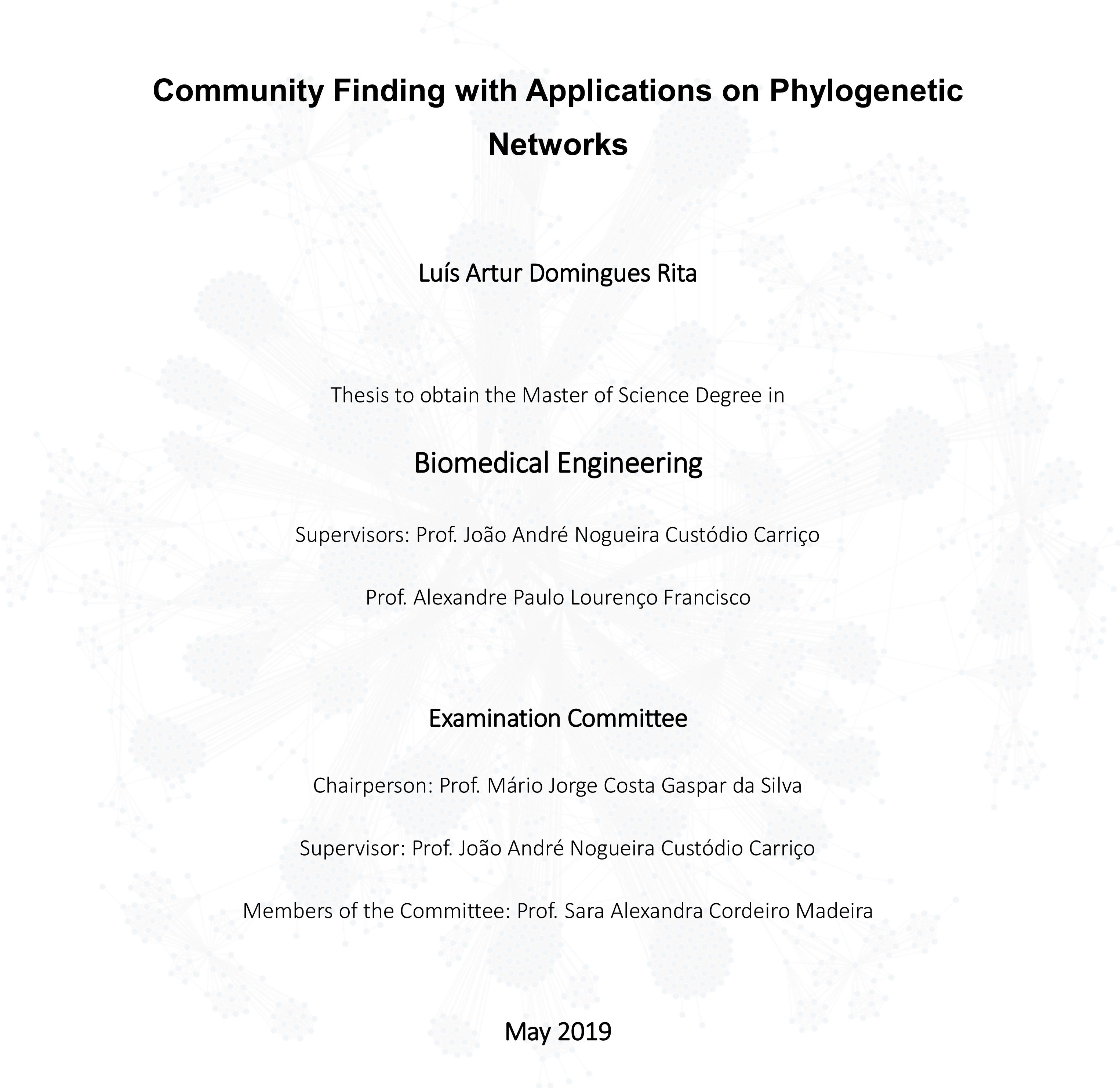


Community Finding with Applications on Phylogenetic Networks

Luís Artur Domingues Rita

Thesis to obtain the Master of Science Degree in

Biomedical Engineering

Supervisors: Prof. João André Nogueira Custódio Carriço

Prof. Alexandre Paulo Lourenço Francisco

Examination Committee

Chairperson: Prof. Mário Jorge Costa Gaspar da Silva

Supervisor: Prof. João André Nogueira Custódio Carriço

Members of the Committee: Prof. Sara Alexandra Cordeiro Madeira

May 2019

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Luís Rita

Preface

The work presented in this thesis was performed at the Ramirez, Mário Lab (Molecular Microbiology and Infection), Institute for Molecular Medicine, University of Lisbon (Lisbon, Portugal), during the period February 2018 – July 2019, under the supervision of Prof. João Carriço. And at the company National Health Institute Dr. Ricardo Jorge (Lisbon, Portugal), during the period May – July 2019, under the supervision of Dr. Vítor Borges. The thesis was co-supervised at Instituto Superior Técnico by Prof. Alexandre Francisco.

Abstract

With the advent of high-throughput sequencing methods, new ways of visualizing and analyzing increasingly amounts of data are needed. Although some software already exist, they do not scale well or require advanced skills to be useful in phylogenetics.

The aim of this thesis was to implement three community finding algorithms – Louvain, Infomap and Layered Label Propagation (LLP); to benchmark them using two synthetic networks – Girvan-Newman (GN) and Lancichinetti-Fortunato-Radicchi (LFR); to test them in real networks, particularly, in one derived from a *Staphylococcus aureus* MLST dataset; to compare visualization frameworks – Cytoscape.js and D3.js, and, finally, to make it all available online (mscthesis.herokuapp.com).

Louvain, Infomap and LLP were implemented in JavaScript. Unless otherwise stated, next conclusions are valid for GN and LFR. In terms of speed, Louvain outperformed all others. Considering accuracy, in networks with well-defined communities, Louvain was the most accurate. For higher mixing, LLP was the best. Contrarily to weakly mixed, it was advantageous to increase the resolution parameter in highly mixed GN. In LFR, higher resolution decreased the accuracy of detection, independently of the mixing parameter. The increase of the average node degree enhanced partitioning accuracy and suggested detection by chance was minimized. It was computationally more intensive to generate GN with higher mixing or average degree, using the algorithm developed in the thesis or the LFR implementation. In the *S. aureus* network, Louvain was the fastest and the most accurate in detecting the clusters of seven groups of strains directly evolved from the common ancestor.

Community Finding | Phylogenetic Networks | Data Visualization | Web Application

Resumo

Com os recentes métodos de sequenciamento de alto rendimento, novas formas de visualizar e analisar dados são necessárias. Apesar de já existirem algumas ferramentas, estas não são escaláveis ou requerem elevado conhecimento técnico para serem úteis em filogenia.

