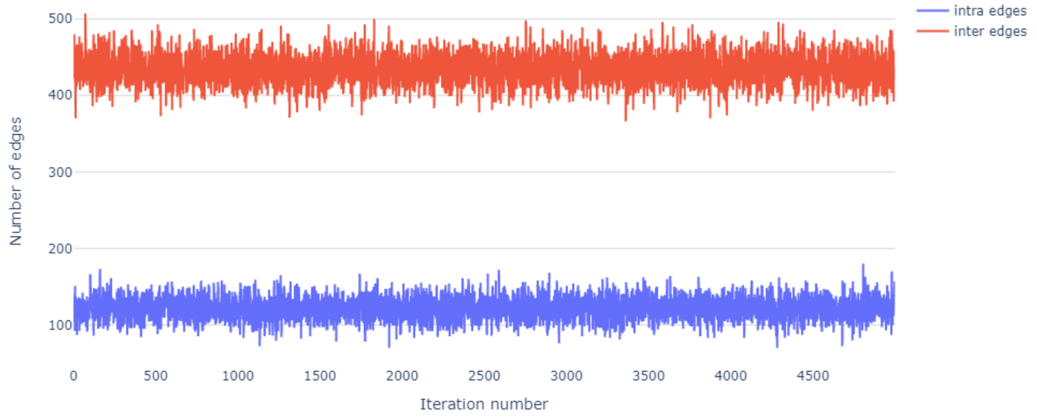
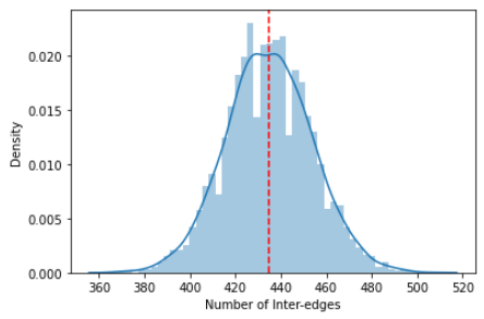
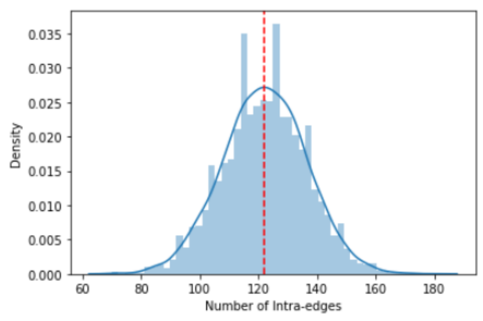


Figure 2.5: Trace plot of number of edges over iterations. Red line indicates the number of inter edges and blue color indicates the number of intra edges.

The Trace plot in Figure 2.5 shows that the number of intra edges and inter edges are varying around 120 and 450 respectively. Based on the density plots in Figure 2.6, it is clear that distributions of $m(C)$ and $\bar{m}(C)$ are symmetric around expected values. The calculated mean of $m(C)$ is 122.7, and the mean of $\bar{m}(C)$ is 433.7. Since we use a mixture of two Gaussian distributions to draw the size of a cluster, we now derive the formulas for $E[m(C)]$ and $E[\bar{m}(C)]$ in section 2.5.1 using the mean of the mixture of Gaussians which explained



(a)



(b)

Figure 2.6: (a): Density plot of inter-edges, (b): Density plot of intra-edges. Both plots are generated using data collected from random partition graphs with mixture of two Normal distributions.

in equation 2.10. Hence when the mixing parameter is equal to 0.5, the new mean is:

$$s = 0.5s_1 + (1 - 0.5)s_2 = \frac{(s_1 + s_2)}{2}. \tag{2.13}$$

For random partition graph with mixture of Gaussians, the number of intra-cluster edges and inter-cluster edges are,

$$E[\bar{m}(C)] = \frac{P_{out}}{2}\left(n\left(n - \frac{(s_1 + s_2)}{2}\right)\right) \tag{2.14}$$

$$E[m(C)] = \frac{P_{in}}{2}\left(n\left(\frac{(s_1 + s_2)}{2} - 1\right)\right). \tag{2.15}$$

Then we calculated the expected values of $m(C)$ and $\bar{m}(C)$ using equations 2.14 and 2.15. We used the same parameters that we used for the network creation (n=70, s=(10+6)/2=8, $P_{in} = 0.5$, and $P_{out} = 0.2$). The values that we observed are, 122.5 for $E[m(C)]$ and 434 for $E[\bar{m}(C)]$.

Chapter 3

Community Detection Algorithms

3.1 Introduction

The baseline idea of community detection is similar to clustering in machine learning (Wilmink and Uytterschaut, 1984). Community detection is also used to detect groups within a network. The main difference which can be seen between clustering and community detection is, that clustering uses multiple attributes to detect groups, while community detection only depends on a single attribute called edges.

Nowadays, many researchers are interested in understanding the community structure using community detection (Chakrabort et al., 2016; Li et al., 2018) as it is important to understand the nature of the spread of a virus, especially during an epidemic. There are various kinds of community detection techniques, and some techniques have their own assumptions. Community detection algorithms can be applied to either directed graphs or undirected graphs. In this study, we mainly consider community detection algorithms, that can be applied to directed graphs. Hence in this section, we describe four important community detection algorithms that can be applied on directed networks. And also, we are explaining theories of similarity measures that can be found throughout the thesis.

3.1.1 Modularity

Most of the algorithms are developed based on Modularity. Because of that, understanding the meaning of modularity is important. It measures the strength of the division of a network into groups. Modularity is defined as,

$$Q = \frac{1}{4m} \sum_{i,j \text{ } in \text{ } same \text{ } module} \left(A_{ij} - \frac{k_i k_j}{2m} \right). \tag{3.1}$$

Here m is the number of edges of the graph, k_i is the degree of node i, and A_{ij} is the adjacency matrix. The above equation shows modularity for an undirected graph. (Dugué and Perez, 2015) developed a new modularity for a directed graph, is given by,

$$Q_d = \frac{1}{m} \sum_{i,j \text{ } in \text{ } same \text{ } module} \left(A_{ij} - \frac{k_i^{in} k_j^{out}}{m} \right). \tag{3.2}$$

The main difference between these two modularities is when calculating directed modularity in-degrees and out-degrees were considered, but when calculating in undirected modularity, only the degree centrality is considered.

3.2 Louvain Algorithm

The Louvain method (Blondel et al., 2008) is an unsupervised algorithm for detecting communities in networks. This has 2 phases; Modularity Optimization and Community Aggregation. These two phases are executing until there are no more changes in the network, and the maximum modularity is achieved. At the Modularity Optimization phase, each node is assigned to its own community. Then node i is removing from its own community and moving it into the neighbor community j. The change in modularity is calculated using,

$$\Delta Q_d = \frac{d_i^c}{m} - \left\lfloor \frac{d_i^{out} . \sum_{tot}^{in} + d_i^{in} . \sum_{tot}^{out}}{m^2} \right\rfloor . \tag{3.3}$$

Here d_i^c is the degree of i in community C and $\sum_{tot}^{in}$ shows number of incoming edges of community C, while $\sum_{tot}^{out}$ shows number of outgoing edges of community C. If no positive increase can be seen in modularity, the node i remains in its current community. This sequence will be repeatedly performed for all nodes until no increment in modularity can be seen. The 1st phase stops when a local maximum has been found.

In the community aggregation step, different communities are considered as nodes. Then a new network is built by considering the communities from the previous phase are as nodes. When the new network is created, the second phase has ended, and the first phase can be re-applied to the new network. These phases are carried out until there is no more change in the community, and a maximum of modularity is achieved.

3.3 Infomap Algorithm

This algorithm repeats the two described phases in Louvain until an objective function is optimized (Fang and Liu, 2015). Instead of modularity, this algorithm optimizes the **'Map equation'**. Louvain algorithm finds community structure by minimizing the description length of a random walker's movements on a network. This random walker randomly moves between each node of the network. The random walker would like to move through the highly weighted edges. Hence, the weights of the connections within the community are greater than the weights of the connections between nodes of different communities.

The definition of the map equation is based on Shannon's Source Coding Theorem, from the field of Information Theory (Rosvall et al., 2009). Here each module has a **'module codebook'** and these module codebooks have **'codewords'** for the nodes within each module, which are derived from the node visit/exit frequencies of the random walker. The **'index codebook'** has codewords for the modules, which are derived from the module switch rates of the random walker. The map equation shows that the average length of the code (a step of the random walker) is equal to the average length of codewords from the index codebook and the module codebooks weighted by their rates of use.

$$L(M) = q_{\curvearrowright} H(Q) + \sum_{i=1}^{m} p_i \circlearrowright H(\rho_i) \tag{3.4}$$

A network with n nodes and m clusters is showed by M, and $L(M)$ shows the per-step description length for module partition of that. Note that $q_{\curvearrowright}$ and $p_i \circlearrowright$ show the rate of use of index codebook and the rate of use of module codebook respectively, while the $H(Q)$ and $H(\rho_i)$ show the frequency-weighted average length of codewords in the index codebook and module codebook i respectively.

3.4 Label Propagation Algorithm

The Label Propagation algorithm (LPA) (Raghavan et al., 2007) is one of the fast semi-supervised algorithms for finding communities in a graph, and the algorithm is based on the assumption that nodes near each other are expected to have similar class labels. The algorithm works as follows; Every node is initialized with a unique community label (an identifier), and these labels propagate through the network. At the propagation step each node update the label based on the labels of their neighbor's. When there are ties, labels are selected uniformly and randomly. This algorithm converges when each node has the majority label of its neighbors. And this stops if either convergence or the use defined maximum number of iterations is achieved.

In LPA, the influence of a node's label on other nodes is determined by their respective closeness and the closeness between nodes is measured by (3.5). Here the euclidean distance between node i and j is showed by d_{ij} and σ^2 shows the parameter to scale proximity. w_{ij}s are used to generate weight matrix and to perform label propagation weight matrix is converted into a transition matrix T using the t_{ij}s by (3.6).

$$w_{ij} = exp\left(-\frac{d_{ij}^2}{\sigma^2}\right) \tag{3.5}$$

$$t_{ij} = \frac{w_{ij}}{\sum_k w_{kj}}. \tag{3.6}$$

3.5 Spinglass Algorithm

Spinglass method minimizes the Hamiltonian of the network (Reichardt and Bornholdt, 2006). Spinglass has the following four requirements, and they are considered to develop communities.

(i) reward internal edges between nodes of the same group (in the same spin state).

(ii) penalize missing edges between nodes in the same group.

(iii) penalize existing edges between different groups (nodes in different spin state).

(iv) reward non-links between different groups.

Hamiltonian for spinglass is given by the following equation. Here A_{ij} is adjacency matrix of the graph. $\delta(\sigma_i, \sigma_j)$ is known as Kronecker delta function, $\delta_{\sigma_i,\sigma_j} = 1$ if $\sigma_i = \sigma_j$, and 0 otherwise. The spin state (or group index) of the node i is showed by $\sigma_i \epsilon \{1, 2, ..., q\}$. $a_{i,j}, b_{i,j}, b_{i,j}, d_{i,j}$ are the weights of the individual contributions for the above requirements, respectively.

$$\begin{aligned} H(\sigma) = &- \sum_{i \neq j} a_{ij} A_{ij} \delta(\sigma_i, \sigma_j) + \sum_{i \neq j} b_{ij} (1 - A_{ij}) \delta(\sigma_i, \sigma_j) \\ &+ \sum_{i \neq j} b_{ij} (1 - A_{ij}) \delta(\sigma_i, \sigma_j) - \sum_{i \neq j} d_{ij} (1 - A_{ij})(1 - \delta(\sigma_i, \sigma_j)) \end{aligned} \tag{3.7}$$

This algorithm tries to minimizes the energy of spinglass with the spin states being the community indices. Here Modularity is rewritten using Hamiltonian as (3.8),

$$Q = -\frac{1}{M} H(\{\sigma\}). \tag{3.8}$$

It applies the simulated annealing optimization technique on this model to optimize the modularity.

3.5.1 Simulated Annealing Optimization Techniques

The simulated annealing (SA) optimization technique is one of the most preferred methods for optimizing parameters in a model. This technique is useful in finding global optima in the presence of large numbers of local optima. The word "Annealing", is referred to the way that metals cool and anneals. Hence in 1983, Kirkpatrick introduced this method based on the physical annealing procedure in real life (Kirkpatrick et al., 1983).

Simulated annealing is a stochastic global search algorithm for function optimization. This technique aims to bring the system from the initial state to a state with the minimum possible energy. At each step of the process, the simulated annealing considers the neighboring state $\bar{s}$ of the current state s and decides whether to moving the system to state $\bar{s}$ or staying in the state s. These probabilities eventually lead the system to move to states of lower energy.

The following pseudocode shows the process of simulated annealing. It starts from a state s_1 and continues until z_{max} steps have been taken. Here $neighbor(s)$ will randomly pick the neighbor of a given state s, and the $random(0, 1)$ will return a value from uniform distribution in the range [0, 1]. The $temperature(r)$ will define the annealing schedule that should allow the temperature to use, given the fraction r of the time budget that has been expended so far.

Algorithm 1: Simulated Annealing (SA) Process

```
Let $s = s_1$;
for For $z = 0$ through $z_{max}$ (exclusive) do
    $T \leftarrow$ temperature($(z+1)/z_{max}$);
    Pick a random neighbor, $s_{new} \leftarrow$ neighbor($s$);
    if P(E(s), E($s_{new}$), T) $\geq$ random$(0,1)$ then
        $s \leftarrow s_{new}$
    end if
end for
return The final state $s$
```

3.6 Similarity Measures

Once identified the communities based on different algorithms, a comparison study can be done using similarity measures. Those indices measure the similarity between community detection results of two algorithms.

3.6.1 Adjusted Rand Index

The Rand Index computes a similarity measure between two clustering by considering all pairs of samples and counting pairs that are assigned in the same or different clusters in the predicted and true clustering. If the number of data vectors for clustering is n, then there are $\binom{n}{2}$ pairs. For every pair of examples, there are three possibilities in terms of grouping. The first possibility is that the paired examples are always placed in the same group, as a result of clustering (a). The second possibility is that the paired examples are never grouped together (b). The third possibility is that the paired examples are sometimes grouped and sometimes not grouped together. The RI of two groupings is then calculated by the following formula:

$$RI = \frac{\text{Count of Pairs in Agreement}}{\text{Total Number of Pairs}} = \frac{a+b}{\binom{n}{2}}. \tag{3.9}$$

RI had one drawback; it yields a high value for pairs of random partitions of a given set of examples. To overcome this drawback, The Rand Index score is then 'adjusted for chance' into the Adjusted Rand Index (Rand, 1971) score using the following scheme:

$$ARI = \frac{RI - \text{Expected RI}}{\text{max(RI)} - \text{Expected RI}}. \tag{3.10}$$

The adjusted Rand index is thus ensured to have a value close to 0.0 for random labeling independently of the number of clusters and samples and exactly 1.0 when the clusterings are identical (up to a permutation).

3.6.2 Normalized Mutual Information

Mutual Information (MI) is a measure of the similarity between two labels of the same data. Where $|U_i|$ is the number of the samples in cluster U_i and $|V_j|$ is the number of the samples in cluster V_j, the Mutual Information between clustering U and V is given as:

$$MI(U,V) = \sum_{i=1}^{|U|} \sum_{j=1}^{|V|} \frac{|U_i \cap V_j|}{N} log \frac{N|U_i \cap V_j|}{|U_i||V_j|}. \tag{3.11}$$

This metric is independent of the absolute values of the labels: a permutation of the class or cluster label values won't change the score value in any way. This metric is furthermore symmetric and can be useful to measure the agreement of two independent label assignments strategies on the same data set when the real ground truth is not known. Normalized Mutual Information (NMI) (Zhang, 2015) is a normalization of the Mutual Information (MI) score to scale the results between 0 (no mutual information) and 1 (perfect correlation).

3.6.3 Adjusted Mutual Information

The baseline value of mutual information between two random clusterings tends to be larger when the two partitions have a larger number of clusters (with a fixed number of nodes). Hence Adjusted Mutual Information (AMI) (Vinh et al., 2010) will be able to adjust the Mutual information (MI) score to account for a chance. The AMI between clustering U and V is given as:

$$AMI(U,V) = \frac{MI(U,V) - E\{MI(U,V)\}}{max\{H(U),H(V)\} - E\{MI(U,V)\}}. \tag{3.12}$$

This metric is independent of the absolute values of the labels: a permutation of the class or cluster label values won't change the score value in any way. Here H(U) and H(V) indicate the entropy associated with the partitioning U and V. The AMI takes a value of 1 when the two partitions are identical and 0 when the MI between two partitions equals the value expected due to chance alone.

Chapter 4

Case Study Using COVID-19 Data

4.1 Introduction

On March 11, 2020, the World Health Organization (WHO) declared the novel coronavirus (COVID-19) outbreak as a global pandemic, which started to spread from Hubei, China (Liu et al., 2020). Human coronaviruses cause infections of the nose, throat, and lungs. It is most commonly spread from an infected person through close contacts, and symptoms may take up to 14 days to appear after exposure to COVID-19. It started to spread across the world as the patients who did not start to show any symptoms have traveled across countries. Finally, countries had to lock down to stop the spread. By the end of April 2020, the virus has spread in more than 150 countries.

Historically, human behavior was responsible for the spread of many diseases. To stop the spread of bubonic plague, which is called as the black death, citizens of the Yorkshire village of Eyam in England had to voluntarily quarantine themselves (Scott Duncan 2001 (Scott and Duncan, 2001)). More recently, during the influenza pandemic, people stayed away from crowded places ((Crosby, 2003)). In the early twenty-first century, the usage of face masks became increased, and traveling behaviors were changed, due to severe acute respiratory syndrome broke out ((Lau et al., 2005)). Thus, in general, to control the spread of viruses and to 'flatten the curve', travel is restricted during pandemics.

In this Chapter, we explain the epidemic spread using social network analysis, based on data from the first three months of the COVID-19 outbreak across the world and in Canada, which has already published in (Wickramasinghe and Muthukumarana, 2021). A network is defined and graphically visualized to understand the spread of coronavirus among countries and the impact of other countries on the spread of coronavirus in Canada. The degree centrality is used to identify the main influencing countries. Exponential Random Graph Models (ERGM) are used to identify the processes that influence link creation between countries. The community detection is done using four different algorithms, and the performance of those algorithms was assessed using the adjusted rand index and normalized mutual information score.

4.2 Data Sets

At the beginning of this study, the main challenge was finding datasets that contain information about travel history, as it was the main attribute to create links between countries. The data used in this research were obtained from two GitHub repositories. The data sets regarding the spread of COVID-19 across the world and in Canada were downloaded from the links given in (Rajkumar, 2020) and (Berry and Soucy, 2020a). The first data set contains information about patients across the world starting from 22 Jan 2020 as multiple updated versions. This dataset, which is a matrix of 13174 (rows) by 44 (columns), is extracted from the Johns Hopkins University's dashboard using the methodology of (Xu et al., 2020) and a sample of the dataset is given in Table 1. The second dataset contains information about patients in Canada starting from 25 Jan 2020. The "COVID-19 Canada Open Data Working Group" (Berry et al., 2020) are collecting these data and all the details about the sources are included in the technical report in (Berry and Soucy, 2020b). For this study, we have used January to March and January to April data points of first and second datasets respectively. Details about these countries were obtained from the worldometers.info website (Worldometers.info, 2020).