O objetivo da tese foi implementar três algoritmos de deteção de comunidades – Louvain, Infomap e Layered Label Propagation (LLP); submetê-los a testes de desempenho, utilizando duas redes sintéticas: Girvan-Newman (GN) e Lancichinetti-Fortunato-Radicchi (LFR); testá-los em redes reais, nomeadamente, numa criada a partir de um perfil de MLST de *Staphylococcus aureus*; comparar ferramentas de visualização de dados - Cytoscape.js e D3.js, e implementar uma aplicação web englobando tudo isto (mscthesis.herokuapp.com).

Louvain, Infomap e LLP foram implementados em JavaScript. Por omissão, as próximas conclusões são válidas para as redes GN e LFR. Louvain foi o mais rápido dos três, e o mais preciso quando executado em redes com comunidades bem definidas. Em redes com maior mistura, LLP apresentou os melhores resultados. Foi vantajoso aumentar o parâmetro de resolução em redes GN mal definidas, contrariamente a bem definidas. Em redes LFR, o aumento do mesmo parâmetro piorou a partição obtida. O aumento do grau médio dos nós melhorou a partição encontrada e sugeriu uma menor deteção fortuita de comunidades. Foi computacionalmente mais exigente gerar redes GN com maior mistura ou grau médio, utilizando o algoritmo aqui desenvolvido ou o da implementação LFR. Em *S. aureus*, Louvain foi o mais rápido e o preciso na deteção dos conjuntos de estirpes derivadas diretamente do ancestral comum.

Deteção Comunidades | Redes Filogenéticas | Visualização Dados | Aplicação *Web*

Acknowledgments

Truly grateful to my supervisors: Alexandre Francisco, from INESC-ID and IST; João André Carriço, from Institute of Molecular Medicine, and Vítor Borges, from National Health Institute Dr. Ricardo Jorge, that crucially contributed to the development of my master thesis. To European Institute of Innovation and Technology Health which allowed me to look for different insights internationally: Madrid, Spain (1st Workshop Master in Technological Innovation in Health); Mannheim, Germany (International Master in Innovative Medicine – Industry & Innovation Day) and Heidelberg, Germany (Antimicrobial Resistance Hackathon 2018). From students and professors from some of the best European medical/engineering universities. To the Student Association from Abel Salazar Biomedical Sciences Institute for giving me the opportunity to present a poster of the thesis in their Biomedical Congress in Oporto, Portugal. And to Santo Fortunato who helped me along the way of the benchmark testing.

Table of Contents

List of Figures

List of Tables

List of Algorithms

List of Acronyms

AGM	Community-Affiliation Graph Model
AMR	Antimicrobial Resistance
CC	Clonal Complex
cgMLST	Core Genome Multilocus Sequence Typing
CI	Confidence Interval
CSS	Cascading Style Sheets
CSV	Comma-Separated Values
GN	Girvan-Newman
HTML	Hypertext Markup Language
JS	JavaScript
JSON	JavaScript Object Notation
LFR	Lancichinetti-Fortunato-Radicchi
LLP	Layered Label Propagation
LP	Label Propagation
MDL	Minimum Description Length
MLST	Multilocus Sequence Typing
MLVA	Multilocus Variable-Number Tandem Repeat
NMI	Normalized Mutual Information
NPM	Node.js Package Manager
rMLST	Ribosomal Multilocus Sequence Typing
SNP	Single Nucleotide Polymorphism
SVG	Scalable Vector Graphics
wgMLST	Whole Genome Multilocus Sequence Typing

1. Introduction

According to *The Economist*, data has become the most important resource in the world [1]. New ways to storage, analyze and visualize increasing amounts of information are becoming a pressing need.

Interactions among different elements of real systems can be seen as networks [2]. From their topology or pattern of connections, functional knowledge can be extracted. Although empirical analysis can be useful in small networks, for big-data it is indispensable the use of algorithms to uncover their properties. Consequently, mathematical models called graphs are used to represent them. Graph theory provides all the tools available to analyze them. This representation implies all the elements of the network and their relations must be acquired. Generally, the closer to the human population is the analysis, the more difficult it gets to gather data. The two main reasons are the lack of non-invasive acquisition methods or privacy issues [3]. In fact, in health, legislation is becoming progressively stricter in respect to patient's data protection [4]. On one hand, it is fundamental to assure ethical use of medical data, on the other, it may constitute a hurdle for investigating the causes of many pathologies. In the next lines, the discussion will focus in the specific topic of infectious diseases.

As stated by the *World Health Organization*, Antimicrobial resistance (AMR) is in the top ten of major threats demanding attention in 2019 [5]. In the last years, AMR has been potentiated by the use of drugs intended to treat diseases caused by pathogenic organisms. Depending on the organism, it is subdivided in antiviral (virus), antibiotic (bacteria), antifungal (fungi) or antiparasitic (parasites) resistance. Although resistance is a natural evolutionary process, in which the most competent strains get selected over time, the frequency of use, or even misuse, of antimicrobial drugs is accelerating the growth of resistance rates for several microbial species [6]. This is particularly relevant not only among the human population, but also in farms where cattle are being given preventive doses of antibiotics as growth adjuvants. Moreover, the frequency people are travelling around the world is facilitating the spread of resistant organisms. Due to all of this and the high mortality of infectious diseases (Figure 1), it is fundamental to empower central health authorities with scalable and easy-to-use tools to allow their intervention in short time scales [7].

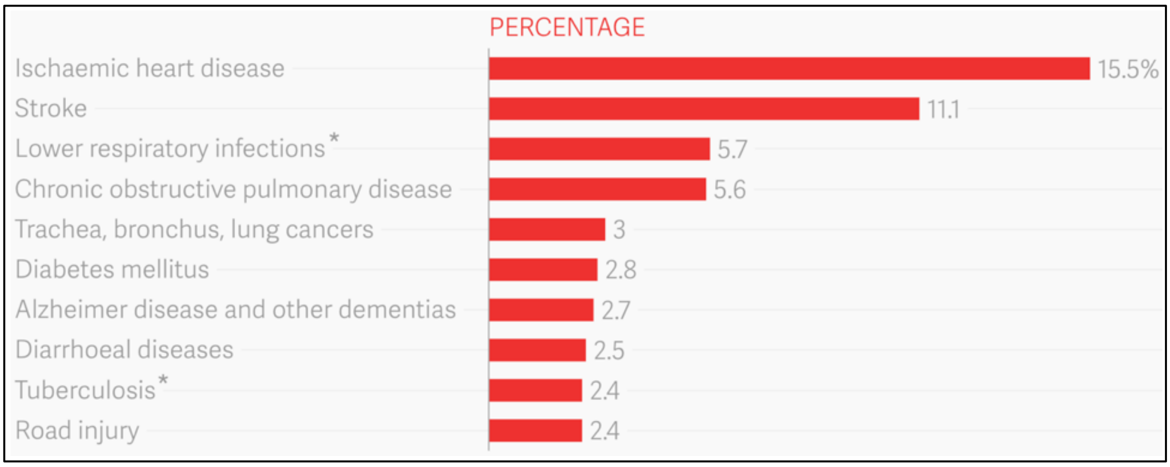


Figure 1 Infections are still one of the major causes of death (*). Data collected by WHO between 2000-2015 from a population of over 90 000 people, in WHO member states. Adapted from [8].

Next Generation Sequencing (NGS) methods, developed in the last years, have been providing us massive quantities of microbiological data [9]. In fact, due to the high discriminative power of whole genome sequencing methods, it has been possible to infer more precise evolutionary relationships among strains. Consequently, allelic profiles with higher number of genes keep increasing and, with them, more complete phylogenetic trees or networks depicting evolutionary distances among strains can be assessed. All of this has been useful in the multiple stages of disease prevention: vaccine design, assess the pathogenicity of a sequenced strain, detect outbreaks and in infection control. As well as diagnosis, enhancing the detection of mixed infections (caused by multiple genetically distinct organisms) [10].

Generally, the goal of this thesis was to extract functional knowledge from the topological structure of phylogenetic networks. For instance, to be able to infer properties like resistance to antimicrobials, based on the similarity and characteristics provided by the other organisms in the network. Contrarily to random networks, real ones present an asymmetrical distribution of edges. From this asymmetry, clusters of nodes can be identified. By definition, communities are regions in the network with higher density of edges. Nonetheless, this definition is not restrictive enough to undoubtedly identify them in graphs. This raises the following question: how can we detect communities if it is not clear what we are looking for? To answer this question, community finding theory has been developed in the past years. In spite there has been a considerable advance in terms of accuracy and speed of detection, some algorithms perform better than others in different networks.

1.1 Objectives

The aim of the thesis was to implement Louvain, Infomap and Layered Label Propagation (LLP) algorithms and to benchmark them against two synthetic networks – Girvan-Newman (GN) and Lancichinetti-Fortunato-Radicchi (LFR) networks. To test them using an Amazon (disconnected) and Zachary's Karate Club (small sized) networks, with varying input parameters. *Staphylococcus aureus* Multilocus Sequence Typing (MLST) Single Locus Variant (SLV) (biological) network was analyzed in detail. To compare two visualization frameworks – D3.js (using Scalable Vector Graphics – SVG, and Canvas elements) and Cytoscape.js, in terms of their tools and robustness to big data. Finally, to develop a web application wrapping it all and running in every browser or device. For advanced users, it was also intended to make all algorithms available in Node.js Package Manager (NPM) for server-side user-specific applications.

1.2 Thesis Outline

In Background, important concepts of graph/network theory are presented, along with community finding and phylogenetic topics. One algorithm representative of each class of community finding methods is introduced. The ones used in the thesis are detailed in Methodology. At the beginning of this section, a clear distinction between the review and innovative components is made. Then, community finding algorithms, benchmark tools, real test data and data visualization libraries are explored. In Results, community finding algorithms are analyzed in terms of accuracy and speed. The speed of generation of benchmark networks is determined based on different properties of the networks. Different web visualization frameworks were compared. And a *S. aureus* MLST SLV network was partitioned using different algorithms and input parameters. In Conclusion, the aim of the thesis is recalled, a retrospective analysis on whether the goals were attained or not is made, main difficulties, future work to be performed in INSaFLU web application is detailed and a set of prospective improvements to the work already done is suggested.

2. Background

In this section, concepts will be introduced from the broadest to the more specific ones. Graph theory and network properties are introduced, along with the models describing their generation and evolution mechanisms. Before introducing community finding algorithms, algorithmic complexity concepts are included. Problems of different complexity are presented, and it is shown it becomes quickly unfeasible to use brute-force approaches to solve optimization problems, when the size of the network increases. Then, a list of community finding algorithms spanning the majority of the most well-known classes is included. A section of phylogenetics is introduced at last and it shows how networks can be used to represent evolutionary relationships among taxa. A series of tree building algorithms are referred along with PHYLOViZ, a phylogenetic analysis tool that allows the user not only to build these trees online, but also to analyze them.

2.1 Graph Theory

Networks are a particular type of graphs that represent real systems and the interaction among their components. Graphs are said to be constituted by vertices and edges (Figure 2), on the other hand, in networks, they are identified as nodes and links. In spite of the different terminology, they aim to simplify and facilitate the analysis of real networks [2].

Graphs are the object of study of Graph Theory, which joins a set of formalities indispensable for their study in a standardized and meaningful way.

Networks are ubiquitous in the present world. In an increasing order of magnitude and anthropocentric point of view: protein interactions, neurons organization/communication, human interactions and societal behavior are often explained by current Graph Theory [11].

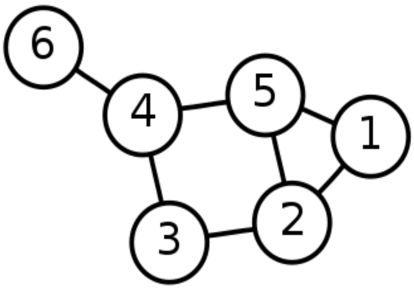


Figure 2 An example of a graph. 6 vertices connected by 7 edges [12].

2.1.1 Networks

A key property of nodes is their degree. It is defined as the sum of the weights of the links to other nodes. Links are the structures connecting nodes. These can be directed, undirected, weighted or unweighted depending on the purpose of the analysis or the information available. In directed links, node degree property splits in incoming and outgoing degrees. Considering, respectively, the sum of the links pointing towards the node or outwards. Both nodes and links local properties have effects in the whole network.

The sum of the weights of all links in the network is identified as the total number of links:

$$L = \frac{1}{2}\sum_{i=1}^{N} k_i \tag{1}$$

$\frac{1}{2}$ coefficient corrects for the fact each link is counted twice in undirected networks. In directed networks, the total number of links is given by:

$$L = \sum_{i=1}^{N} k_i^{in} = \sum_{i=1}^{N} k_i^{out} \tag{2}$$

The average degree is given by:

$$\langle k \rangle = \frac{1}{N}\sum_{i=1}^{N} k_i \tag{3}$$

And, in the case of directed networks, where:

$$k_i = k_i^{in} + k_i^{out} \tag{4}$$

The average incoming and outgoing degrees are:

$$\langle k^{in} \rangle = \frac{1}{N}\sum_{i=1}^{N} k_i^{in} = \langle k^{out} \rangle = \frac{1}{N}\sum_{i=1}^{N} k_i^{out} = \frac{L}{N} \tag{5}$$

One measure that arises from the pattern of connections between nodes is the degree distribution. It gained a particular relevance when random and scale-free networks were discovered.

All previous measures can be derived from a single mathematical representation of the network: the adjacency matrix A_{ij}. It includes not only the connections between nodes in the network, but also the respective weights (in case the network is weighted).