4.2.1 Data Cleaning

Here some patients' records did not have any travel information, and some did not report the country they traveled to. So they were considered as

Table 4.1: Sample of world data set

ID	**age**	**sex**	**city**	**...**	**travel_history_ location**	**...**
1	30	male	Chaohu City, Hefei City	**...**	Wuhan	**...**
2	47	male	Baohe District, Hefei City	**...**	Wuhan	**...**
3	49	male	High-Tech Zone, Hefei City	**...**	Wuhan	**...**
4	47	female	High-Tech Zone, Hefei City	**...**	Wuhan	**...**

locally infected cases. Also, according to travel history, some people had visited more than one country. So multiple rows were added for the same patient by splitting the travel history variable. In some records, the values of "travel_history_location" were cities, and we had to replace them with the countries of those cities. Finally, two new data sets were created, where "Travel location" was considered as the "Source", "Country" was considered as the "Target" and the number of patients who traveled between these locations was considered as "Weight" variable. In the Canada data set, provinces were considered separately. Hence that "Province" was considered as the "Target" variable. These new data sets were used to create social networks. Table 4.2 and 4.4 show samples of those newly generated data sets.

Table 4.2: Sample of new data from data set 1 (Across the world)

Source	Target	Weight
Australia	Australia	1
Canada	Canada	4
China	Australia	13
China	Belgium	1
China	Cambodia	1
China	Canada	5

Table 4.3: Sample of new data from data set 2 (Canada)

Source	Target	Weight
Alberta	Alberta	16
Cruise	Alberta	4
Netherlands	Alberta	1
Turkey	Alberta	1
Ukraine	Alberta	1
United States	Alberta	3

4.3 Analysis of the World Data Set

The spread of the COVID-19 across the world (within the first three months),was plotted on the world map. Figure 4.1 shows how this virus spread around the world, through the travel of COVID-19 patients. This plot shows that many links are connected to China and European countries. China has connections from all around the world except South America and Africa.

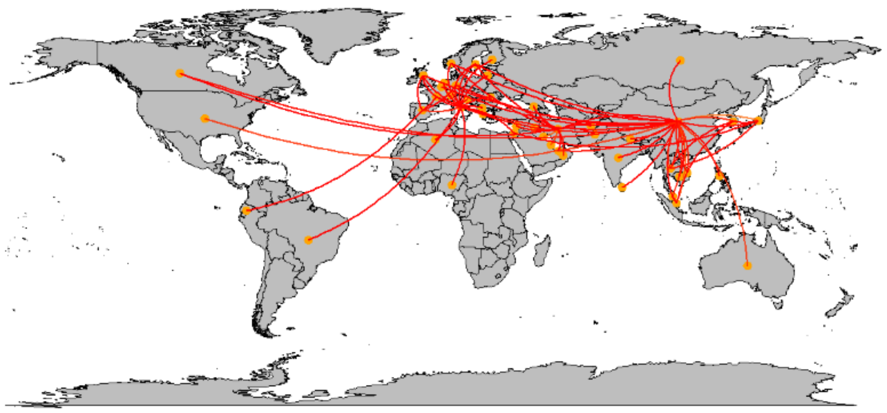


Figure 4.1: Spread of COVID-19 across the world

Based on the cleaned data set world networks were created. In this network, nodes represent each country, and a connection (link) between nodes was created when a person traveled from one location to another. The node size is indicated by “in-degree” and “out-degree” centralities. Figure 4.2 and 4.3 show indegree and outdegree networks of world network respectively.

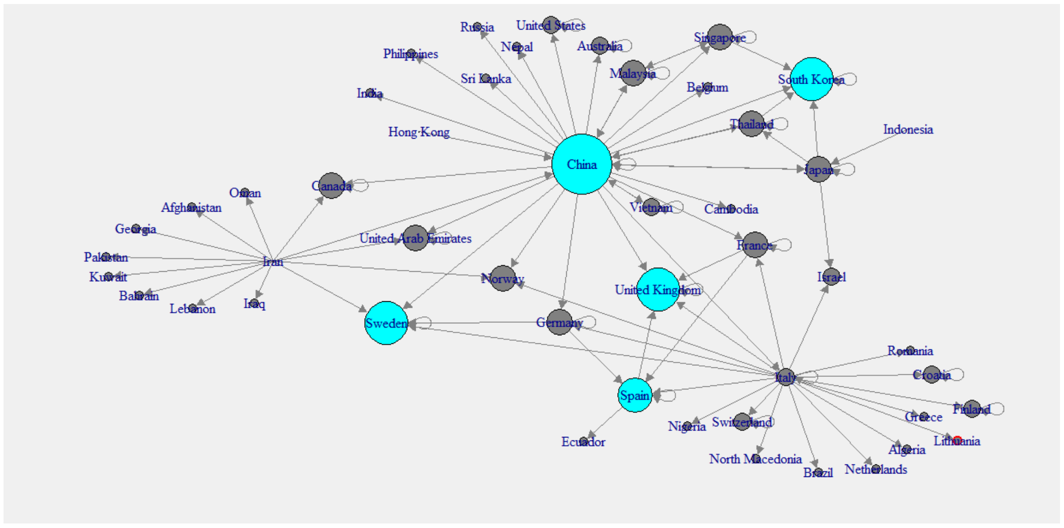


Figure 4.2: In-degree network of across the world

Cyan color nodes in Figure 4.2 indicate the countries with higher in-degree, while yellow color nodes in Figure 4.3 indicate the countries with higher out-degree. Those countries which have higher out-degree consider as the main countries where many patients came from. Patients of these countries have traveled to many countries. So those countries can be considered as the main hubs of COVID-19. Higher in-degree nodes can be used to identify the main countries which found many patients who came from different other countries.

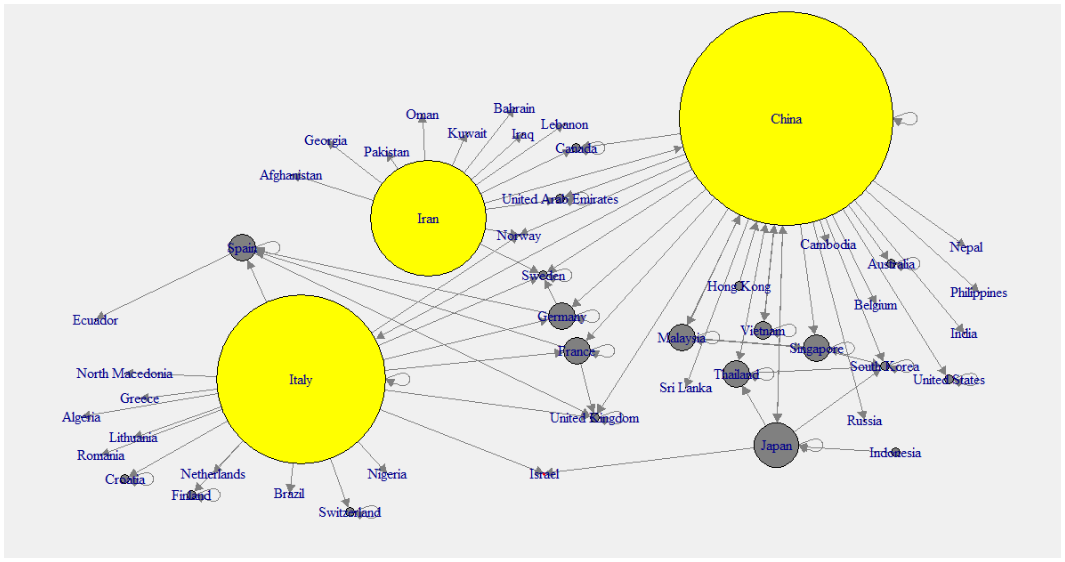


Figure 4.3: Out-degree network of across the world

4.3.1 Models Creation and Validation

After the visualization, ERGM models were fitted to identify the variables that influence link creation between countries. The models were fitted using network and node attributes. Even though there were many attributes, only a few were filtered as significant covariates. Among these models, the model with the smallest AIC and BIC values is considered as the best model. According to the Table 4.4, model 4 was considered as the best model.

According to the goodness of fit plots, which are shown in left panel of Figure 4.4, it is clear that the model 4 is achieving a good fit for key structural properties of the network with these covariates as the means of the simulated values of the plot are overlapped with the observed values. The result of MCMC diagnosis is shown in the right panel of Figure 4.4. Here all the covariates have

Table 4.4: Model comparison (Across the world)

	Model1	Model2	Model3	Model4
edges	-3.44 ***	-4.68 ***	-3.35 ***	-4.98 ***
	(0.12)	(0.23)	(0.12)	(0.26)
mutual		1.61 **		
		(0.53)		
indegree		0.04 **		0.05 **
		(0.02)		(0.02)
outdegree		0.03 ***		0.06 ***
		(0.00)		(0.00)
recover			-2.11 *	
			(1.01)	
death				-0.001 ***
				(0.00)
AIC	652.70	456.95	645.25	424.84
BIC	658.47	480.00	656.78	447.89
Log Likelihood	-325.35	-224.47	-320.63	-208.42

bell-curved density plots. The models appear to have converged to a desired state for each ERGM because the density is not skewed and is centered over zero.

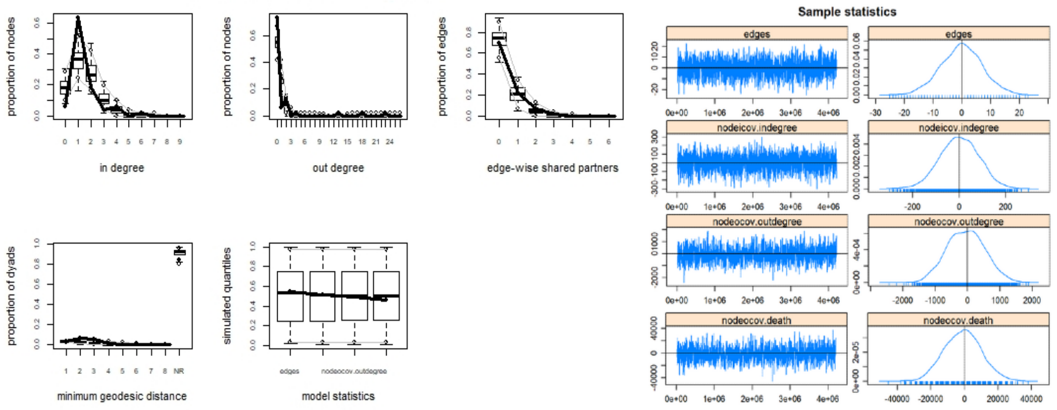
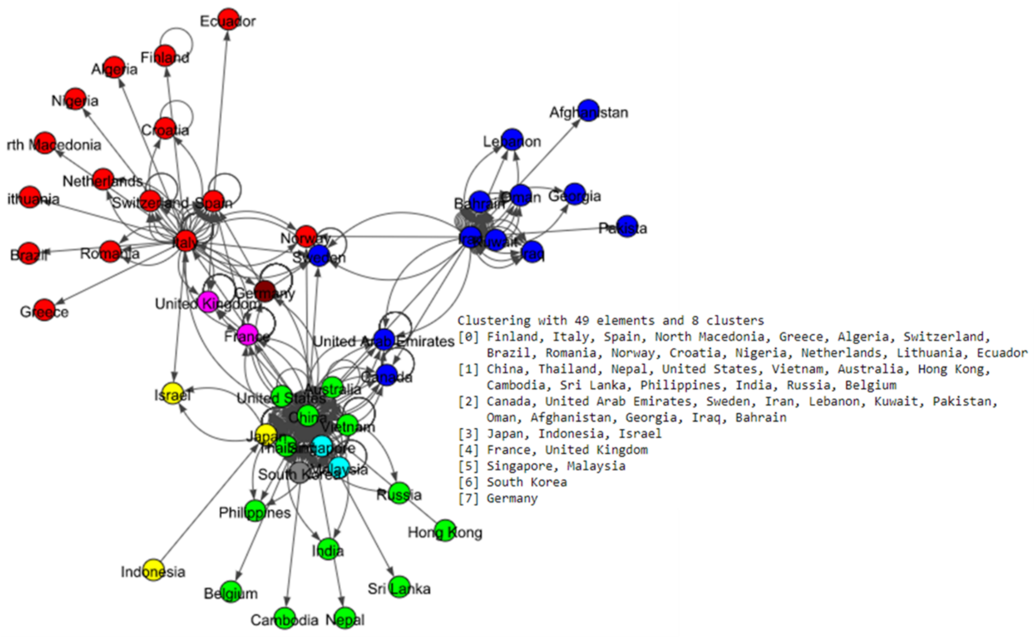


Figure 4.4: Goodness-of-fit and MCMC diagnostics for model 4

4.3.2 Community Detection using the World Data Set

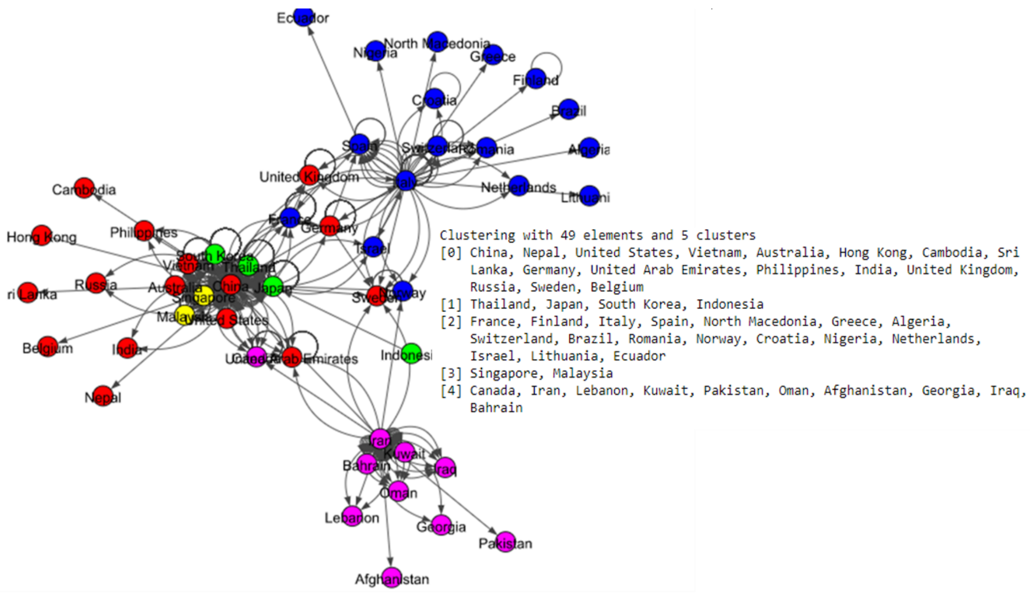
After figuring out the influencing variables on ties creation, the COVID-19 communities across the world were identified using four different algorithms. The output community labels of Louvain, Infomap, Spinglass, and Label propagation algorithms are shown in the following Figures, 4.5, 4.6, 4.7 and 4.8.



```
Clustering with 49 elements and 8 clusters
[0] Finland, Italy, Spain, North Macedonia, Greece, Algeria, Switzerland,
    Brazil, Romania, Norway, Croatia, Nigeria, Netherlands, Lithuania, Ecuador
[1] China, Thailand, Nepal, United States, Vietnam, Australia, Hong Kong,
    Cambodia, Sri Lanka, Philippines, India, Russia, Belgium
[2] Canada, United Arab Emirates, Sweden, Iran, Lebanon, Kuwait, Pakistan,
    Oman, Afghanistan, Georgia, Iraq, Bahrain
[3] Japan, Indonesia, Israel
[4] France, United Kingdom
[5] Singapore, Malaysia
[6] South Korea
[7] Germany
```

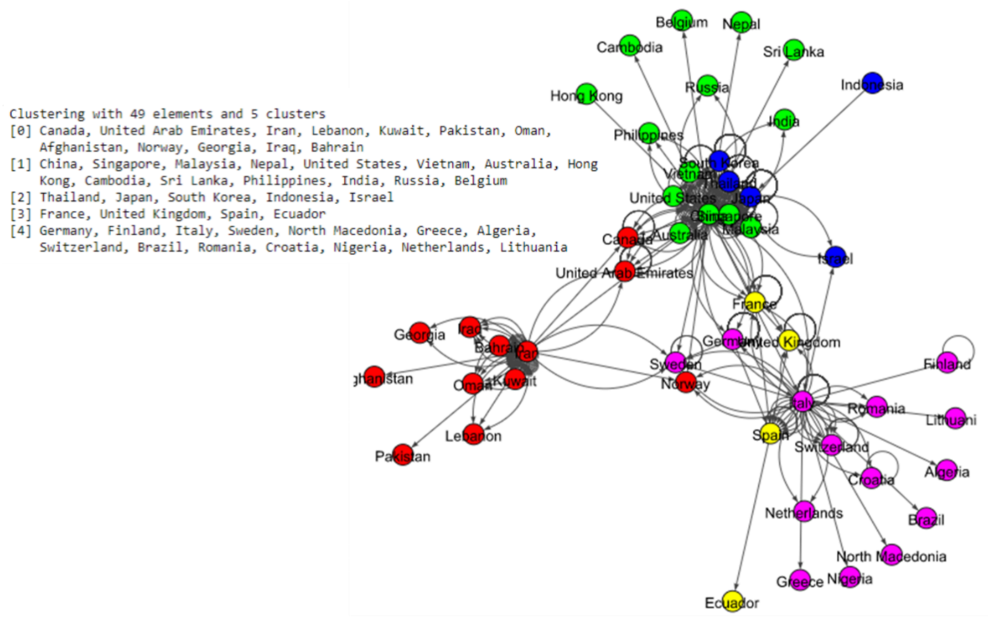
Figure 4.5: Results of Louvain algorithm for the world dataset

The Figure 4.5 shows eight communities based on the Louvain algorithm while the Figure 4.6 shows five communities based on the Infomap algorithm. Spinglass algorithm also shows five communities, which is shown in the Figure 4.7. The Figure 4.8 shows thirteen communities based on label propagation