Whenever nodes i and j are connected in an unweighted network, the adjacency matrix entry is:

$$A_{ij} = 1 \tag{6}$$

In a weighted network:

$$A_{ij} = w_{ij} \tag{7}$$

w_{ij} is the weight of the link. If the same pair of nodes is not connected:

$$A_{ij} = 0 \tag{8}$$

The degree of each node can be derived based on this matrix:

$$k_i = \frac{1}{N}\sum_{j=1}^{N} A_{ji} = \frac{1}{N}\sum_{i=1}^{N} A_{ji} \tag{9}$$

In directed networks, the incoming and outgoing degrees are:

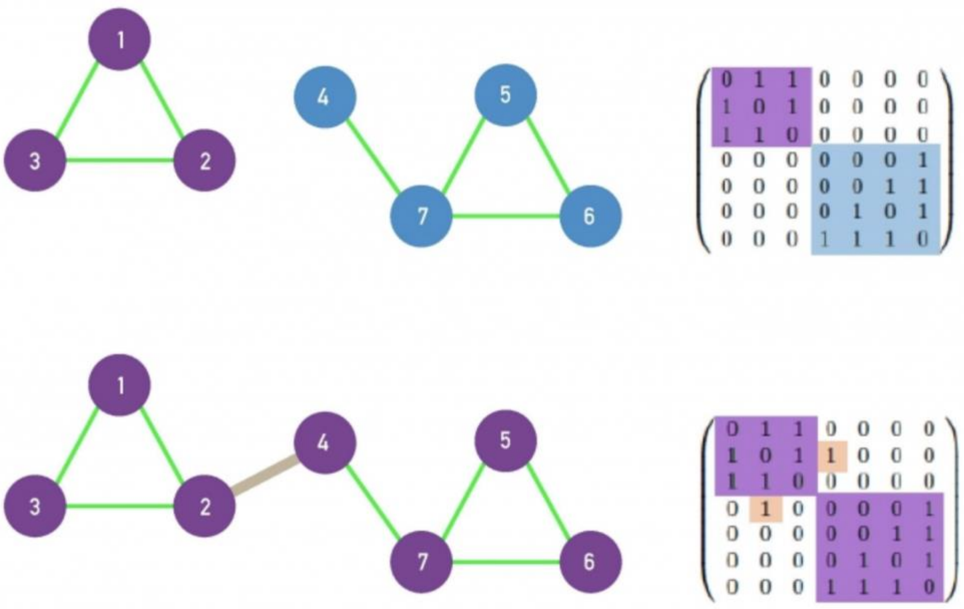
$$k_i^{in} = \frac{1}{N}\sum_{j=1}^{N} A_{ij} \tag{10}$$

$$k_i^{out} = \frac{1}{N}\sum_{j=1}^{N} A_{ji} \tag{11}$$

$$2L = \sum_{ij}^{N} A_{ij} \tag{12}$$

Still regarding the overall topology of the network, a set of additional concepts will help us understanding the theory behind. Paths in the network and distances between nodes are common definitions used in network theory. Generally, a path is a sequence of nodes connected by the respective links joining 2 nodes. Shortest paths are constituted by the sequence of nodes that minimize the distance between the pair. Multiple shortest paths can coexist in the same network. An average path length is defined as the average distance among the multiple available paths connecting the pair of nodes. Whenever a path has the same start and end node, it is called a cycle. Eulerian paths are the ones that do not contain repeated links. The diameter of a network can be defined as the longest distance among all the shortest paths between every node pair.

Connectedness is another important property of networks. A network is considered connected if it is possible to reach any node starting in any other. When this is not possible, multiple components coexist in the network. This property has consequences in the respective adjacency matrix. In the presence of a disconnected network, it is possible to write it in the block diagonal form (Figure 3).



$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

Figure 3 Adjacency matrix of a connected and a non-connected network [2].

Most real networks are sparse. This means the number of links can be approximated by the number of nodes. Assuming the maximum number of links in an undirected network is given by:

$$L_{max} = \frac{N(N-1)}{2} \tag{13}$$

The fraction $\frac{L}{L_{max}}$ is close to 0. This means most entries in the adjacency matrix are 0. Thus, it is often more efficient to store the links instead of the nodes of a network.

Local clustering coefficient examines the degree the neighbors of a given node are connected to each other:

$$C_i = \frac{2L_i}{k_i(k_i - 1)} \tag{14}$$

L_i is the total number of edges connecting neighbors of node i and k_i its degree. $C_i$is 1 when all neighbors are connected and 0 when there are no links connecting them.

Generalizing this concept to the whole network:

$$\langle C \rangle = \frac{1}{N}\sum_{i=1}^{N} C_i \tag{15}$$

This is the probability of finding 2 neighbors connected in a random network.

One way to determine the value of a real network is given by the Metcalfe's Law. It states the value is proportional to the number of links between nodes. This means, it is proportional to the square number of nodes. In the case of technological networks, the cost of a network is proportional to the number of nodes (Figure 4).

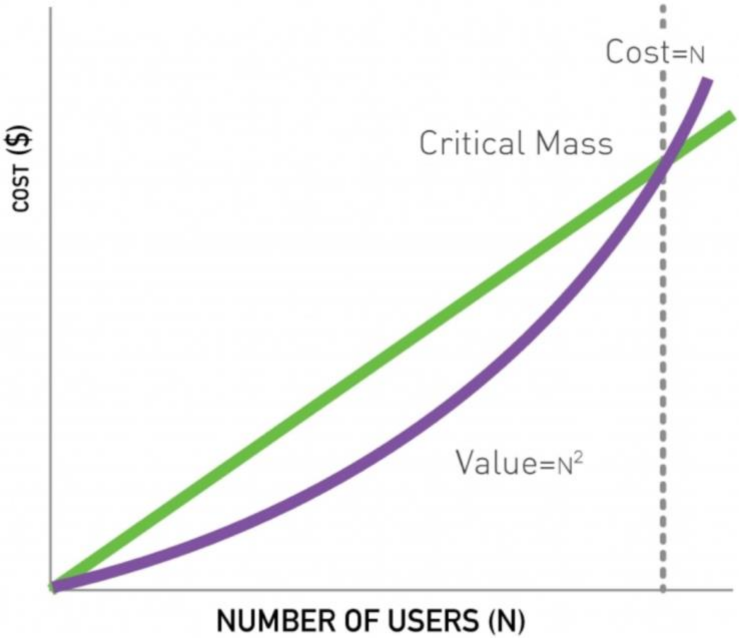


Figure 4 Variation of the value and costs of a common technological network in terms of the number of users [2].

2.1.2 Network Models

In order to model real networks, random models were the first to be proposed. Although they do not accurately replicate the structure of real networks, there are community finding algorithms making use of the randomized version of the original network. After obtaining degree-preserved randomization and, based on the difference of link density expected by chance and the one in the original network, a partition is inferred.

After introducing two random models, one that better captures several properties of real networks (including phylogenetic networks) – Albert-Barabási Model, is presented. Its introduction is useful not only to understand the generative processes in their origin, but also their evolution.

2.1.2.1 Random Model

There are two random model definitions that differ in the fixed variables. The one proposed by Erdős and Rényi [13] considers the number of links and the total number of nodes in the network fixed. In the model introduced by Gilbert [14], the probability of connection between two random nodes is fixed, along with the total number of nodes.

While Erdős and Rényi assume an unrealistic property, fixed number of links, Gilbert model is considered to be the one that better approximates a real network. By omission, it is considered until the end of this section.

Next, the features that arise from the random model are going to be tested against what is verified in real networks. In order to calculate the average number of links and the average node degree of the model, the probability of finding L links in the network is calculated:

$$p_L = \binom{\frac{N(N-1)}{2}}{L} p^L (1-p)^{\frac{N(N-1)}{2}-L} \tag{16}$$

Once p_L is a binomial distribution, the average number of links is:

$$\langle L \rangle = \sum_{L=0}^{\frac{N(N-1)}{2}} L p_L = p\frac{N(N-1)}{2} \tag{17}$$

The average degree of the nodes in the network is:

$$\langle k \rangle = \frac{2\langle L \rangle}{N} = p(N-1) \tag{18}$$

Similarly to (16), (19) describes network's degree distribution:

$$p_k = \binom{N-1}{k} p^k (1-p)^{N-1-k} \tag{19}$$

Although (19) is the exact formula of the degree distribution in a random network, a Poisson alternative is well approximated for $N >> \langle k \rangle$:

$$p_k = e^{-\langle k \rangle} \frac{\langle k \rangle^k}{k!} \tag{20}$$

Once most real networks are sparse, this is a good approximation. Poisson distribution facilitates the calculation of network's average degree $\langle k \rangle$, second moment $\langle k^2 \rangle$ (useful for estimating variance), third moment $\langle k^3 \rangle$ (skewness measure) and standard deviation σ. For this reason, unless otherwise stated, this distribution will be further considered (Figure 5).

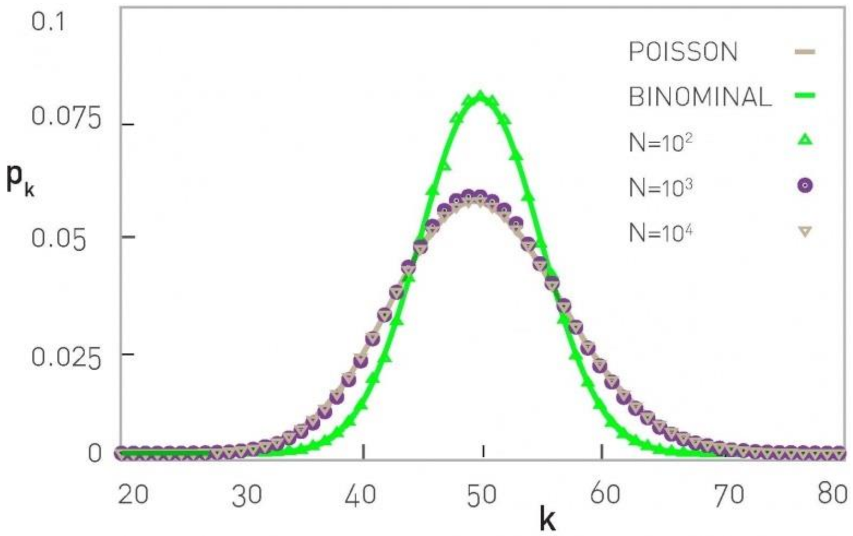


Figure 5 Binomial and Poisson distributions of networks with different number of nodes [2].

There are 3 properties of real networks that are not in agreement with the random model. Those are the presence of hubs, connectedness and the clustering coefficient of the networks.

Replacing the factorial of the number of links in the degree distribution, it becomes clear the reason why hubs are not present:

$$k! \approx \sqrt{2\pi k}\left(\frac{k}{e}\right)^k \tag{21}$$

Rewriting (20),

$$p_k = e^{-\langle k\rangle}\frac{\langle k\rangle^k}{\sqrt{2\pi k}\left(\frac{k}{e}\right)^k} = \frac{e^{-\langle k\rangle}}{\sqrt{2\pi k}}\left(\frac{e\langle k\rangle}{k}\right)^k \tag{22}$$

Whenever $e\langle k\rangle < k$, p_k decreases faster than exponentially due to the denominator terms.

In real networks, most nodes are connected. In order to observe this in random networks, it is necessary to increase the average node degree until a given threshold. When this happens, a giant component appears connecting most of the nodes. In order to estimate this value, the probability a random node is not connected to this component is calculated:

$$(1-p)^{N_G} \approx (1-p)^N \tag{23}$$

Considering the giant component spreads approximately over the whole network, $N_G \approx N$. Consequently, the expected number of isolated nodes is:

$$I_N = N(1-p)^N = N\left(1-\frac{Np}{N}\right)^N \approx Ne^{-Np} \tag{24}$$

Assuming all nodes are connected:

$$I_N = 1 \Leftrightarrow Ne^{-Np} = 1 \Leftrightarrow p = \frac{\ln(N)}{N} \tag{25}$$

Once $Np = \langle k\rangle$,

$$\langle k\rangle = \ln(N) \tag{26}$$

The inconsistency between the random model and real networks is that the latter are connected and present an average node degree significantly lower than (26).

Local clustering coefficient is given by the proportion of links there are between the neighbors of a given node and all the possibilities:

$$C_i = \frac{2p\frac{k_i(k_i-1)}{2}}{k_i(k_i-1)} = p = \frac{\langle k\rangle}{N} \tag{27}$$

In random networks, clustering coefficient is independent of k. Although, in real networks, this is not the case.

Due to these 3 incompatibilities, it was necessary to build a model that could explain them.

2.1.2.2 Barabási-Albert Model

This model was proposed by Albert-László Barabási and Réka Albert. It focuses not only on the structural aspect of networks, but also in their formation process. Thus, it is expected to fill the gaps left by the random models.

Hubs present in real networks and network's degree distribution are explained by two fundamental phenomena of their formation: growth and preferential attachment.

The model starts by considering a set of m_0 connected nodes. At each time step, a node with m links is added to the network ($m \leq m_0$). The links are connected to node i with a probability $\prod(k_i)$, which is proportional to its degree:

$$\prod(k_i) = \frac{k_i}{\sum_j k_j} \tag{28}$$

This way, the rate node i increases its degree is proportional to k_i and m:

$$\frac{dk_i}{dt} = m\prod(k_i) = m\frac{k_i}{\sum_{j=1}^{N-1} k_j} \tag{29}$$

Once,

$$\sum_{j=1}^{N-1} k_j = 2mt - m \tag{30}$$

The rate of change of k_i is given by:

$$\frac{dk_i}{dt} = \frac{k_i}{2t-1} \tag{31}$$

Rearranging and integrating (31), $k_i(t)$ is obtained:

$$k_i(t) = m\left(\frac{t}{t_i}\right)^{\frac{1}{2}} \tag{32}$$

(32) suggests every node increases its degree at the same rate, independently of the moment it was added to the network. Not only this assures a universal power-law increase, but also suggests that older nodes tend to have

more connections. The latter is usually called, in many domains, the first-mover advantage. Particularly important in the context of phylogenetics in the detection of ancestrality. This property justifies the presence of hubs and, consequently, the connectedness of the whole network.

In order to derive the degree distribution of the Barabási-Albert model, cumulative degree distribution is calculated.