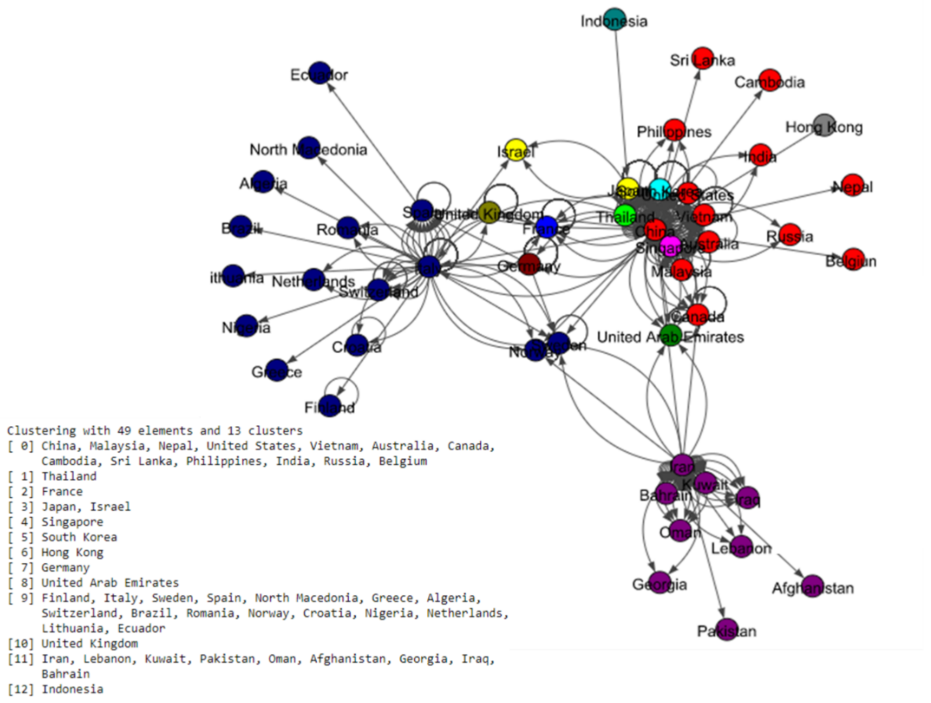
```
Clustering with 49 elements and 5 clusters
[0] China, Nepal, United States, Vietnam, Australia, Hong Kong, Cambodia, Sri
    Lanka, Germany, United Arab Emirates, Philippines, India, United Kingdom,
    Russia, Sweden, Belgium
[1] Thailand, Japan, South Korea, Indonesia
[2] France, Finland, Italy, Spain, North Macedonia, Greece, Algeria,
    Switzerland, Brazil, Romania, Norway, Croatia, Nigeria, Netherlands,
    Israel, Lithuania, Ecuador
[3] Singapore, Malaysia
[4] Canada, Iran, Lebanon, Kuwait, Pakistan, Oman, Afghanistan, Georgia, Iraq,
    Bahrain
```

Figure 4.6: Results of Infomap algorithm for the world dataset



```
Clustering with 49 elements and 5 clusters
[0] Canada, United Arab Emirates, Iran, Lebanon, Kuwait, Pakistan, Oman,
    Afghanistan, Norway, Georgia, Iraq, Bahrain
[1] China, Singapore, Malaysia, Nepal, United States, Vietnam, Australia, Hong
    Kong, Cambodia, Sri Lanka, Philippines, India, Russia, Belgium
[2] Thailand, Japan, South Korea, Indonesia, Israel
[3] France, United Kingdom, Spain, Ecuador
[4] Germany, Finland, Italy, Sweden, North Macedonia, Greece, Algeria,
    Switzerland, Brazil, Romania, Croatia, Nigeria, Netherlands, Lithuania
```

Figure 4.7: Results of Spinglass algorithm for the world dataset



```
Clustering with 49 elements and 13 clusters
[ 0] China, Malaysia, Nepal, United States, Vietnam, Australia, Canada,
     Cambodia, Sri Lanka, Philippines, India, Russia, Belgium
[ 1] Thailand
[ 2] France
[ 3] Japan, Israel
[ 4] Singapore
[ 5] South Korea
[ 6] Hong Kong
[ 7] Germany
[ 8] United Arab Emirates
[ 9] Finland, Italy, Sweden, Spain, North Macedonia, Greece, Algeria,
     Switzerland, Brazil, Romania, Norway, Croatia, Nigeria, Netherlands,
     Lithuania, Ecuador
[10] United Kingdom
[11] Iran, Lebanon, Kuwait, Pakistan, Oman, Afghanistan, Georgia, Iraq,
     Bahrain
[12] Indonesia
```

Figure 4.8: Results of Label Propagation algorithm for the world dataset

algorithm. Most of the communities are around the countries which have higher out-degree.

We now pairwise compare the performance of the four algorithms using two different metrics, namely, Adjusted Rand Index (ARI) (Rand, 1971), and Normalized Mutual Information (NMI) (Zhang, 2015). Table 4.5 represents the results of comparing four algorithms (Infomap, Label Propagation, Louvain and Spinglass) with respect to two topological metrics (ARI and NMI). The larger (smaller) the value of ARI and NMI, the more (less) similar are the two algorithms being compared.

Table 4.5: Comparison of different algorithms with respect ARI and NMI for the world network.

		Infomap	**Louvain**	**Spinglass**	**Label_Propagation**
ARI	**Infomap**	1	0.71	0.55	0.64
NMI	**Infomap**	1	0.75	0.64	0.70
ARI	**Louvain**		1	0.67	0.75
NMI	**Louvain**		1	0.74	0.81
ARI	**Spinglass**			1	0.66
NMI	**Spinglass**			1	0.73
ARI	**Label_Prop**				1
NMI	**Label_Prop**				1

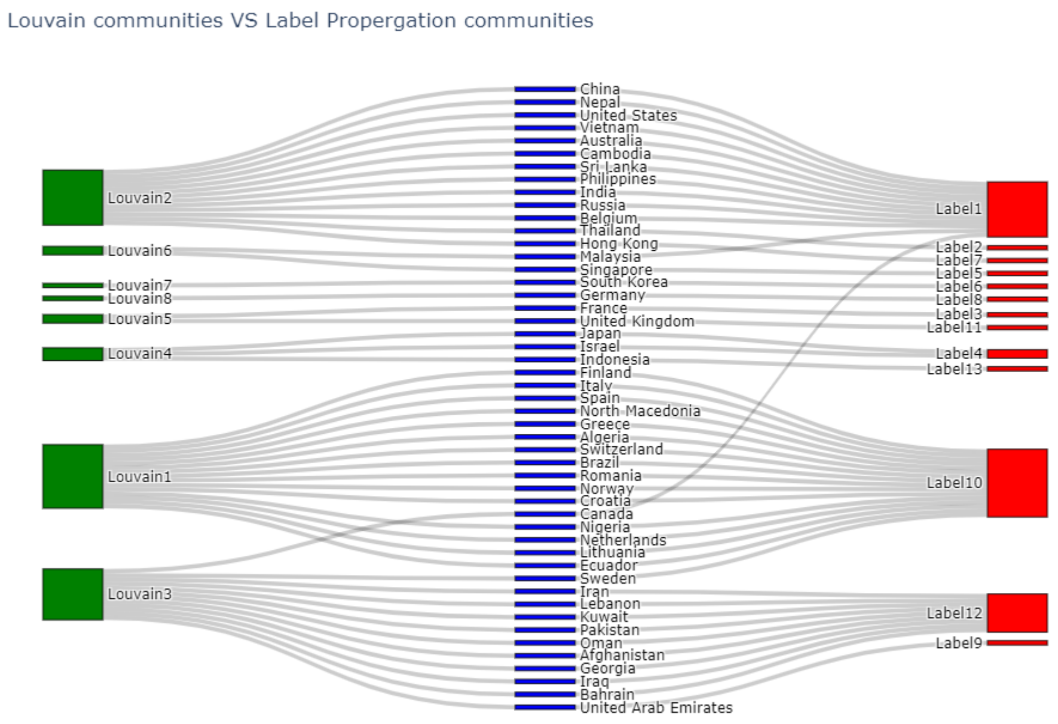


Figure 4.9: Comparing communities of Louvain and Label propagation algorithms

Louvain and Label Propagation algorithms are the most similar to each other amongst all pairs of comparisons in Table 4.5. A Sankey plot (Reda et al., 2011) which is a graphical comparative illustration of the communities of these two algorithms is given in Figure 4.9. The communities of the Louvain algorithm, and the Label Propagation algorithm are indicated by green and red color nodes respectively. Blue color nodes show all the countries of the world network.

4.4 Analysis of Canada Data Set

Same as the analysis of the world, Figure 4.10 illustrates how Canada has been affected by other countries. According to this plot, orange color dots, which indicate countries can be seen in every continent. It means patients from all over the world have visited Canada. But based on the provinces of Canada the impact is different. The province of Yukon which is located at the left upper corner, has only one link, while other states have multiple links.

In this analysis, the in-degree network graph is showed in the left panel of the Figure 4.11. Same as the above in-degree network graph, cyan color nodes indicate the higher in-degree provinces. According to this plot, the province of Ontario has the main effect from other countries. The right panel network of the Figure 4.12 shows the out-degree network in Canada. Note that yellow color indicates the nodes with higher out-degree. According to this graph, Canadian provinces are mainly affected by the United States (USA), the United Kingdom, and the Carrabian islands.

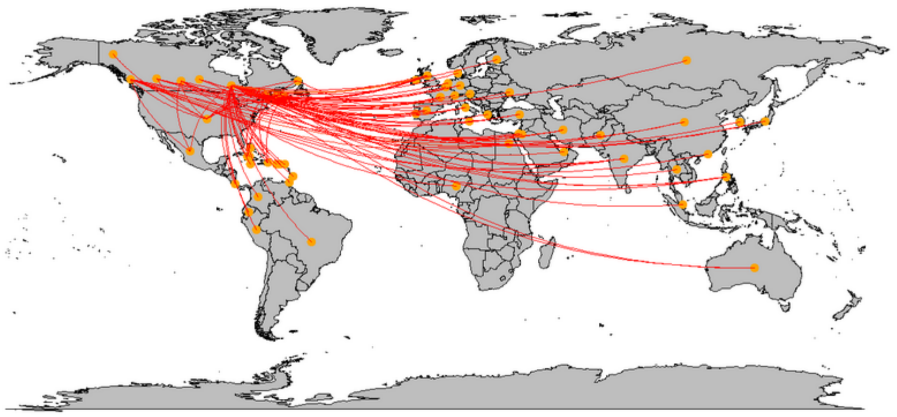


Figure 4.10: Impact of other countries on spread of COVID-19 in Canada

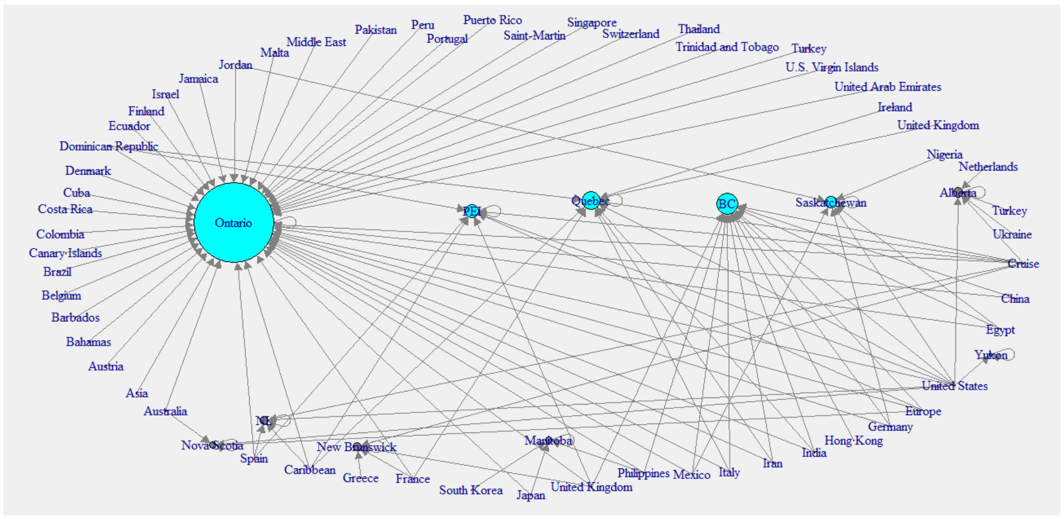


Figure 4.11: In-degree network of Canada

4.4.1 Models Creation and Validation

We fit ERGM models and filter out the significant covariates. The model 2 and model 4 have similar covariates. Hence the model 4 was not considered in this analysis. Finally, according to the AIC, and BIC values, model 2 was

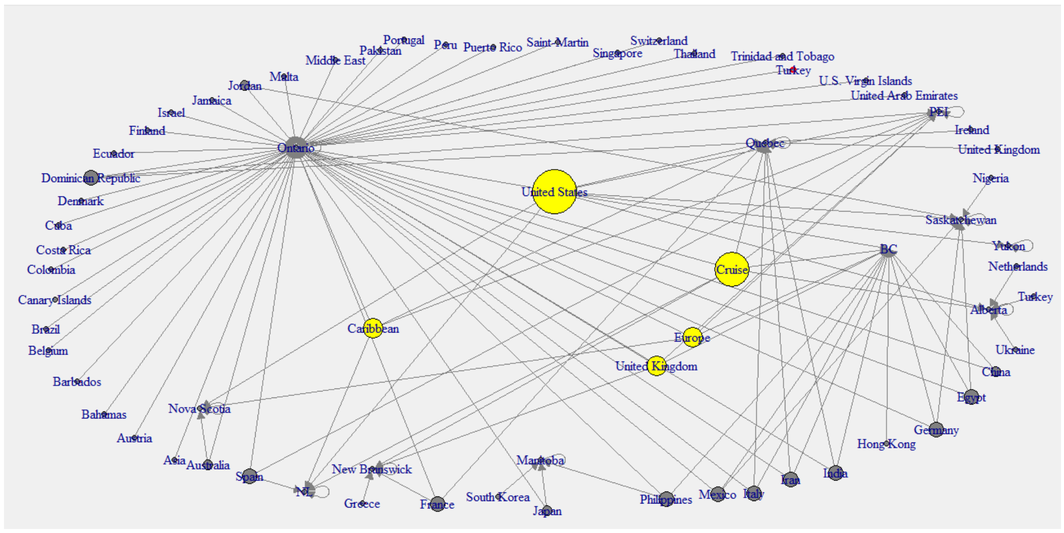


Figure 4.12: Out-degree network of Canada

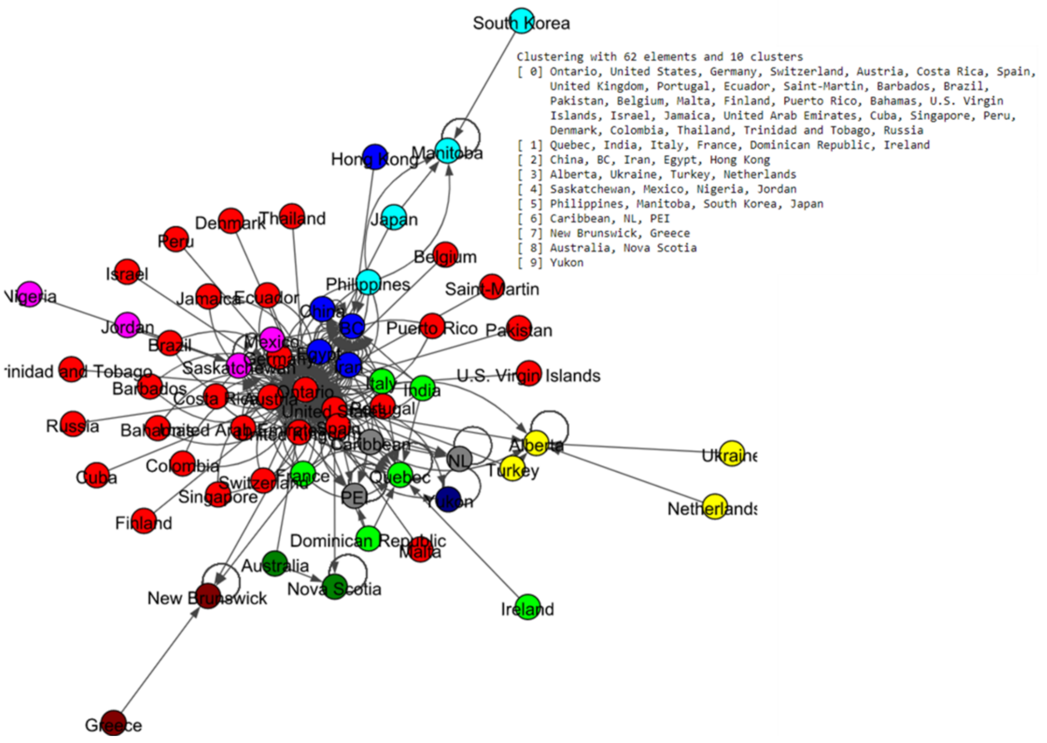
considered as the best model. Table 4.6 shows the model comparison. The models were validated using Goodness of fit and MCMC diagnostics as in Figure 4.4, and indicated that model 2 is a good fit. That also indicated the model has converged to a desired state.