Supposing,

$$k_i(t) < k \Leftrightarrow m\left(\frac{t}{t_i}\right)^{\frac{1}{2}} < k \Leftrightarrow t_i < t\left(\frac{m}{k}\right)^2 \tag{33}$$

The number of nodes with degree lower than k is $t\left(\frac{m}{k}\right)^2$. Approximating the total number of nodes,

$$N = m_0 + t \approx t \tag{34}$$

The cumulative distribution is given by:

$$P(k) = 1 - \left(\frac{m}{k}\right)^2 \tag{35}$$

The degree distribution is the derivative of (35):

$$p_k = \frac{dP(k)}{dk} = \frac{2m^2}{k^3} \tag{36}$$

(36) evidences the power-law degree dependence, contrarily to the exponential verified in the Poisson/Binomial distributions. Whenever a network follows a power law distribution:

$$p_k \propto k^{-\gamma} \tag{37}$$

It favors the existence of hubs due to its long tail, which includes several nodes with higher degrees. Networks with this characteristic are called scale-free [11]. This term refers to the nature of the second and third moments of the degree distribution that diverge to infinity when the degree exponent is $2 \leq \gamma \leq 3$ (verified in many real networks) and the total number of nodes in the network goes to infinity [2].

While in the random model the average clustering coefficient is proportional to $\frac{1}{N}$, in Barabási-Albert networks, it behaves differently:

$$\langle C \rangle \propto \frac{(lnN)^2}{N} \tag{38}$$

The term in the numerator increases the clustering coefficient in large networks. This prediction is in line with what is verified in real networks, higher clustering [2].

However, there are a couple of open questions Barabási-Albert model cannot explain. First, it considers the degree exponent $\gamma = 3$. While, in real networks, $2 \leq \gamma \leq 5$ [2]. Only undirected networks were covered. Other dynamics for node and link generation were not considered. Finally, differences among nodes were resumed to their degrees, not considering other intrinsic characteristics.

2.1.3 Algorithmic Complexity

Due to the size of real networks, it is sometimes unfeasible to use brute-force algorithms to define communities. Algorithms used to handle these problems, in the best-case scenario, run in polynomial time. Although, most of the times, it is necessary to explore an exponential number of possibilities which grows with the network size. First problems are called *P problems* and, the second, *NP problems*.

In the next sub-sections, the problem of whether *P = NP* is explored and the consequences of this confirmation or refutation are broadly presented, but that can be directly extrapolated to the domain of community finding algorithms presented in the thesis. After, it is introduced a system that characterizes algorithms' complexity in terms of their speed and memory requirements – Big-O Notation.

2.1.3.1 P vs NP problem

P vs NP problem was introduced in 1971 by Stephen Cook [15] and, in 2000, was considered one of the most important open questions in computer science [16]. It is part of a set of six problems that were defined by the Clay Mathematics Institute as the 7 Millennium Prize Problems [17]. And the person who can solve one of them, will be awarded with a prize of 1 million dollars. *P vs NP problem* can be stated as: *Can every problem whose solution can be quickly verified by a computer also be quickly solved by a computer?* [18].

P problems are characterized for being solvable in polynomial time by a Deterministic Turing Machine (DTM). On the other hand, *NP* category includes those only solvable in polynomial time using a Non-deterministic Turing Machine (NTM). They are both part of a set of decision problems whose answer is binary. DTM is a non-branching machine that can only proceed to one next step at each time. This is how an ordinary computer works. NTM is a conceptual machine that is able to analyze, simultaneously, a polynomial number of different options at each step. Thus, a branching device. A *quickly verified* solution means it can be found in polynomial time. The word computer refers to a Deterministic Turing Machine (DTM).

Although the question whether *P = NP* remains unanswered, by definition, there is a relation between the 2 groups. *NP problems* include those whose solution may not be obtainable but can be verified in polynomial time in a DTM. This means, every *P problem* belongs to *NP* once it is always easier to verify a solution than to find it. And they can find it in polynomial time as discussed. Trivially, $P \subset NP$.

A different class of problems are the NP-complete. Not only they are NP, but also NP-hard (Figure 6). A problem is NP-hard when an algorithm belonging to NP can be reduced to another in the first class. This means, any problem in NP-hard class is at least as hard to solve than any in NP. The reduced term means that the inputs of a problem A can be converted in the inputs of a problem B, using a polynomial time algorithm, and provide the exact same output. In case A can be reduced to B, this has a couple of immediate consequences:

1. If $B \in P$, then $A \in P$;
2. If $B \in NP$, then $A \in NP$;
3. If A is NP-hard, then B is NP-hard.

Consequently, if a solution in polynomial time is found to a NP-complete problem, then *P vs NP problem* is solved and $P = NP$. Thus, it is not only fundamental to find a polynomial solution but also to prove a problem X is NP-complete. Based on the definition above, this can be split in 2 steps:

1. Showing X belongs to NP. It can be done by verifying a given solution in polynomial time or finding a non-deterministic algorithm for X;
2. Showing X is NP-hard. By reducing a known NP-complete problem to X. In this circumstance, the third consequence derived above implies X is NP-hard.

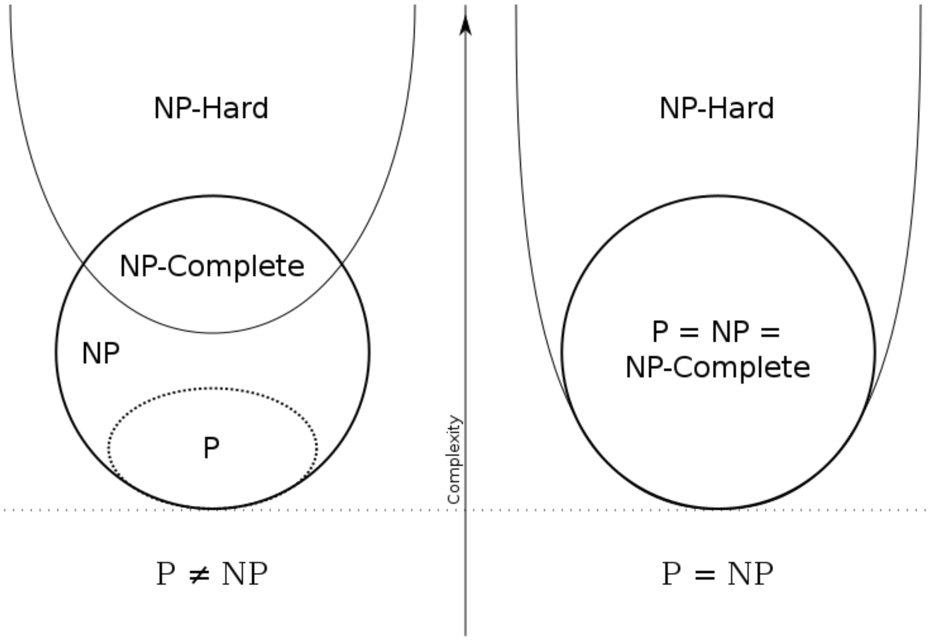


Figure 6 *P vs NP problem* diagram representation [19].