Table 4.6: Model comaparison (Canada)

	Model1	Model2	Model3
edges	-3.70 ***	-4.77 ***	-3.64 ***
	(0.11)	(0.18)	(0.11)
indegree		0.02 ***	
		(0.00)	
outdegree		0.02 ***	
		(0.00)	
region			-1.41 *
			(0.72)
AIC	860.13	523.47	855.87
BIC	866.37	542.19	868.34
Log Likelihood	-429.07	-258.74	-425.93

4.4.2 Community Detection using Canada Data Set

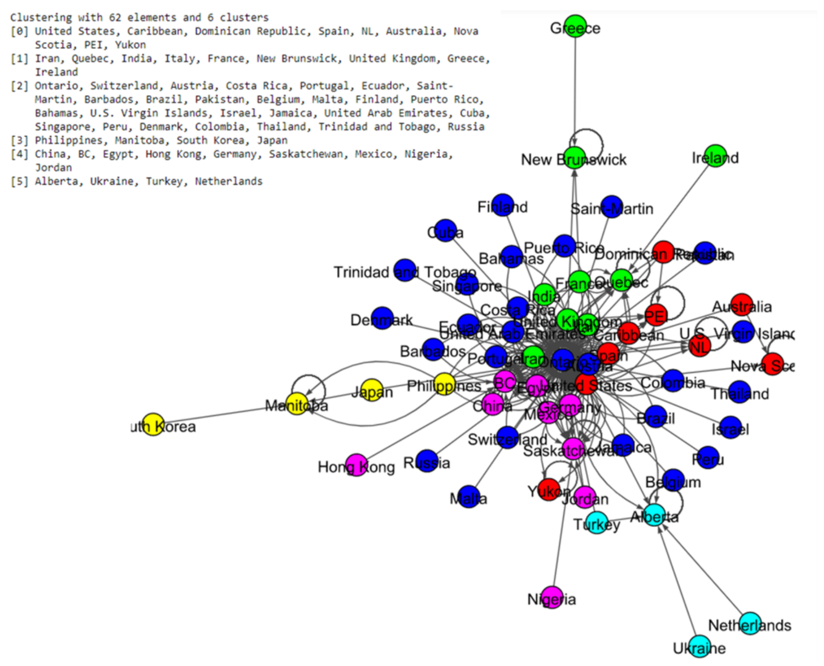
The community labels which are based on Louvain and Spinglass algorithms were plotted as in Figure 4.13 and Figure 4.14 respectively. The Louvain algorithm shows ten communities, while the Spinglass algorithm shows only six communities.



```
Clustering with 62 elements and 10 clusters
[ 0] Ontario, United States, Germany, Switzerland, Austria, Costa Rica, Spain,
     United Kingdom, Portugal, Ecuador, Saint-Martin, Barbados, Brazil,
     Pakistan, Belgium, Malta, Finland, Puerto Rico, Bahamas, U.S. Virgin
     Islands, Israel, Jamaica, United Arab Emirates, Cuba, Singapore, Peru,
     Denmark, Colombia, Thailand, Trinidad and Tobago, Russia
[ 1] Quebec, India, Italy, France, Dominican Republic, Ireland
[ 2] China, BC, Iran, Egypt, Hong Kong
[ 3] Alberta, Ukraine, Turkey, Netherlands
[ 4] Saskatchewan, Mexico, Nigeria, Jordan
[ 5] Philippines, Manitoba, South Korea, Japan
[ 6] Caribbean, NL, PEI
[ 7] New Brunswick, Greece
[ 8] Australia, Nova Scotia
[ 9] Yukon
```

Figure 4.13: Results of Louvain algorithm for the Canada data set

The left panel network of Figure 4.15 shows the result of label propagation community detection for this data set. There are 62 nodes, and this algorithm shows that all these states and countries belong to 60 communities. All-most all the nodes have their own communities. The result of the Infomap algorithm



```
Clustering with 62 elements and 6 clusters
[0] United States, Caribbean, Dominican Republic, Spain, NL, Australia, Nova
    Scotia, PEI, Yukon
[1] Iran, Quebec, India, Italy, France, New Brunswick, United Kingdom, Greece,
    Ireland
[2] Ontario, Switzerland, Austria, Costa Rica, Portugal, Ecuador, Saint-
    Martin, Barbados, Brazil, Pakistan, Belgium, Malta, Finland, Puerto Rico,
    Bahamas, U.S. Virgin Islands, Israel, Jamaica, United Arab Emirates, Cuba,
    Singapore, Peru, Denmark, Colombia, Thailand, Trinidad and Tobago, Russia
[3] Philippines, Manitoba, South Korea, Japan
[4] China, BC, Egypt, Hong Kong, Germany, Saskatchewan, Mexico, Nigeria,
    Jordan
[5] Alberta, Ukraine, Turkey, Netherlands
```

Figure 4.14: Results of Spinglass algorithm for the Canada data set

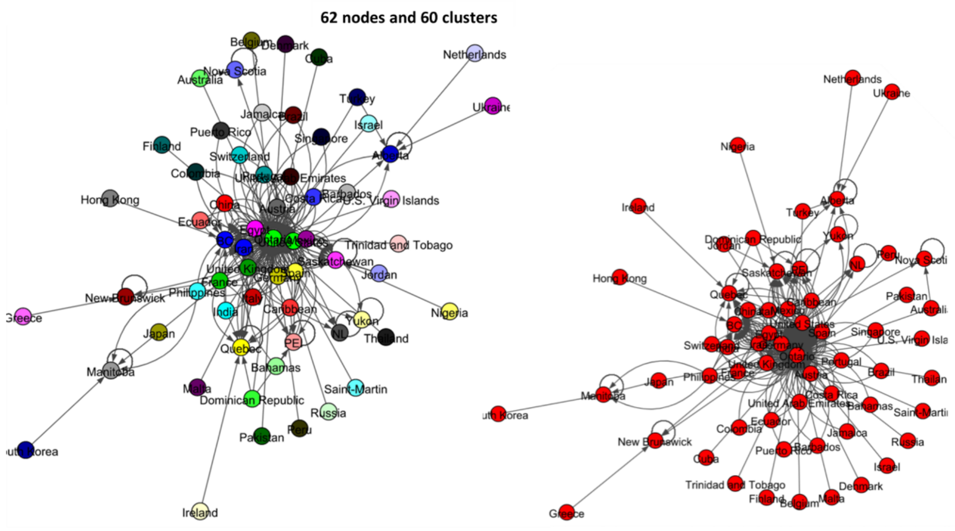


Figure 4.15: Results of Label Propagation (left panel) and Infomap (right panel) algorithms for the Canada data set

shows in the right panel of Figure 4.15 and it shows that all the countries and states of Canada belong to one community.

Pairwise comparison of above algorithms was done using ARI and NMI. Table 4.7 shows the results of comparing 4 algorithms (Infomap, Label Propagation,Louvain and Spinglass) for the Canada network.

Table 4.7: Comparison of different algorithms with respect to ARI and NMI for the Canada network.

		Infomap	**Louvain**	**Spinglass**	**Label_Prop**
ARI	**Infomap**	1	0	0	0
NMI	**Infomap**	1	3.19e-17	2.14e-16	0
ARI	**Louvain**		1	0.72	0.01
NMI	**Louvain**		1	0.74	0.60
ARI	**Spinglass**			1	0
NMI	**Spinglass**			1	0.54
ARI	**Label_Prop**				1
NMI	**Label_Prop**				1

Based on Table 4.7, Louvain and Spinglass are most similar to each other. The comparison of node-communities from Louvain and Spinglass methods is compared using sankey plots, where the green nodes correspond to the communities of Spinglass method and red nodes correspond to the communities of Louvain method.

4.5 Discussion

Within the first few months of the COVID-19 outbreak, this virus has spread rapidly over the world. According to this study, China, Iran, and Italy have appeared as most out-degree countries around the world while the United Kingdom, South Korea, Sweden, and Spain have appeared as most in-degree

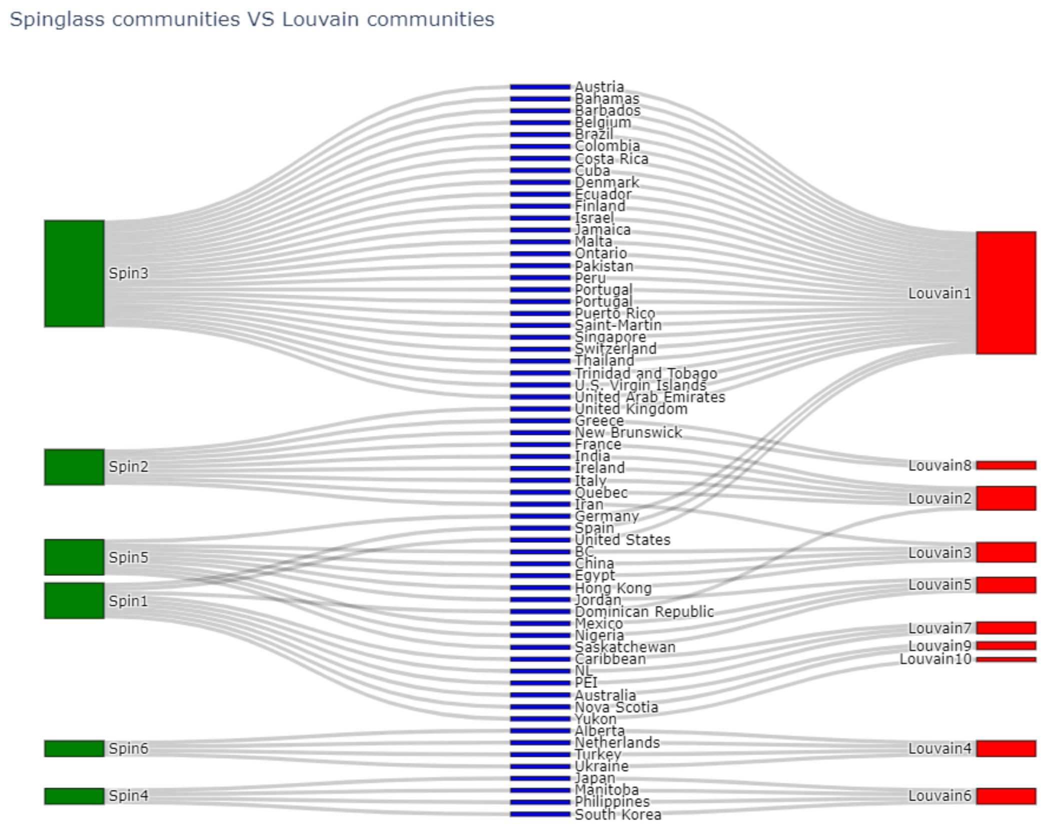


Figure 4.16: Comparing communities of Louvain and Label propagation algorithms

countries. ERGM models show that mutuality, in-degree, and out-degree have a significant positive effect on the probability of a tie in the world network, while the number of deaths have negative effects.

The analysis shows that, United States has the highest out-degree impact on Canada. This could happen as they are neighboring countries. The states, Ontario, British Colombia (BC), and Quebec have highly affected by other countries. According to the ERGM models, in-degree and out-degree, have significantly positive effect on the probability of a tie in the network of Canada.

Community detection in the spread of coronavirus across the world shows significant communities around Italy, China, and Iran which are the countries with the highest out-degree. But for the Canada data set, the resulted communities are not significant and clear. But in the results of Louvain and Spinglass, there is a significant community around Ontario, which has the highest in-degree. It is clear that all the community detection results depend on the algorithms.

We assessed the community detection algorithm performance using adjusted rand score (Rand, 1971) and normalized mutual information score (Zhang, 2015) which are given in Table 4.5 and 4.7. We noticed a fair agreement between algorithms performances in the world network than in the Canada network. More specifically, there is a good agreement between Louvain and Label propagation algorithms in community detection of world network, while there is a good agreement between Louvain and Spinglass algorithms in community detection of Canada network.

Chapter 5

Simulation Studies of Networks

In the case study, which we explained in Chapter 4, we could identify communities using different algorithms. But the challenge was determining the best method for community detection as we had no information about actual communities of countries. Hence in this simulation study, we focus on developing a process to identify the best community detection result for a network with no prior knowledge about the actual label.

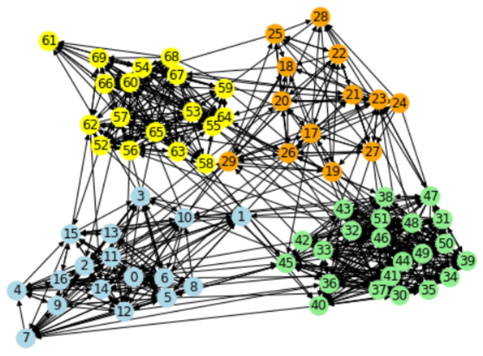
For that, first of all, we need to create networks with inbuilt partitions. We could use the Gaussian random partition graph or mixture based Gaussian Random Partition Graph developed in section 2.5.2. to develop networks. We will compare those two methods in this Chapter. There are multiple parameters we should set to generate networks (as discussed in Chapter 2). Therefore we consider various parameter configurations in this simulation study in order to understand the hidden truth when changing these parameters and the effect of these parameters on community detection.

The newly developed method to identify the best community detection results using two features is explained in section 5.4. We use Infomap, Louvain, Spinglass, and Label propagation as the community detection algorithms and use ARI, NMI, and AMI to identify similarities.

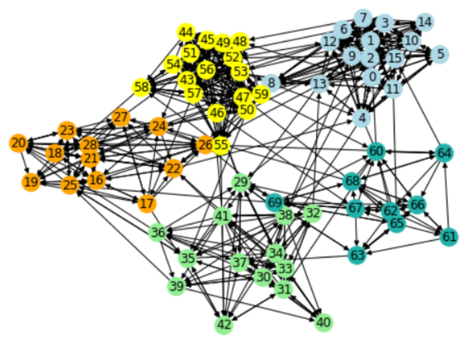
5.1 Graphical Comparison of Two Network Generation Methods

As we discussed in Chapter 3, for this study, we have considered two main random network graph generators. The first method is the Gaussian random partition graph, and it generates partitioned graphs, each partition with a size drawn from a normal distribution. Hence each of the communities will have a similar number of nodes. Sometimes the last partition can have a considerably smaller number of nodes, containing all the remaining nodes after grouping.

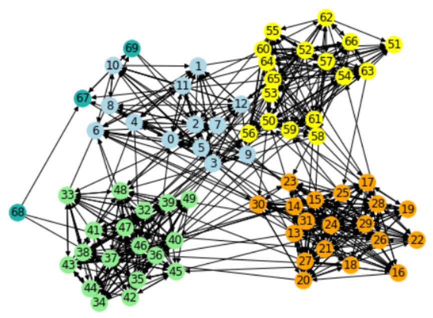
Figure 5.1 shows four random network graphs which were generated from Gaussian random network graph. To generate these networks we used same parameters (number of nodes = 70, mean cluster size = 15, shape parameter = 5, intra-probability = 0.5, inter-probability = 0.02), and ran it for four times. Since 70/15 = 4.66, we can see that most of the networks have either four or five communities. In plot (c), we can clearly see a cluster with only three nodes, and it should be that last partition of this network, based on the theory we learned.



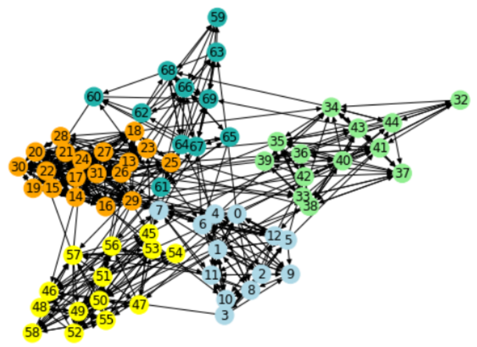
(a)



(b)



(c)



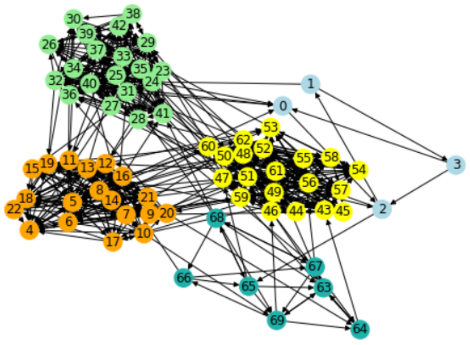
(d)

Figure 5.1: Plots (a) to (b): Some randomly generated graphs from Gaussian random partition graph. All four were generated using same parameters.

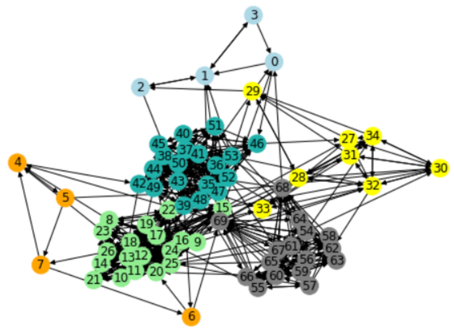
As we discussed at the beginning, based on Figure 5.1, we can see that almost all of the communities have a similar number of nodes. But in reality, we won't always be able to find this kind of community. There can be communities with a large number of nodes while some other communities with few nodes. Hence, to generate more practical networks, we developed our new network graph generator using the mixture of Gaussians.

In the new method, we have used a mixture of two normal distributions to pull cluster size. For example, in Figure 5.2 we can see network graphs that were generated from our new method. The number of nodes is also equal to 70, inter probability is 0.5, and inter probability is 0.02. For the Gaussian mixture, we used two normal distributions with mean 20 and 5. Therefore, some communities should contain nodes around 20, and others have nodes around 5. Due to that reason, we can observe, in these networks, some communities with a large number of nodes and a small number of nodes.

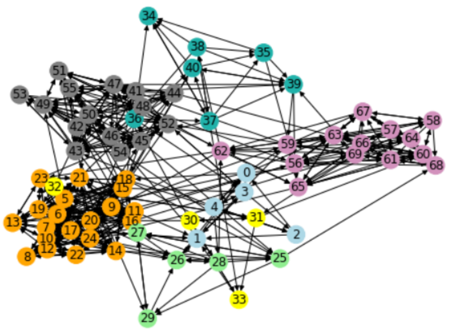
We can change inter and intra probabilities to very low values to have more realistic and sparse networks. The following Figure 5.4 shows a network with 70 nodes but very low inter and intra probabilities (intra-probability = 0.2, inter-probability = 0.01). Here we can see some nodes which are connected to the network by only a few edges. So based on different structures, the results of the community detection algorithm will vary. We can easily check that by changing these parameters.



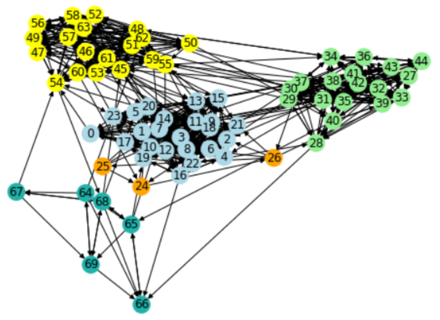
(a)



(b)



(c)



(d)

Figure 5.2: Plots (a) to (b): Some randomly generated graphs from newly developed method using mixture of two Gaussians. All four were generated using same parameters.

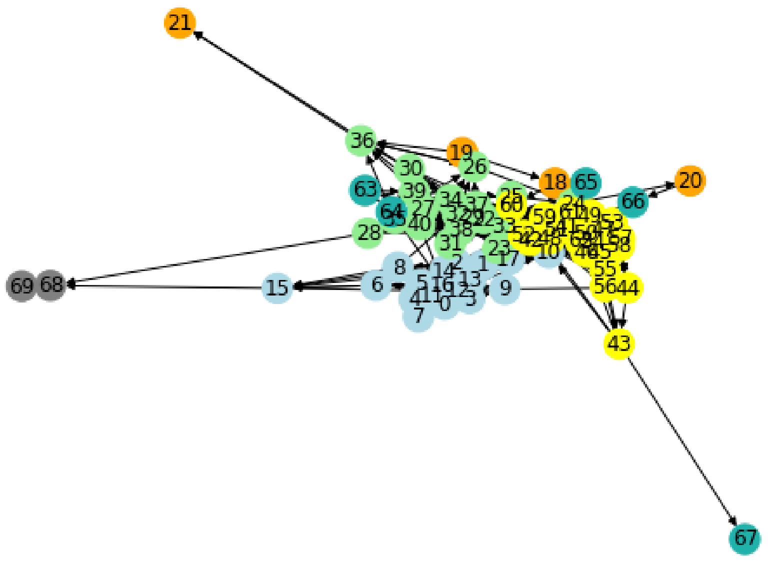


Figure 5.3: Example of a sparse network

5.2 Simulation Process

In this section, we describe our simulation process where we consider various parameter values. The following flow chart in Figure 5.4 shows the process step by step. That illustrates the simulation process when changing the number of nodes while all the other parameters are fixed. Based on this flow chart, first, we are generating a network with the number of nodes equal to 10. This network will be developed with community labels for each node, and we consider those labels as actual labels. Then we use the same generated network to find communities by applying our four community detection algorithms (Louvain, Label propagation, Spin glass, and Infomap). After that, we can evaluate community results based on actual labels using similarity matrices (ARI, NMI, and AMI).

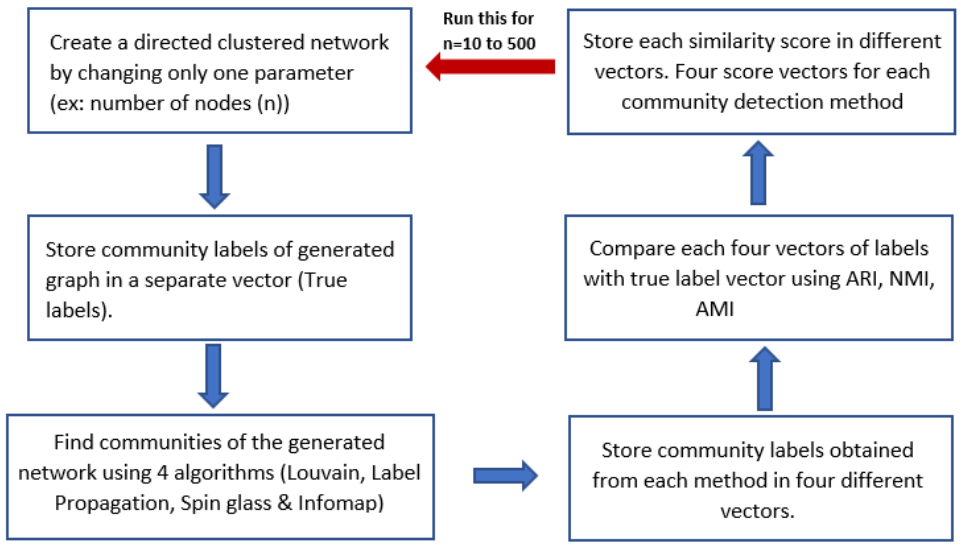

Create a directed clustered network by changing only one parameter (ex: number of nodes (n))
Run this for n=10 to 500
Store each similarity score in different vectors. Four score vectors for each community detection method
Store community labels of generated graph in a separate vector (True labels).
Compare each four vectors of labels with true label vector using ARI, NMI, AMI
Find communities of the generated network using 4 algorithms (Louvain, Label Propagation, Spin glass & Infomap)
Store community labels obtained from each method in four different vectors.

Figure 5.4: General simulation process

For each community detection method, we have three different similarity scores. We store each value in different vectors, and once we repeat the same process while changing sample size from n=10 to 500, those vectors should have 490 different scores for each. That will describe the level of accuracy of community detection when increasing the number of nodes in networks. It will help us find community detection algorithms that work well for a fewer number of nodes and a higher number of nodes.

If we describe the process with network plots, Figure 5.5 illustrates a network that we generated using the Gaussian random partition graph generator. It shows different communities with different colors. Then plots in Figure 5.6 show community results which we got by applying four community detection methods on the same network which we generated. In this simulation study we generated all the synthetic networks using 'Networkx' package in python (Hagberg et al., 2008).

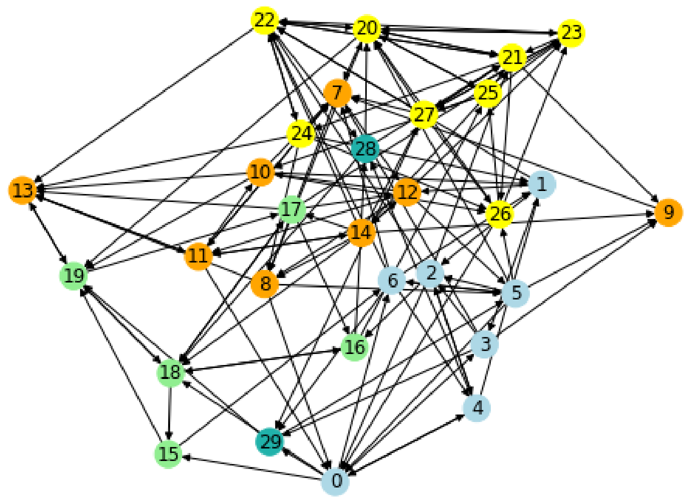
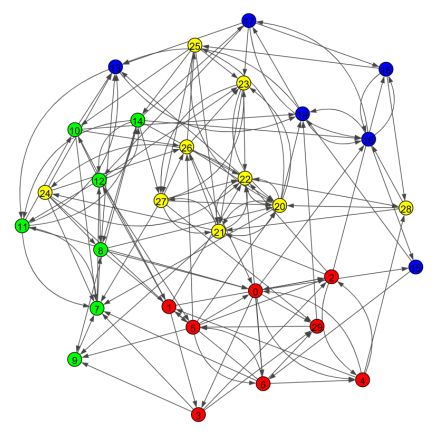
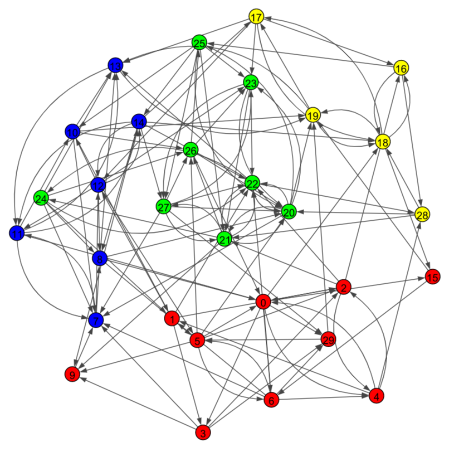


Figure 5.5: Generated network from Gaussian Random Partition Graph

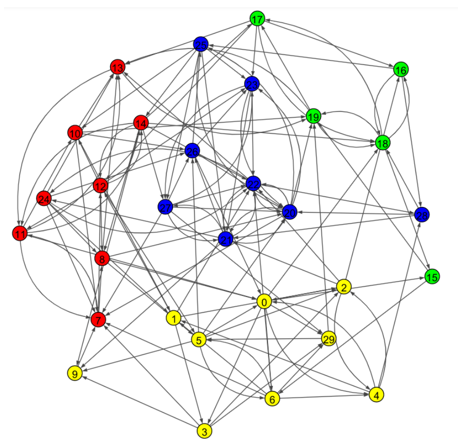
By exploring these figures, it is clear that there are differences and similarities between observed communities and true communities. These similarities were calculated using similarity measures, and it shows which community detection method has the highest similarity. Once we calculated the similarity scores for the network communities in Figure 5.6, it showed that the Infomap method has the highest similarity score to the true communities. Still, to understand the unknown truth, we needed to generate multiple networks and check the similarity score. That is why we used a simulation study to create



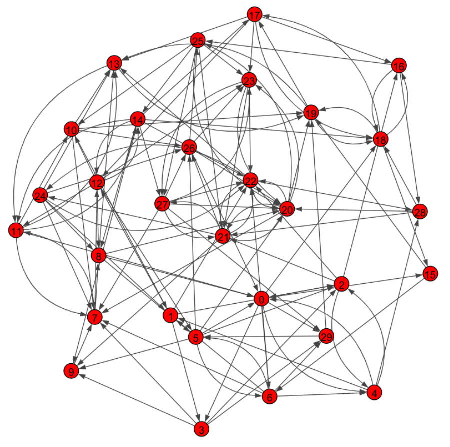
(a)



(b)



(c)



(d)

Figure 5.6: Community detection results using (a): Infomap, (b): Louvain. (c): Spinglass, (d): Label propagation. Each colour indicates a different community.

various networks by changing different parameters. To calculate similarity scores we used scikit-learn package in python (Pedregosa et al., 2011).

5.2.1 Changing Number of Nodes

In this simulation, when creating networks, we fixed all other parameters (mean cluster size, shape parameter, probabilities of inter and intra cluster connection). We changed the number of nodes from 50 to 300. These networks have inbuilt partitions, and those are considered as true community labels. Then each network was clustered using community detection algorithms and compared true labels with generated labels using similarity matrices. The change of similarity scores based on each community detection algorithm results are shown in Figure 5.7 to Figure 5.9.

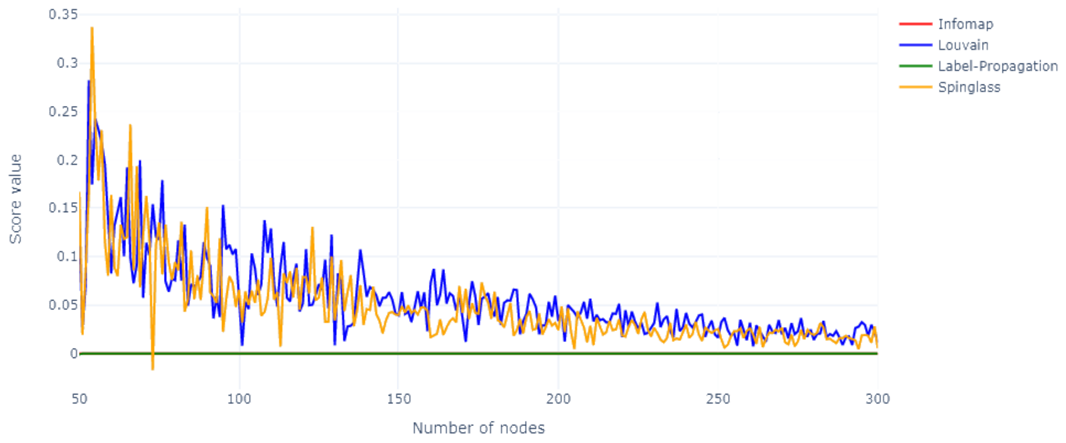


Figure 5.7: ARI similarity score variation over number of nodes. Each line shows similarity scores of different community detection algorithms

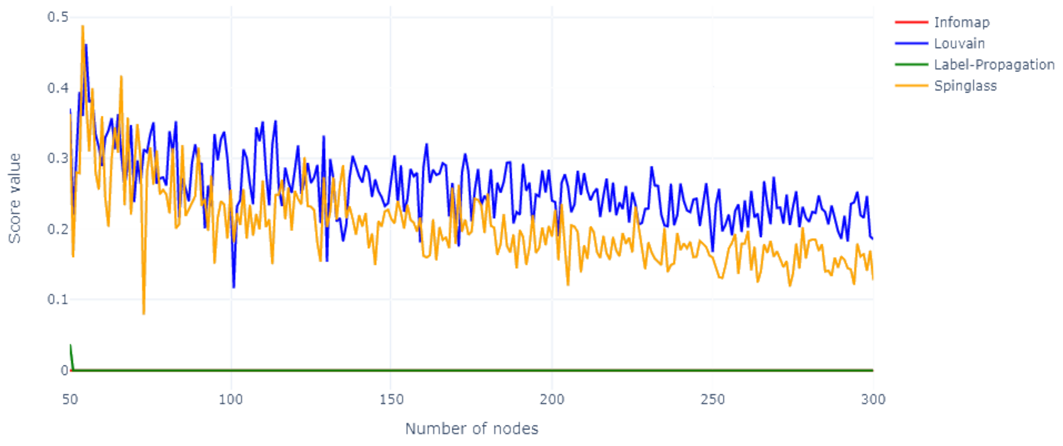


Figure 5.8: NMI similarity score variation over number of nodes

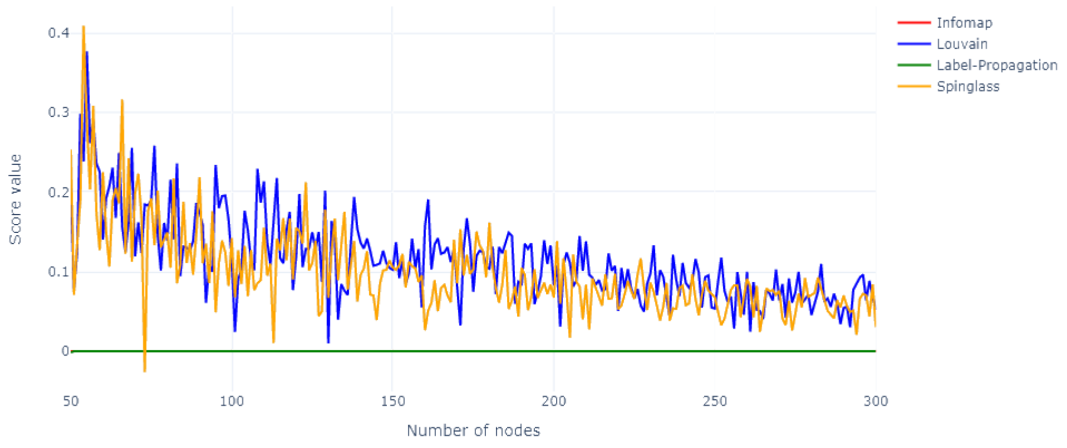


Figure 5.9: AMI similarity score variation over number of nodes

Based on these plots, we can see that ARI, NMI, and AMI scores of communities based on Louvain and Spinglass methods decrease when the number of nodes increases. ARI, NMI, and AMI scores for Infomap and Label-propagation methods are always constant with zero value. It shows that the number of nodes does not affect the similarity score of those two methods.

5.2.2 Changing Mean Cluster Size

In this simulation, when creating networks, we fixed all other parameters and changed the mean cluster size from 3 to 50. Mean cluster size is the average number of nodes within a cluster, and it will be considered when creating networks from the Gaussian random partition method. The change of similarity scores based on each community detection algorithm results are shown in Figure 5.10 to Figure 5.12.

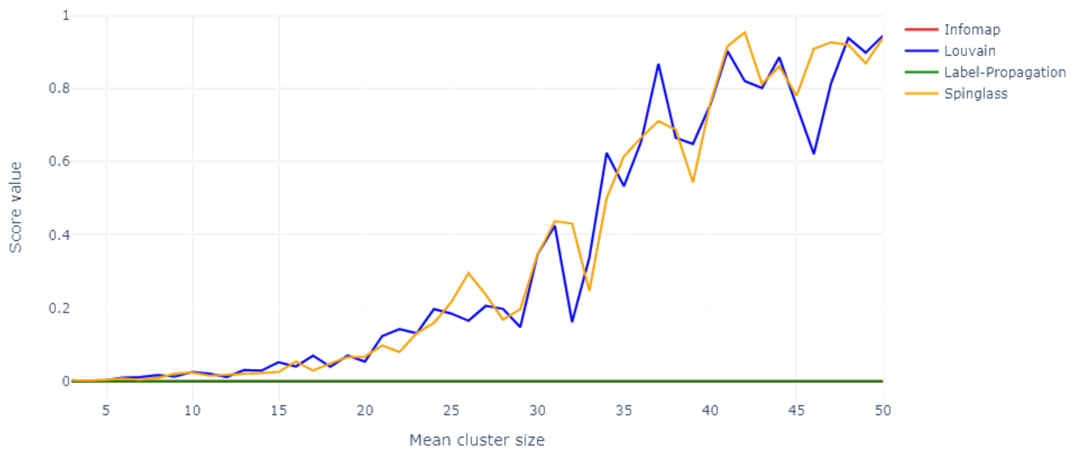


Figure 5.10: ARI similarity score variation over mean cluster size

According to the above plots, it is illustrated that as the mean cluster size is increasing, all similarity scores are also increasing for the community

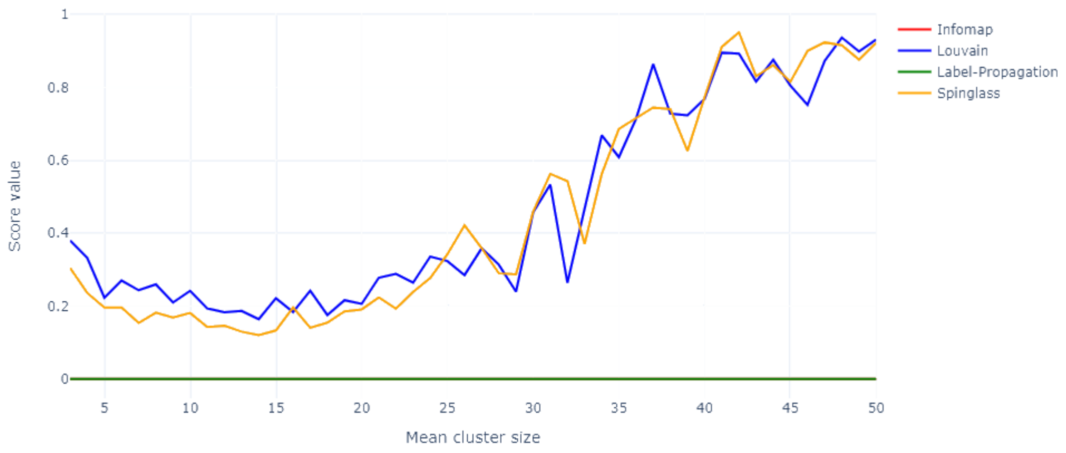


Figure 5.11: NMI similarity score variation over mean cluster size

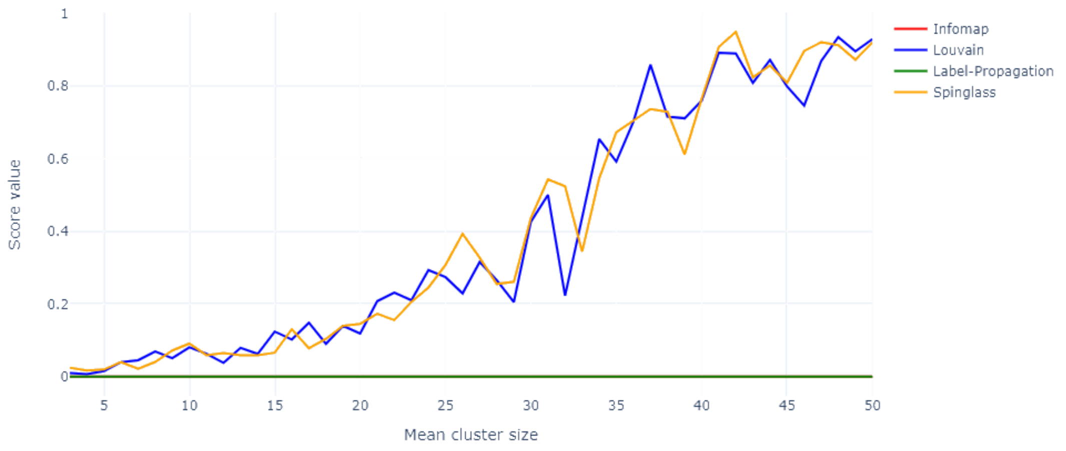


Figure 5.12: AMI similarity score variation over mean cluster size

detection results of Louvain and Spinglass methods. When the mean cluster size is equal to 50, similarity scores are almost equal to 1. Similarity scores equal to one means a perfect match with actual labels. ARI, NMI, and AMI scores for Infomap and Label-propagation methods remain in value zero.

5.2.3 Changing Probability of Connecting Nodes Within a Group

The probability of connecting nodes within a cluster is a parameter that we have to provide to create a random network. Then the edges which connect nodes in the same community of the network will be made based on that given probability. In this section, we have changed that probability from 0.1 to 0.9 with a 0.01 increase. This probability value should be within 0 to 1. While changing the probability, we fixed other parameters as constants. Then for each network, we identified the communities based on four different algorithms. Finally, the change of similarity score on each community detection algorithm results while changing the intra probability was plotted, and Figure 5.13 to Figure 5.15 illustrate those plots.