Although it is not proved, it is believed by the majority of computer scientists $P \neq NP$ [20]. Proving $P = NP$ would have important consequences in the real world. Not only biological modelling would be enhanced, but also the efficiency in transportation systems, multimedia processing, economic simulations, among others, would be dramatically improved. On the other side, there are real systems based on the fact $P \neq NP$. One of them are the modern cryptographic systems which rely in prime factorization to secure data. This problem is considered *NP*.

A significant proportion of problems in network science are either NP, NP-hard or NP-complete. Thus, it is not surprising the importance of the solution in, for example, finding communities in phylogenetic networks.

A mathematical representation of algorithmic time/memory complexity is presented next.

2.1.3.2 Big-O Notation

While analyzing algorithmic performance, time and memory complexities are usually considered. They are particularly important when there is limited computational power and big data analysis is required.

In some cases, the solution for the same problem can be found in disparate time scales. For this reason, it is important to have a way to upper bond the algorithms' execution time, but also to estimate its average value. This led to the creation of Big-O notation.

Depending on the number of instructions the algorithm has to execute and the size of the input, the runtime differs. Big-O notation offers a way to quantify the execution needs of an algorithm, independently of the machine that is running it. The identified complexity can be linear, logarithmic, quadratic, cubic, exponential… An ordered list, in terms of the time of execution of the most common algorithms (with some examples), is presented in Table 1.

Table 1 Time complexity of some of the most well-known algorithmic approaches.

Time Complexity	Algorithm Description	Reference
N	Analysis and manipulation of arrays.	[21]
$\log(N)$	Divide-and-conquer algorithms. The main problem in decomposed in several smaller ones.	[22, 23]
N^2	Adjacency matrices often imply the use of quadratic algorithms. Cubic or higher order degrees are valid for similar data structures of higher dimensionality.	[24]
2^N	Few algorithms are useful with this complexity. When brute-force approaches are the only option, they are considered	[25]

After determining the complexity of an algorithm, it is possible to predict the time and memory requirements of the machine to execute it.

Definition

A function $f(N)$ is said to be $O(g(N))$ if constants k and N_0 exist, such that $0 \leq f(N) \leq k \times g(N)$ for all $N \geq N_0$ [26].

2.2 Community Finding

The first community finding experiment remotes 1927, when Stuart Rice tried to identify clusters in political bodies, using voting patterns [27]. In 1959, Erdős and Rényi introduced the first model of a random network [13]. Although, it was only in 2002, when Girvan and Newman used the Zachary's Karate Club network as a benchmark test, that community detection gained an increased interest among the scientific community [23]. As soon the paper was published, the number of citations in the theme steeply rose.

Along the thesis, lacking a clear definition of community, 4 hypotheses were formulated to provide us an intuition about the properties of these structures.

Fundamental Hypothesis

It states that network's community structure is uniquely encoded in its wiring diagram [2]. Thus, they cannot be identified based exclusively in the degree distribution of a network. This can be verified upon degree-preserved randomization of networks.

Connectedness and Density Hypothesis

A community can be described as a dense and connected subgraph [2]. Depending on other criteria, stricter or looser rules can be used to refine this hypothesis. Identifying communities as maximum cliques, strong or weak communities are explored in the next section.

Random Hypothesis

Random networks do not present community structure [2]. This hypothesis allows us to find communities by comparing the studied network with a degree-preserved randomization of itself. This comparison measure is called modularity.

Maximal Modularity Hypothesis

A partition with higher modularity offers a better representation of the network's community structure [2]. This quality measure is detailed in section 1.2.5.

2.2.1 Communities

Based on the hypotheses above, additional criteria to clearly identify communities is proposed. Although, it is important to state that, for now, there is no golden-standard working the best for every network or purpose.

In the first references to communities, it was required all nodes to be connected to all others inside the same group – maximum clique [2]. Thus, each community is identified as a clique – a complete subgraph. This definition was considered too rigid. In spite most real networks present triangles of connected nodes, larger cliques are rare. Moreover, this is not a robust definition of a community. One missing link would be enough to disqualify it as a clique. None of the networks present in the thesis would be correctly partitioned using it.

To overcome the rigidity of cliques, two definitions were proposed: strong and weak communities. The definition of a strong community states that the inner degree of every node i should be greater than its external degree [2]:

$$k_i^{int}(C) > k_i^{ext}(C) \tag{39}$$

A weak community relaxes the previous definition in the way it only requires the sum of the internal degrees of every node from a community to be greater than the sum of the external degrees [2]:

$$\sum_{i \in C} k_i^{int}(C) > \sum_{i \in C} k_i^{ext}(C) \tag{40}$$

In k-cores, a community is formed by a set of nodes which are connected to at least k others belonging to the same community. In the same network, k-cores of different degrees may coexist [28].

In p-cliques, the network is partitioned in groups in which all nodes have, at least, a fraction $0 \leq p \leq 1$ of links connecting them to nodes in the same cluster [29].

Whenever all nodes are separated by a distance not higher than n, they are part of the same community [30]. These communities are called n-cliques.

2.2.2 Graph Partitioning vs Community Finding

Before presenting community finding algorithms, it is important to distinguish graph partitioning from community finding. In the first case, it is known the number of communities and the size of each one in the network. In community finding, the problem resides in partitioning the original graph without knowing, *a priori*, the number of communities or their sizes. This condition significantly increases the computational power needed to unleash them [2].

Graph Partitioning

The problem of reducing the number of edges connecting different groups is highly important in, for example, circuitry design and data processing. Whenever printing transistor (fundamental building block of modern electronic devices) circuitry in modules, it is fundamental to reduce wire overlap. Moreover, multicore property of actual CPUs is turning parallel computation possible. The goal here is to reduce communication between cores which significantly increases execution time [31].

The simplest problem of graph partitioning is called graph bisection. By analyzing the number of possible partitions in a network with size N and 2 communities with N_1 and N_2 nodes, it becomes rapidly unfeasible to analyze every single one. The number of possible partitions is given by:

$$\frac{N!}{N_1!\,N_2!} \tag{41}$$

Using Stirling approximation, a more meaningful formula is deduced:

$$n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n \tag{42}$$

$$\begin{aligned}\frac{N!}{N_1!\,N_2!} &\approx \frac{\sqrt{2\pi N}\left(\frac{N}{e}\right)^N}{\sqrt{2\pi N_1}\left(\frac{N_1}{e}\right)^{N_1}\sqrt{2\pi N_2}\left(\frac{N_2}{e}\right)^{N_2}} = \frac{\sqrt{2\pi}N^{\frac{1}{2}}\frac{N^N}{e^N}}{\sqrt{2\pi}N_1^{\frac{1}{2}}\frac{{N_1}^{N_1}}{e^{N_1}}\sqrt{2\pi}N_2^{\frac{1}{2}}\frac{N_2^{N_2}}{e^{N_2}}} = \frac{1}{\sqrt{2\pi}}\frac{N^{\frac{1}{2}}\frac{N^N}{e^N}}{\frac{{N_1}^{N_1+\frac{1}{2}}{N_2}^{N_2+\frac{1}{2}}}{e^{N_1+N_2}}} \\ &\propto \frac{N^{N+\frac{1}{2}}}{{N_1}^{N_1+\frac{1}{2}}{N_2}^{N_2+\frac{1}{2}}}\end{aligned} \tag{43}$$