Based on these figures, it is clear that communities found from Louvain and Spinglass methods show better agreement with the actual labels for the networks which are created with an intra probability greater than 0.3. For Infomap and Label propagation communities, it is hard to find a specific intra probability value that shows a higher similarity score with the actual labels. But those methods also work well when the intra probability is more elevated.

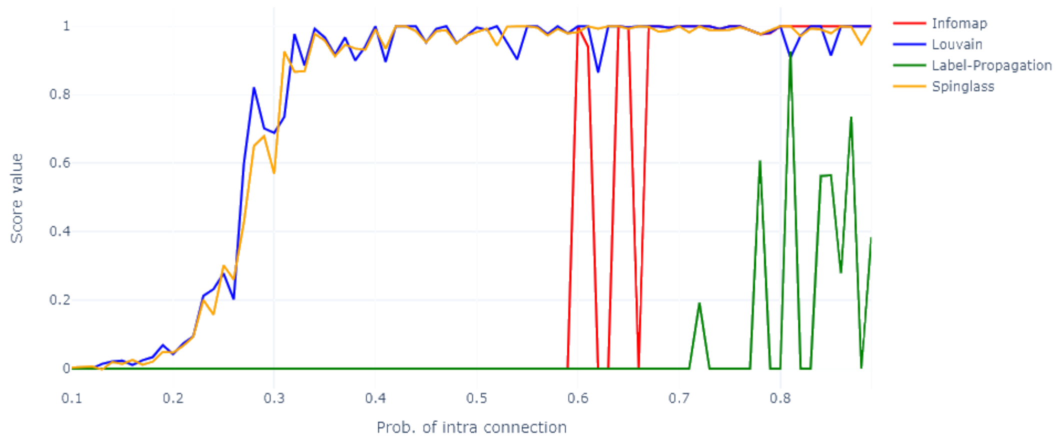


Figure 5.13: ARI similarity score variation over probability of intra connection

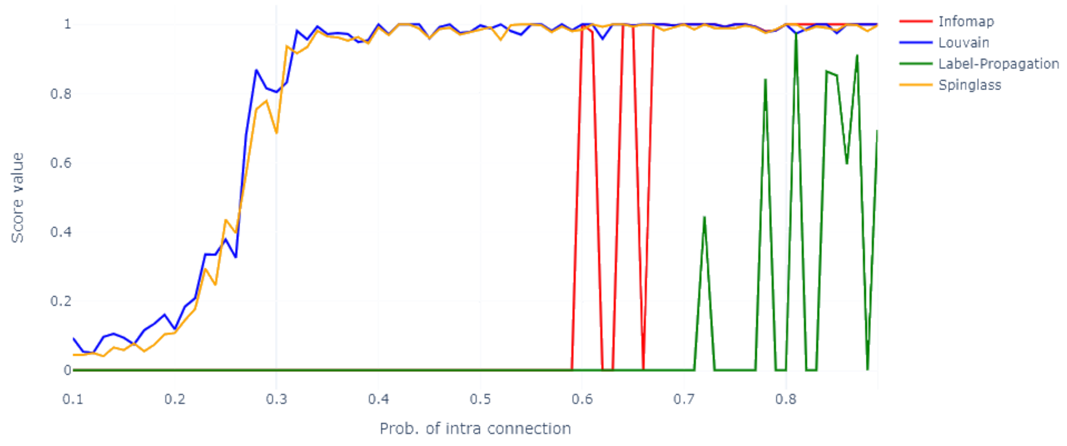


Figure 5.14: NMI similarity score variation over probability of intra connection

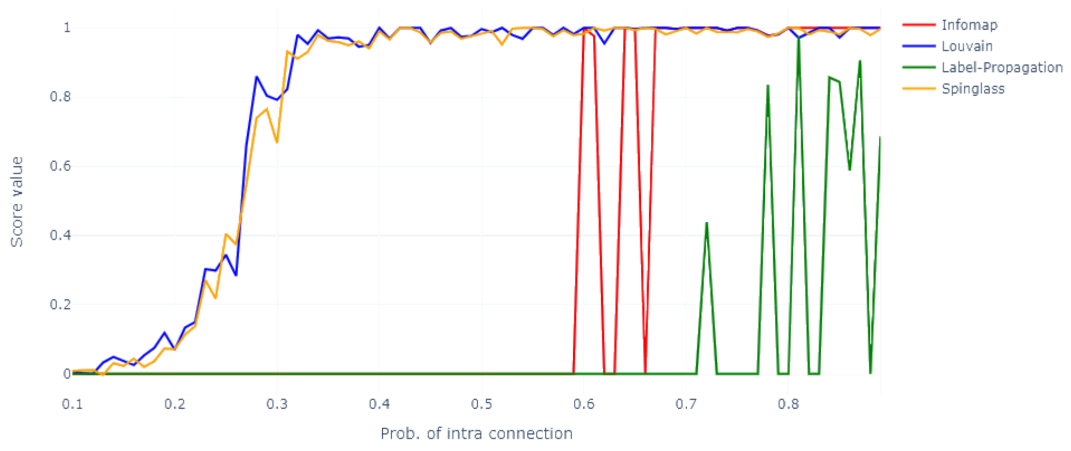


Figure 5.15: AMI similarity score variation over probability of intra connection

5.2.4 Changing Probability of Connecting Nodes Between Groups

In this section, we created different networks by changing the inter-cluster probability. Inter-cluster probability means the probability of connecting nodes between groups. That helped us figure out the effect of a network's inter-cluster probability to identify correct clusters. Same as the intra-cluster probability, we again used probabilities from 0.1 to 0.9 with 0.01 increment. Communities of the networks are identified using the four algorithms discussed earlier and calculated the similarity scores for each network. Figure 5.16 to 5.18 show the change of similarity scores based on each community detection algorithm results.

All the above similarity score plots illustrate that Louvain and spinglass

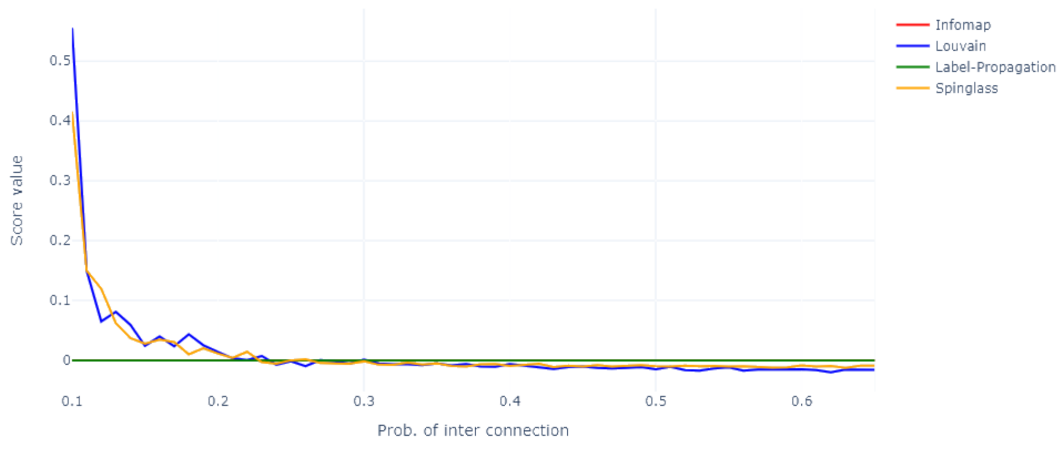


Figure 5.16: ARI similarity score variation over probability of inter connection

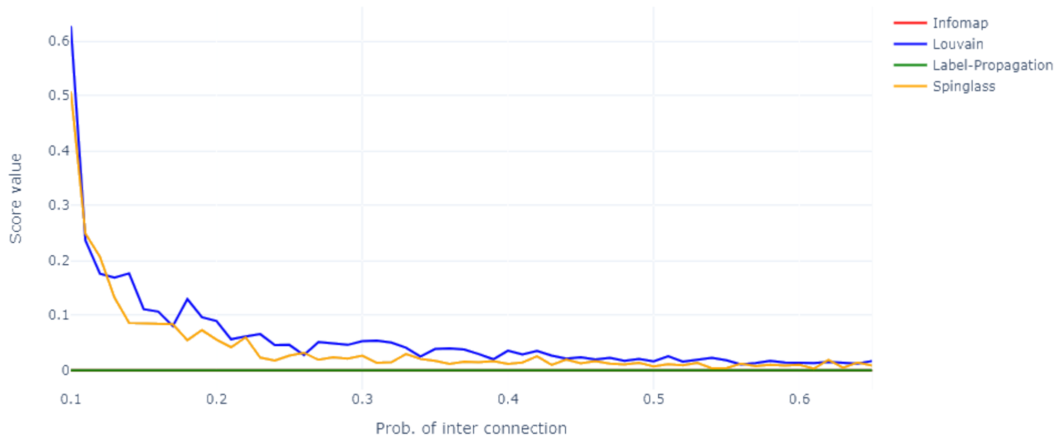


Figure 5.17: NMI similarity score variation over probability of inter connection

algorithms have higher similarity scores with true clusters when the networks have lower inter-connection probability. The highest similarity score is achieved when the inter probability is 0.1. After that score drastically drops down and

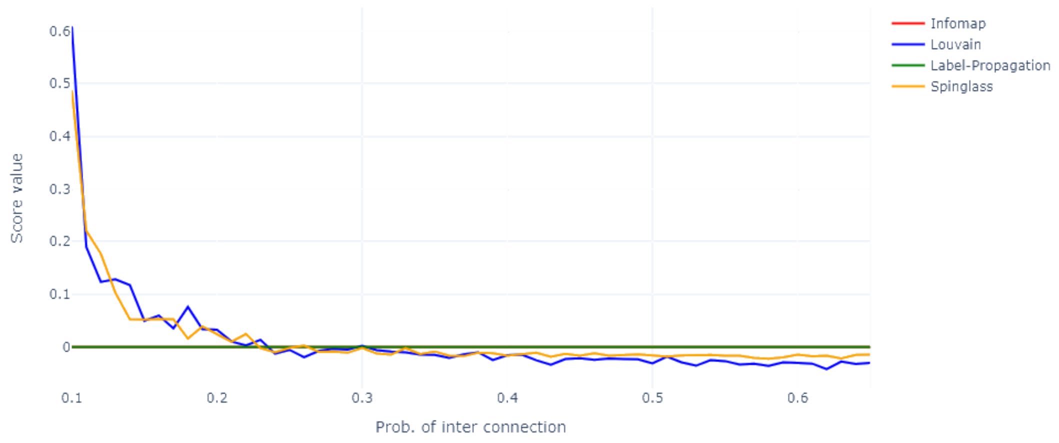


Figure 5.18: AMI similarity score variation over probability of inter connection

remains around zero. The results of Infomap and label propagation algorithms always show zero similarity scores for all three similarity scores.

5.3 Simulation Results for a Sparse Network

In the previous section 5.2 we explained the simulation process and discussed the results using the networks which are generated from the Gaussian random partition graph. Those networks used in that section have a node with many connected links (dense networks). But in real life, we don't always get dense networks. For example, in our COVID-19 case study, not all the countries in those networks had many connections with others. Those types of networks are called as sparse networks. Sparse networks have much fewer edges than the possible maximum number of edges.

In this section, we created sparse networks using our new network generation method, using a mixture of Gaussians. We conducted the same simulation process to identify the differences when using dense networks and sparse networks. To have a sparse network, we used much lower values for intra probability and inter probability. Figure 5.19 shows the network we generated from the new method when n = 30, 1^{st} mean = 3, 2^{nd} mean = 9, intra-probability = 0.3, inter-probability = 0.02 and both standard deviance are equal to 1. In this network, some nodes connect with this network by only one edge. That has a similar network structure to the networks that we generated in the case study.

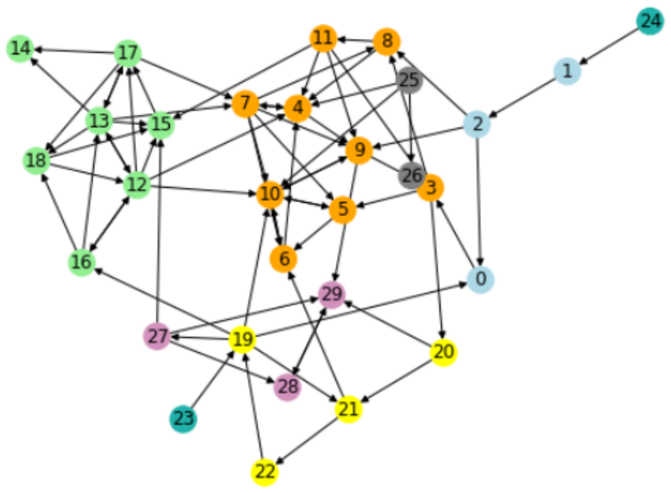


Figure 5.19: Generated network from newly developed Graph Partition method

The results that we got from community detection algorithms are shown in Figure 5.20. In the previous section, we saw that the label propagation algorithm fail to detect multiple communities in the dense network. But once we apply label propagation to this sparse network, it could detect at least a few communities correctly. All the other methods show some acceptable results. But to understand this correctly, we conducted the same simulation study by changing the parameters. But here, we did not consider changing the mean cluster size as we use the mixture of Gaussian, and it has two different mean values.

5.3.1 Changing Number of Nodes

Similar to the simulation in section 5.2, we fixed all other parameters while changing the number of nodes from 30 to 300. The behavior of the similarity scores; ARI, NMI, and AMI for the results of each community detection algorithms, when increasing the number of nodes, illustrate in Figure 5.21, Figure 5.22 and Figure 5.23 respectively. When generating random sparse networks, some networks end up with isolated nodes. Spinglass network does not work on networks that have isolated nodes. Hence for the simulation study of sparse networks, we did not use the Spinglass method.

In section 5.2.1, the results of both Label propagation and Infomap algorithms showed zero scores for all the similarity scores when increasing the number of nodes. But based on the results show in Figure 5.21 to Figure 5.23 it is clear that in the sparse networks when the sample size is small, the results of all three algorithms show a higher similarity with true labels. Label

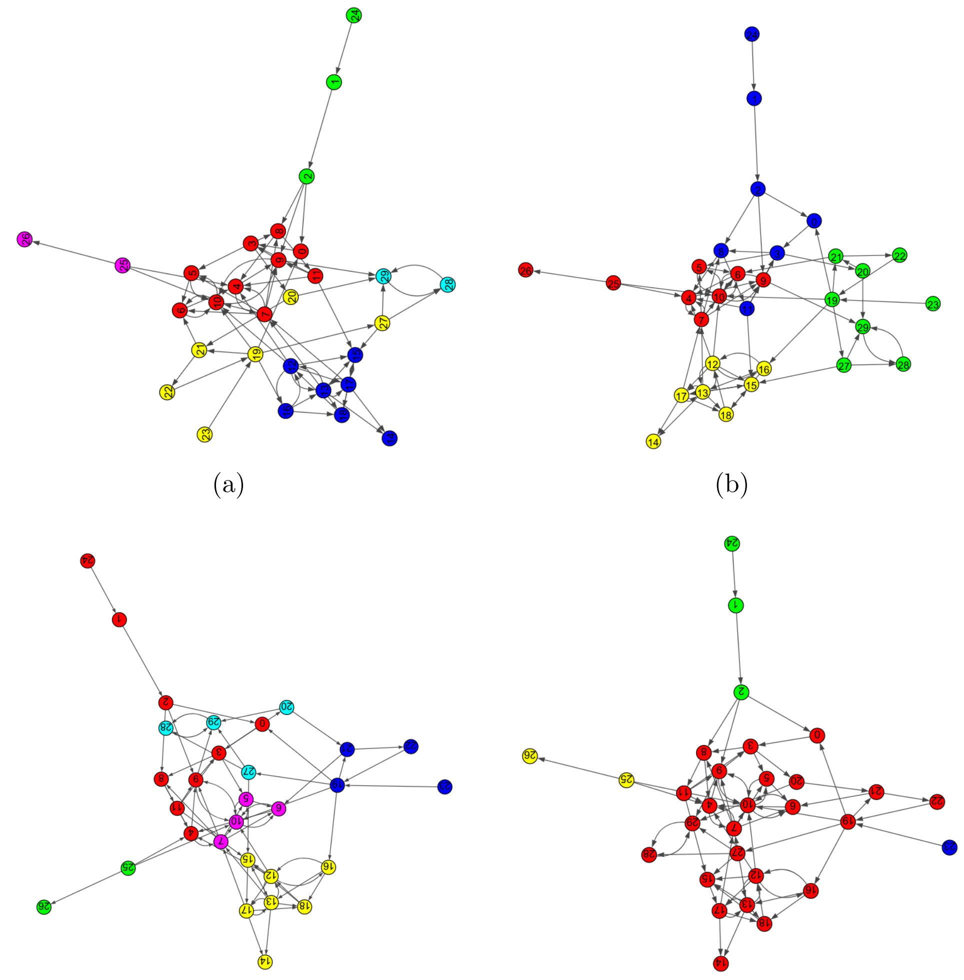


Figure 5.20: Community detection results of the sparse network using (a): Infomap, (b): Louvain. (c): Spinglass, (d): Label propagation. Each colour indicates a different community.

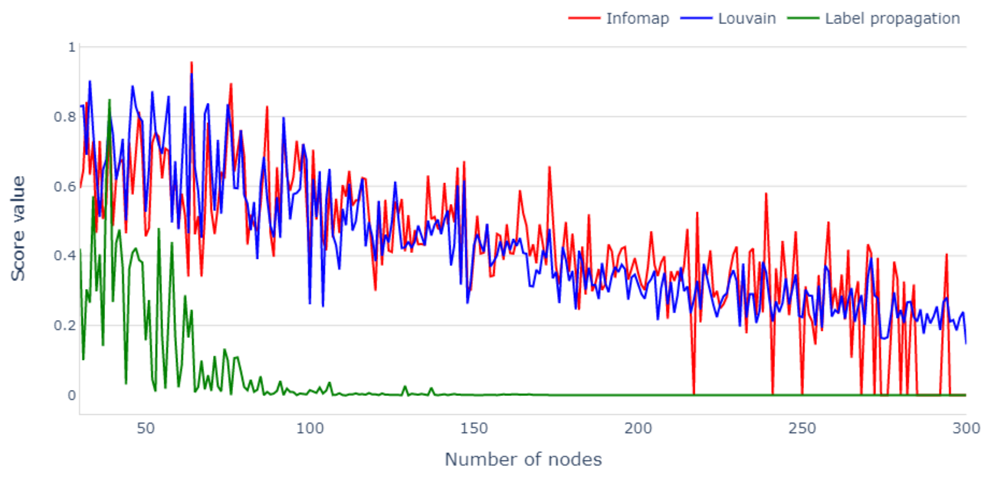


Figure 5.21: ARI similarity score variation with the change of number of nodes

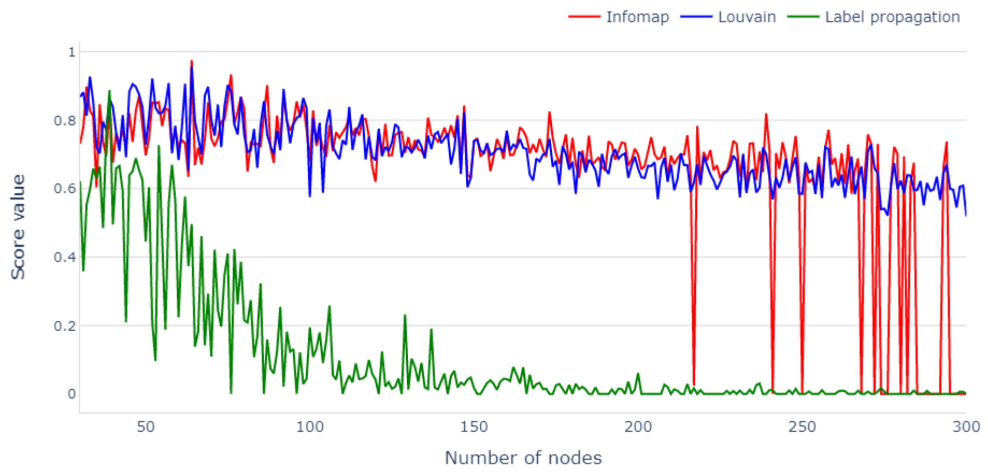


Figure 5.22: NMI similarity score variation with the change of number of nodes

propagation shows a rapid decline, while Infomap and Louvain show a gradual decline with the increase of frequency of nodes.

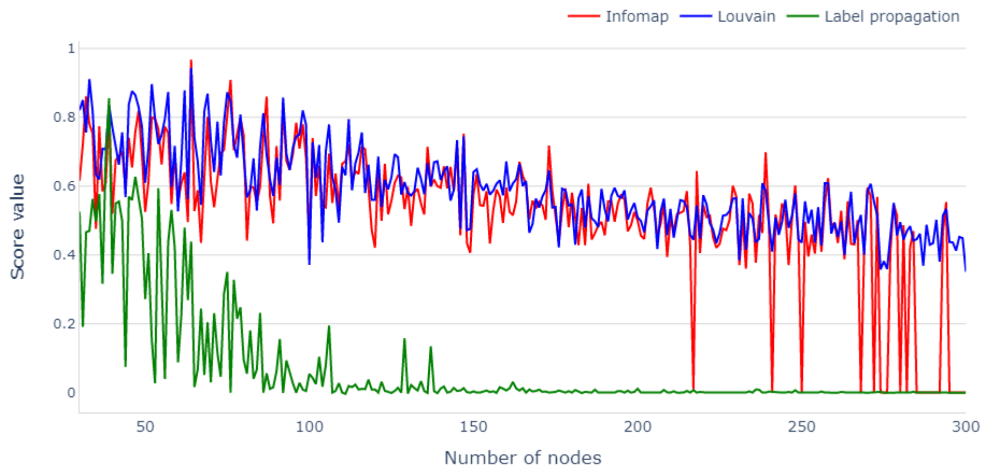


Figure 5.23: AMI similarity score variation with the change of number of nodes

5.3.2 Changing Probability of Connecting Nodes Within a Group

We did this simulation based on section 5.2.3, and intra probability was changed from 0.1 to 0.9 with a 0.01 increment. We fixed inter probability to 0.02 and the number of nodes to 50. Figures from 5.24 to 5.26 shows the results of this simulation.

Figures in section 5.2.3 show that the results of Infomap and Label propagation have zero similarity score till the intra probability reaches 0.6. But in this sparse network, all three algorithms show a gradual increase of similarity score with the rise of intra probability. Here the results of Louvain and Infomap show better and similar results than the Label propagation algorithm.

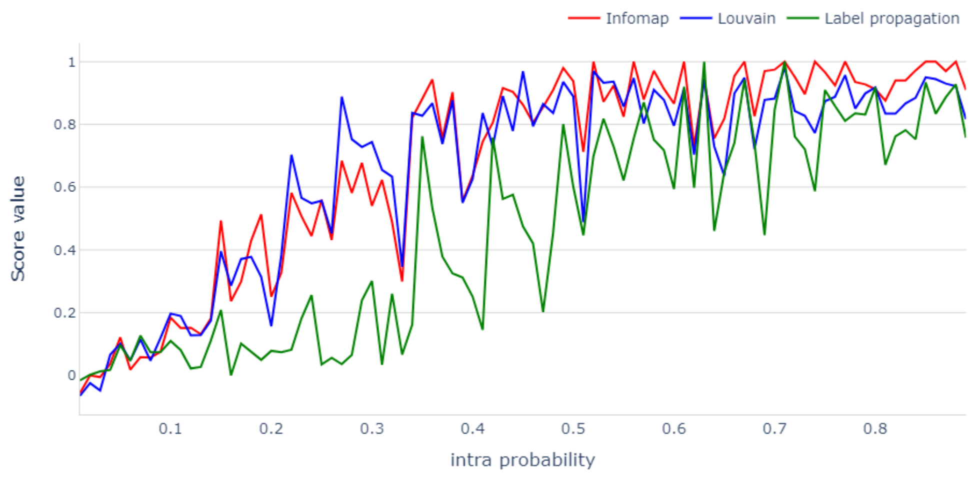


Figure 5.24: ARI similarity score variation with the change of intra probability

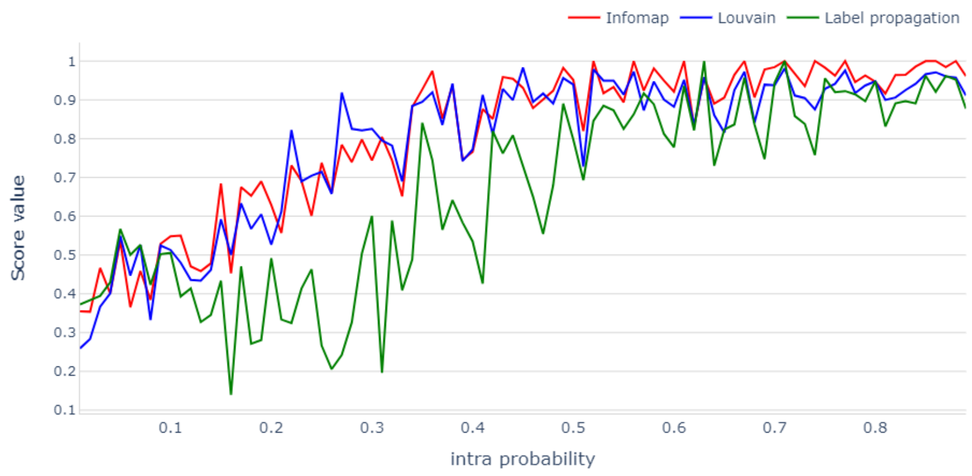


Figure 5.25: NMI similarity score variation with the change of intra probability

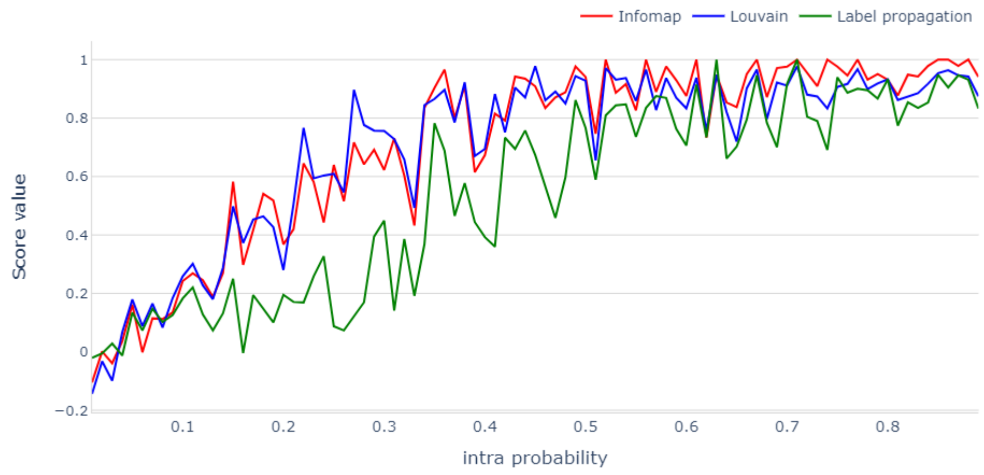


Figure 5.26: AMI similarity score variation with the change of intra probability

5.3.3 Changing Probability of Connecting Nodes Between Groups

In this simulation, we changed the probability of connecting nodes between groups (inter probability) from 0.01 to 0.9 with a 0.01 increment. We fixed the intra probability to 0.3 and the number of nodes to 50. The variation of similarity scores (ARI, NMI, and AMI) for the results of each community detection algorithm was plotted in the Figures from 5.27 to 5.29.

When the inter probability is lower than 0.1, we can see higher similarity scores for all the community detection algorithms.

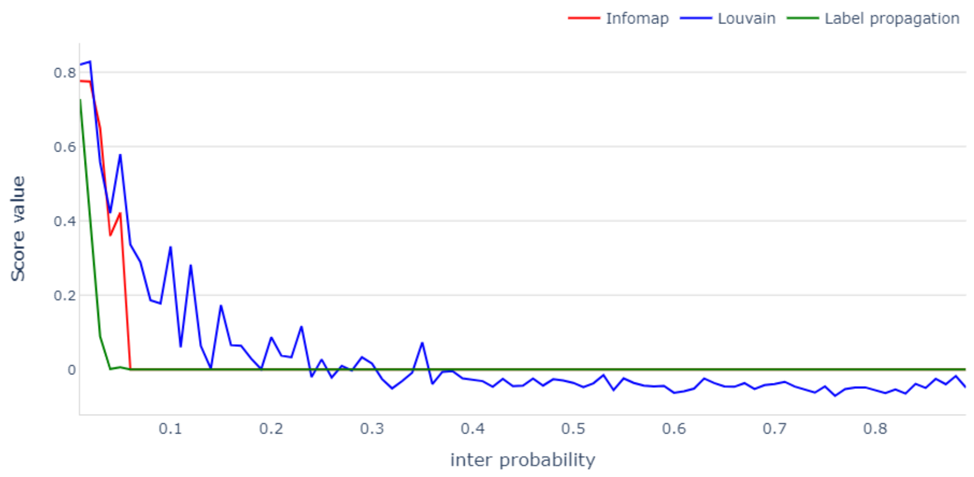


Figure 5.27: ARI similarity score variation with the change of inter probability

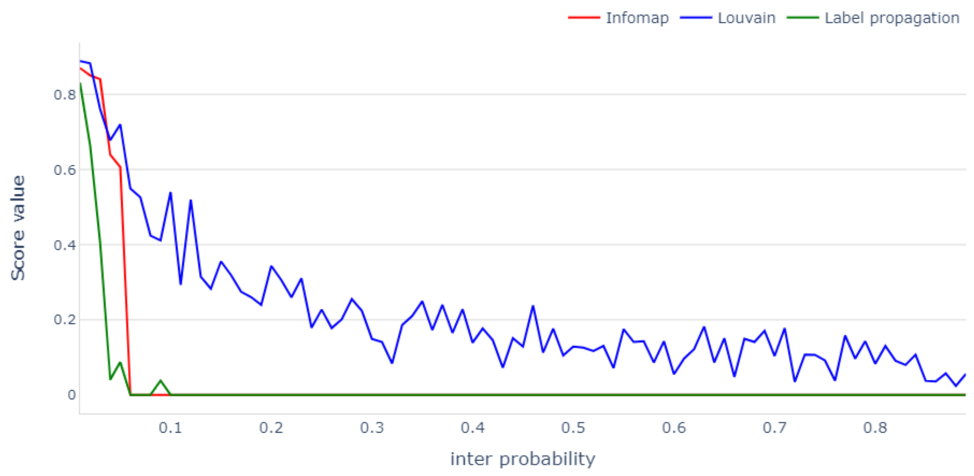


Figure 5.28: NMI similarity score variation with the change of inter probability

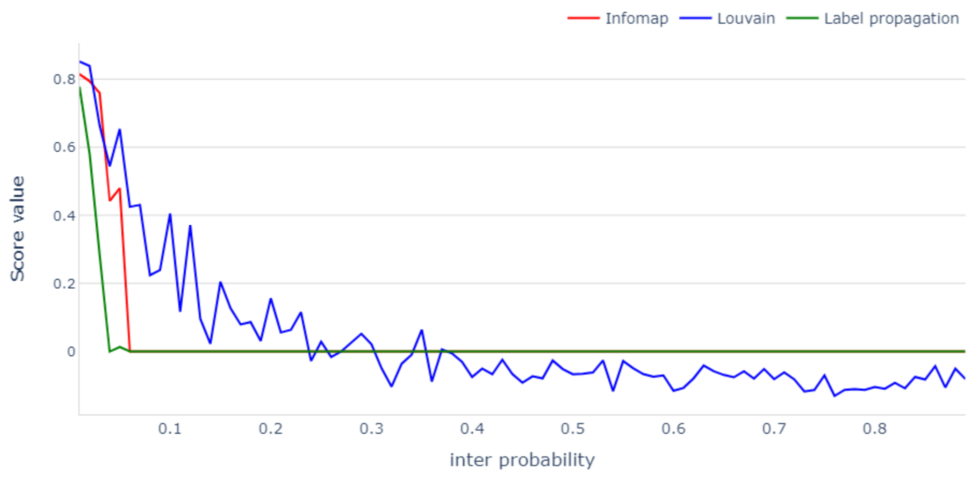


Figure 5.29: AMI similarity score variation with the change of inter probability

5.4 A Comparative Analysis of Community Detection Algorithms

In the section 5.2 and 5.3, we discussed how the agreement with actual labels changes with the change of different parameters which we use to generate networks. By changing these parameters, we tried to change the structure of networks and identify the effect of that to detect communities by different community detection algorithms. But in those sections, we changed one parameter at a time. In this section, we explain a procedure to consider two parameters and understand the hidden truth of the results of different community detection algorithms.

5.4.1 Matrix Creation Process

This process explains the steps to create a matrix for each community detection algorithm, which shows the values for the given similarity score measure, with the change of intra probability (P_{in}) and inter probability (P_{out}). Here rows and columns will indicate P_{in} and P_{out}. Through this matrix we can identify a range of network attributes (P_{in} and P_{out}) of the community network that can identify communities more accurately.

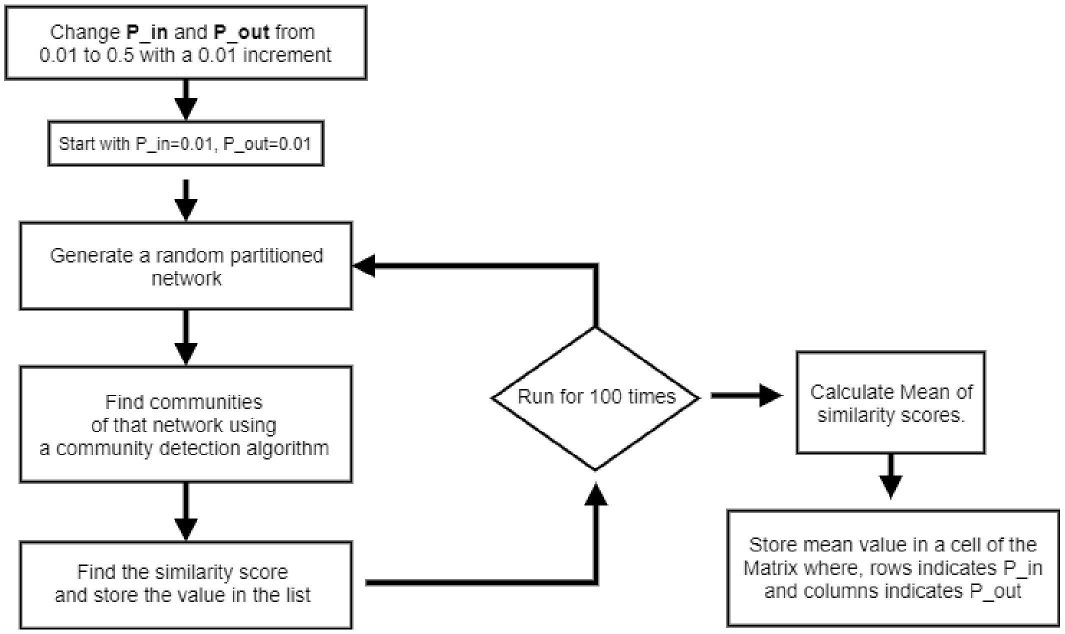

Change **P_in** and **P_out** from 0.01 to 0.5 with a 0.01 increment
Start with P_in=0.01, P_out=0.01
Generate a random partitioned network
Find communities of that network using a community detection algorithm
Find the similarity score and store the value in the list
Run for 100 times
Calculate Mean of similarity scores.
Store mean value in a cell of the Matrix where, rows indicates P_in and columns indicates P_out

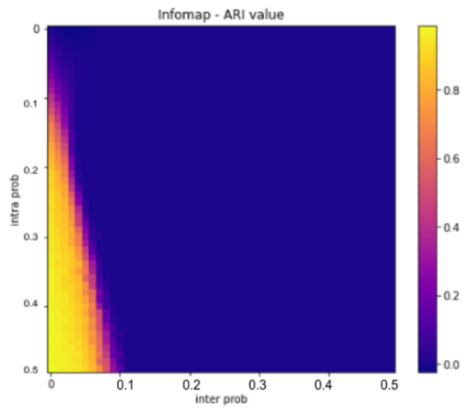
Figure 5.30: The process of creating a similarity score value matrix

Figure 5.30 explains the process step by step. As the first step, we fixed number of nodes (n), two mean values (s1, s2), two standard deviance (σ_1, σ_2) values and changed P_{in} and P_{out} values from 0.01 to 0.5 with 0.01 increment. Here we increased these probability values to 0.5 because we needed to generate sparse networks. So the first value that we allocated to both P_{in} and P_{out} is 0.01. Second, we generated a network using a random partition graph with a mixture of two Gaussians and considered those partitions as true communities. Next, we applied the community detection algorithm (Louvain, Infomap, or Label Propagation) to the network and identified the communities. True community labels and identified community labels were used to find similarity scores using the given measure (ARI, NMI, or AMI). Then we stored the similarity score in a vector. We repeated the same procedure 100 times to generate 100 random networks and filled the vector with 100 similarity scores. We then calculated the mean of those similarity values to get the average similarity score for the given P_{in} and P_{out}. This average similarity score will be stored in the first cell of the matrix. Then repeat the same procedure for different P_{in} and P_{out} values and fill the matrix.

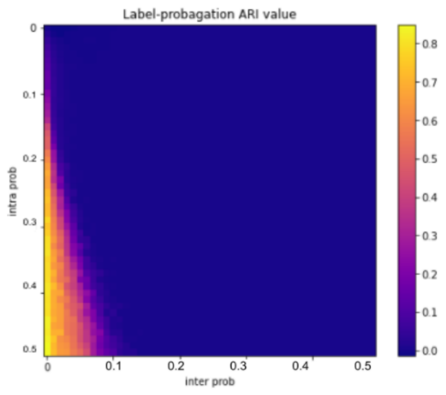
After creating the similarity score matrix, we could easily convert it to a color heat map. Heat maps appeal to the eyes, and visualization is generally easier to understand than reading the values. Through the heat maps, we could easily identify the capacity of the given algorithm to identify correct communities.

5.4.2 Results of Similarity Matrices

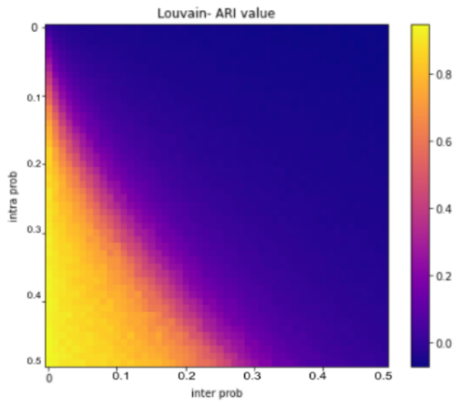
According to the steps we described in section 5.4.1 we generated matrices for each community detection algorithm. Since we are considering three similarity measures, ARI, NMI, and AMI, we should get three matrices for each algorithm. Hence all together, we generated nine matrices to compare the community detection algorithms.



(a)



(b)

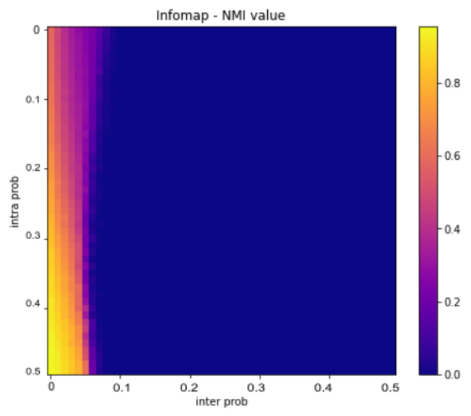


(c)

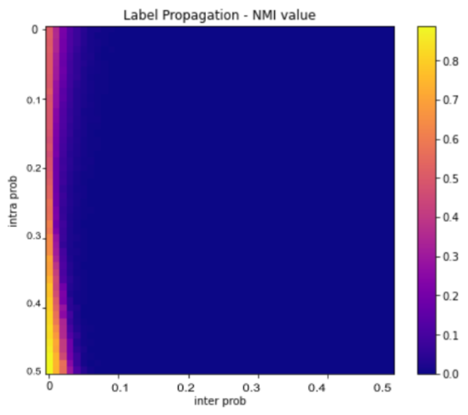
Figure 5.31: ARI similarity score heat maps for (a): Infomap, (b): Label propagation (c): Louvain. From dark blue color to yellow color indicates low to high similarity score.

Figure 5.31 shows ARI similarity score heat maps for the results of Infomap, Label propagation, and Louvain algorithms. The color range from dark blue to yellow shows the lowest similarity score to the highest similarity score in these heat maps. In both section 5.2 and 5.3, we saw that all the community detection algorithms show higher similarity scores for the networks with lower inter probability and higher intra probability. Also, in these heat maps, we can see that left bottom corners, which have higher intra probabilities and lower inter probabilities, have an area in yellow color, which indicates a high similarity score. Hence, it is clear that the results of all the community detection algorithms are perfect if their community networks have work high intra probabilities and low inter probabilities. For example, let's consider the heatmap of the Infomap algorithm. Once we generate the community detection result for a network using the Infomap algorithm, we can calculate intra probability (P_{in}) values and inter probability (P_{out}) values for the resulted community network. Then if those values are within the yellow color area of the heat map, we can evaluate that the community results are perfect.

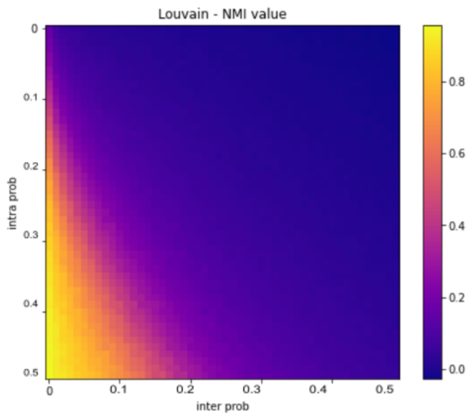
When comparing these three heat maps, we can observe that plot (b) has the smallest yellow color area, and plot (c) has the largest yellow color area. Plot (b) and (c) show the results of the Label propagation algorithm and Louvain algorithm. Throughout the simulation Chapter, we observed that the label propagation algorithm failed to detect true communities most of the time, and this heat map gives the reason for it. Based on plot (a), Infomap is also not always able to find the communities, but it is better than the Label propagation algorithm. Throughout the simulation study, Louvain gave a high similarity score for most of the networks. Based on plot (c), it is clear that this algorithm can identify communities of networks with various network structures.



(a)



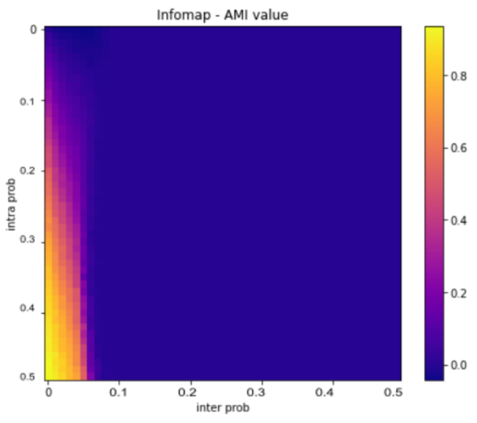
(b)



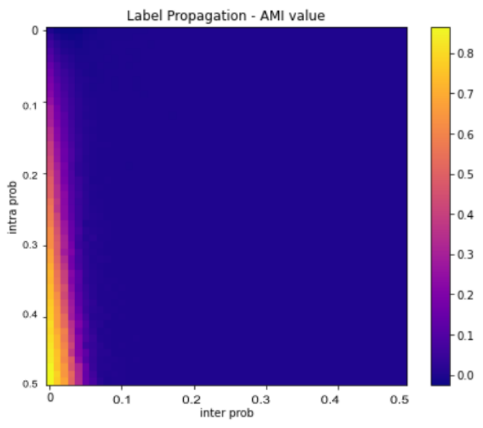
(c)

Figure 5.32: NMI similarity score heat maps for (a): Infomap, (b): Label propagation (c): Louvain. From dark blue color to yellow color indicates low to high similarity score.

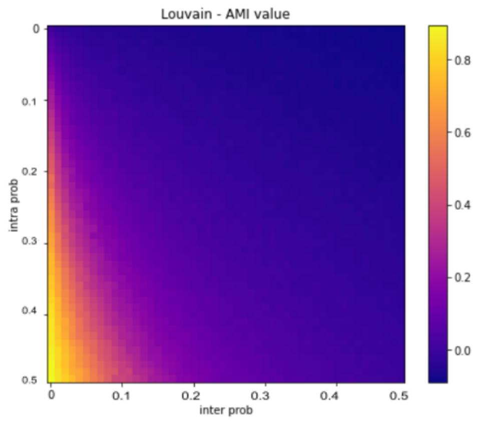
Figure 5.32 and 5.33 show NMI and AMI similarity score heat maps,



(a)



(b)



(c)

Figure 5.33: AMI similarity score heat maps for (a): Infomap, (b): Label propagation (c): Louvain. From dark blue color to yellow color indicates low to high similarity score.

respectively. We can see yellow color areas (high similarity scores) for higher intra probabilities and lower inter probabilities in those heat maps as well. These sets of heatmaps also show that the Louvain algorithm has the highest capacity to detect communities while Label propagation has the lowest capacity.

Chapter 6

Conclusion

Since the community detection is specially developed for social network analysis which depends on edges, it is clear that the results should depend on the network structure. Our simulation study showed that most of the community detection algorithms gave better results for sparse networks than dense networks. Even the community detection algorithms struggling to detect communities in dense networks could detect communities up to some level in sparse networks. In both sparse and dense networks, we observed that the ability to detect communities of algorithms increased with the increment of the probability of having edges between nodes in the same community and the increment of average community size. The detection ability decreased with the increment of the probability of having edges between nodes in different communities and the number of nodes.

Using the heat maps we generated, we could understand the community detection capacities of different community detection algorithms. Overall, the results of all the community detection methods are perfect if the community network has higher intra probabilities and lower inter probabilities. Still, the Louvain algorithm has the largest range of inter and intra probabilities, while label propagation has the smallest range. Throughout the study, we observed that the Louvain algorithm could detect communities better than other algorithms. Even though both Label propagation and Infomap failed to detect correct communities in dense networks, we could see a significant improvement of Infomap in sparse networks.

The Gaussian random partition graph generator is one of the best methods to generate random networks with partitions. It uses Gaussian distribution to draw the community size, and because of that, the output networks contain communities with a similar number of nodes. But in real networks, we see communities with a different number of nodes; large communities and small communities. Hence we developed this novel method to generate networks using a mixture of Gaussians and observed that it could generate networks similar to the structure of real-world disease transmission networks.

This study can be further expanded by updating the network generator using a Dirichlet process with Gaussian distribution as the base distribution. Then we can compare the network structures of the networks generated from the existing Gaussian random partition graph with the networks generated from the newly updated method with the Dirichlet process. Furthermore, we can improve our approach to evaluate temporal community detection in real world applications.

Bibliography

Akaike, H. (1973). Information theory and an extension of the maximum likelihood principle. In B. Petrov and F. Csaki (Eds.), *2nd International Symposium on Information Theory*, pp. 267–281. Budapest, Hungary: Akadémiai Kiadó. (Cited on page 13.)

Berry, I. and J. P. R. Soucy (2020a). ccodwg/covid19canada. (Cited on page 39.)

Berry, I. and J. P. R. Soucy (2020b). Technical report. (Cited on page 39.)

Berry, I., J. P. R. Soucy, et al. (2020). Open access epidemiologic data and an interactive dashboard to monitor the covid-19 outbreak in canada. *CMAJ 192*(15), E420–E420. (Cited on page 39.)

Blondel, V., J. L. Guillaume, et al. (2008). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics Theory and Experiment*, P10008. (Cited on page 27.)

Brandenberger, L. and S. Martínez (2019). Lab 2 and 3 - intro to ergms, marginal effects and goodness-of-fit. (Cited on page 14.)

Brandes, U., M. Gaertler, and D. Wagner (2003). Experiments on graph clustering algorithms. In U. Di Battista, Giuseppe Zwick (Ed.), *Algorithms - ESA 2003*, Volume 2832 of *Lecture Notes in Computer Science*, pp. 568–579. Springer,Berlin, Heidelberg. (Cited on pages 4, 7, 15 and 16.)

Burns, J., A. Movsisyan, et al. (2020). Travel-related control measures to contain the covid-19 pandemic: a rapid review. *Cochrane Database of Systematic Reviews*. (Cited on page 2.)

Chakrabort, D., A. Singh, and H. Cherifi (2016). Immunization Strategies Based on the Overlapping Nodes in Networks with Community Structure. In H. Nguyen and V. Snasel (Eds.), *Computational Social Networks. CSoNet 2016*, Volume 9795 of *Lecture Notes in Computer Science*, pp. 62–73. Springer, Cham. (Cited on page 25.)

Chinazzi, M., J. T. Davis, M. Ajelli, C. Gioannini, M. Litvinova, S. Merler, A. Pastore y Piontti, K. Mu, L. Rossi, K. Sun, C. Viboud, X. Xiong, H. Yu, M. E. Halloran, I. M. Longini, and A. Vespignani (2020). The effect of travel restrictions on the spread of the 2019 novel coronavirus (covid-19) outbreak. *Science 368*, 395–400. (Cited on page 2.)

Craft, M. E. (2015). Infectious disease transmission and contact networks in wildlife and livestock. *Philosophical transactions of the Royal Society of London. Series B, Biological sciences 370.* (Cited on page 1.)

Crosby, A. W. (2003). *America's Forgotten Pandemic: The Influenza of 1918* (2 ed.). Cambridge University Press. (Cited on page 38.)

Dugué, N. and A. Perez (2015). Directed Louvain : maximizing modularity in directed networks. Research report, Université d'Orléans. (Cited on page 26.)

Eames, K. T. D., N. L. Tilston, P. E. Brooks, and W. J. Edmunds (2012). Measured dynamic social contact patterns explain the spread of h1n1v influenza. *PLoS computational biology 8*, e1002425. (Cited on page 2.)

Fang, H. and Y. Liu (2015). A novel algorithm infomap-sa of detecting communities in complex networks. *Journal of Communications 10*, 503–511. (Cited on page 28.)

Goeyvaerts, N., E. Santermans, G. Potter, A. Torneri, K. K. Van, L. Willem, M. Aerts, P. Beutels, and N. Hens (2018). Household members do not contact each other at random: Implications for infectious disease modelling. *Proceedings of the Royal Society B: Biological Sciences 285.* (Cited on page 12.)

Groendyke, C., D. Welch, and D. R. Hunter (2012). A Network-based Analysis of the 1861 Hagelloch Measles Data. *Biometrics 68*, 755–765. (Cited on page 12.)

Hagberg, A., P. Swart, and D. S Chult (2008). Exploring network structure, dynamics, and function using NetworkX . In G. Varoquaux, T. Vaught, and J. Millman (Eds.), *Proceedings of the 7th Python in Science Conference*, Pasadena, CA USA, pp. 11–15. (Cited on page 64.)

Hunter, D. R., S. M. Goodreau, and M. S. Handcock (2008). Goodness of fit of social network models. *Journal of the American Statistical Association 103*, 248–258. (Cited on page 12.)

Jebabli, M., H. Cherifi, C. Cherifi, and A. Hamouda (2018). Community detection algorithm evaluation with ground-truth data. *Physica A: Statistical Mechanics and its Applications 492*, 651–706. (Cited on page 3.)

Kirkpatrick, S., C. D. Gelatt, and M. P. Vecchi (1983). Optimization by simulated annealing. *Science 220*, 671–680. (Cited on page 32.)

Lau, J. T., X. Yang, E. Pang, et al. (2005). Sars-related perceptions in hong kong. *Emerging infectious diseases 11*, 417–424. (Cited on page 38.)

Li, W., S. Jiang, and Q. Jin (2018). Overlap community detection using spectral algorithm based on node convergence degree. *Future Generation Computer Systems 79*, 408–416. (Cited on page 25.)

Liu, Y. C., R. L. Kuo, and S. R. Shih (2020). Covid-19: The first documented coronavirus pandemic in history. *Biomedical Journal 43*, 328–333. (Cited on page 37.)

Otte, E. and R. Rousseau (2002). Social network analysis: a powerful strategy, also for the information sciences. *Journal of Information Science 28*, 441–453. (Cited on page 1.)

Papoulis, A. (1984). *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill. (Cited on page 17.)

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12*, 2825–2830. (Cited on page 66.)

Pol, J. (2019). Introduction to Network Modeling Using Exponential Random Graph Models (ERGM): Theory and an Application Using R-Project. *Computational Economics 54*, 845–875. (Cited on page 12.)

Raghavan, U. N., R. Albert, and S. Kumara (2007). Near linear time algorithm to detect community structures in large-scale networks. *Physical review. E, Statistical, nonlinear, and soft matter physics 76*, 036106. (Cited on page 29.)

Rajkumar, S. (2020). Novel corona virus 2019 dataset. (Cited on page 39.)

Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association 66*, 846–850. (Cited on pages 3, 34, 47 and 56.)

Reda, K., T. C. et al. (2011). Visualizing the evolution of community structures in dynamic social networks. *In: Computer Graphics Forum 30*, 1061–1070. (Cited on page 49.)

Reichardt, J. and S. Bornholdt (2006). Statistical mechanics of community detection. *Physical Review E 74*, 016110. (Cited on page 30.)

Rohani, P., X. Zhong, and A. A. King (2010). Contact network structure explains the changing epidemiology of pertussis. *Science 330*, 982–985. (Cited on page 1.)

Rossetti, G., L. Pappalardo, and S. Rinzivillo (2016). A novel approach to evaluate community detection algorithms on ground truth. In H. Cherifi, B. Gonçalves, R. Menezes, and R. Sinatra (Eds.), *Complex Networks VII*, Volume 644 of *Studies in Computational Intelligence*, pp. 133–144. Springer, Cham. (Cited on page 3.)

Rosvall, M., D. Axelsson, and C. T. Bergstrom (2009). The map equation. *The European Physical Journal Special Topics 178*, 982–985. (Cited on page 28.)

Savage, D., X. Zhang, et al. (2014). Anomaly detection in online social networks. *Social Networks 39*, 62–70. (Cited on page 3.)

Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics 6*, 461–464. (Cited on page 13.)

Scott, S. and C. J. Duncan (2001). *Biology of Plagues: Evidence from Historical Populations*. Cambridge: Cambridge University Press. (Cited on page 38.)

Stehlé, J., N. Voirin, et al. (2011). Simulation of an seir infectious disease model on the dynamic contact network of conference attendees. *BMC medicine 9*, 87. (Cited on page 1.)

Tan, F., Y. Xia, and B. Zhu (2014). Link prediction in complex networks: A mutual information perspective. *PLOS ONE 9*, 1–8. (Cited on page 3.)

Vinh, N. X., J. Epps, and J. Bailey (2010). Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research 11*(95), 2837–2854. (Cited on page 35.)

White, L. A., J. D. Forester, and M. E. Craft (2018). Dynamic, spatial models of parasite transmission in wildlife: Their structure, applications and remaining challenges. *Journal of Animal Ecology 87*, 559–580. (Cited on page 1.)

Wickramasinghe, A. N. and S. Muthukumarana (2021). Social network analysis and community detection on spread of covid-19. *Model Assisted Statistics and Applications 16*, 37–52. (Cited on page 38.)

Wilmink, F. W. and H. T. Uytterschaut (1984). Cluster analysis, history, theory and applications. In V. G. Van and W. Howells (Eds.), *Multivariate Statistical*

Methods in Physical Anthropology, pp. 135–175. Springer, Dordrecht. (Cited on page 25.)

Worldometers.info (2020). Worldometer covid-19 data. (Cited on page 39.)

Xu, B., B. Gutierrez, S. Mekaru, et al. (2020). Epidemiological data from the covid-19 outbreak, real-time case information. *Scientific data 7*, 106. (Cited on page 39.)

Yang, J. and J. Leskovec (2015). Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems 42*, 181–213. (Cited on page 3.)

Yilmazkuday, H. (2020). COVID-19 Spread and Inter-County Travel: Daily Evidence from the U.S. Technical report. (Cited on page 2.)

Zanin, M., P. Cano, J. M. Buldú, and O. Celma (2008). Complex networks in recommendation systems. In *Proceedings of the 2nd WSEAS International Conference on Computer Engineering and Applications*, Stevens Point, Wisconsin, USA, pp. 120–124. World Scientific and Engineering Academy and Society (WSEAS). (Cited on page 3.)

Zhang, P. (2015). Evaluating accuracy of community detection using the relative normalized mutual information. *Journal of Statistical Mechanics: Theory and Experiment 2015*(11), P11006. (Cited on pages 3, 35, 47 and 56.)