Assuming communities 1 and 2 have the same size, (43) turns:

$$\frac{N^{N+\frac{1}{2}}}{\left(\frac{N}{2}\right)^{\frac{N+1}{2}}\left(\frac{N}{2}\right)^{\frac{N+1}{2}}} = \frac{N^{N+\frac{1}{2}}}{\left(\frac{N}{2}\right)^{N+1}} = \frac{1}{\frac{\left(\frac{N}{2}\right)^{\frac{1}{2}}}{2^{N+\frac{1}{2}}}} = \frac{2^{N+1}}{\sqrt{N}} = e^{ln\left(\frac{2^{N+1}}{\sqrt{N}}\right)} = e^{(N+1)\,ln(2)-\frac{1}{2}ln(N)} \qquad (44)$$

(44) shows the number of partitions grows exponentially with N.

Community Finding

Before presenting community finding algorithms, it is reasonable to ask how feasible it would be to use a brute force approach to inspect all partitions of a network. The total number of partitions in a network with N nodes is given by the Bell Number [2]:

$$B_N = \frac{1}{e}\sum_{j=0}^{\infty}\frac{j^N}{j!} \qquad (45)$$

B_N increases faster than exponentially with N (Figure 7).

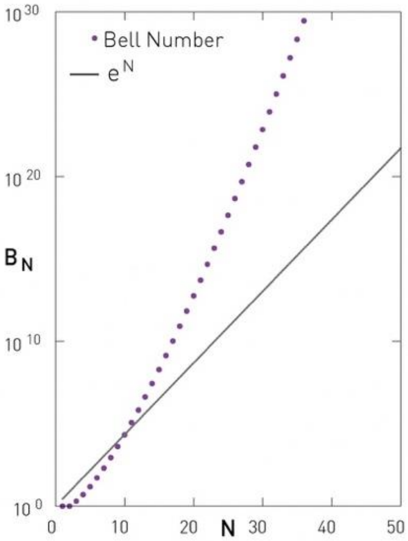


Figure 7 Evolution of the number of partitions with the size of the network [2].

Community finding algorithms executing in polynomial time are crucial to analyze a common size network. Nevertheless, for large networks, linear time is the ideal. Algorithms identifying separate and overlapped community structures are presented in the next section.

2.2.3 Algorithms

After describing the major challenges of community finding over graph partitioning – no information about the size or number of clusters, this section provides an overview of a representative set of algorithms, spanning the most common categories. It is provided not only detailed information regarding their execution, but also time complexity.

Community finding algorithms used in the thesis are detailed in the Methodology section, along with the respective pseudocode that resulted from their implementation in the web application.

2.2.3.1 Hierarchical Clustering

This category is subdivided in agglomerative and divisive procedures: Ravasz [21] and Girvan-Newman [23, 22] algorithms, respectively.

Ravasz Algorithm

It is divided in 4 sequential steps:

1. Definition of a Similarity Matrix

Matrix entries can represent evolutionary distances between nodes or the number of neighbors a pair of nodes has in common. In the second case, each entry is defined as:

$$x_{ij}^{o} = \frac{J(i,j)}{min\left(k_i, k_j\right) + 1 - \Theta(A_{ij})} \tag{46}$$

This implies $x_{ij}^{o} = 0$ when nodes i and j do not have neighbors in common:

$$J(i,j) = 0 \Rightarrow x_{ij}^{o} = 0 \tag{47}$$

The maximum value is obtained when both nodes are connected and have the same neighbors:

$$J(i,j) = min\left(k_i, k_j\right) \Rightarrow x_{ij}^{o} = 1 \tag{48}$$

2. Group Similarity Criteria

After joining the most similar nodes, clusters need to be compared with the remaining elements of the network (nodes/clusters). Three clustering approaches can be used:

- Single Linkage: similarity between two groups is equal to the similarity between the most similar elements;
- Complete Linkage: analogous to the previous measure but using as reference the most dissimilar nodes from each cluster;
- Average Linkage: considers the average distance of every possible pair combination in the 2 clusters.

3. Hierarchical Clustering Procedure

Having defined a similarity matrix and a similarity criterion to compare clusters, the following steps are executed:

- Assign a similarity value to every pair of nodes in the network;
- Identify the most similar community/node pair and join both. Similarity matrix is updated based on group similarity criteria;
- Second step is repeated until all nodes are in the same community.

4. Dendrogram Cut

At the end of the execution, a single tree joining all nodes is obtained – dendrogram. Although it is possible to identify the most similar nodes, it does not return the best partition of the network. In fact, the dendrogram can be cut in one out of several levels. To solve this, modularity is calculated for each partition and the one with the highest value is chosen.

Combining the four steps, the complexity of the algorithm is estimated:

- Step 1: similarity between every pair of nodes is calculated. Complexity should be $O(N^2)$, being N the number of elements in the network;
- Step 2: each community is compared against the others. This requires $O(N^2)$ calculations;
- Steps 3 and 4: using a convenient structure to represent data, in the worst-case scenario, the dendrogram can be built in $O(NlogN)$ steps.

$$O(N^2) + O(N^2) + O(NlogN) = O(N^2) \tag{49}$$

Despite this algorithm is slower than some presented in the next sections, it is significantly faster than the brute force approach which requires $O(e^N)$ operations.

Girvan-Newman Algorithm

Instead of connecting nodes based on similarity criteria, the algorithm developed by Michelle Girvan and Mark Newman removes edges based on centrality criteria. This is repeated until none remains.

1. Defining Centrality

GN algorithm identifies the pair of nodes that most likely belong to different communities and removes the link connecting them. Centrality matrix needs to be recalculated after each removal. Each entry is calculated using 2 alternative approaches:

- Link Betweenness: it is proportional to the number of shortest-paths connecting all pairs of nodes that cross the respective link. Complexity is $O(LN)$ or $O(N^2)$, in sparse networks.
- Random-Walk Betweenness: after picking random nodes m and n, a random path between those is traced. Doing it for every combination of nodes, the average number of times the link (i, j) is crossed is recorded. x_{ij} is proportional to this value. The first step of this calculation requires the inversion of a $N \times N$ matrix, thus computational complexity is $O(N^3)$. The average flowing over all pairs of nodes requires $O(LN^2)$ steps. In the case of a sparse network, the overall complexity is $O(N^3)$.

2. Hierarchical Clustering Procedure

After choosing one of the two centrality criteria:

- Calculate the centrality for every pair of nodes;
- Link with the highest centrality is removed from the network. In case of a tie, one is randomly picked;
- Centrality matrix is updated;
- Two previous steps are repeated until any link is left in the network.

3. Dendrogram

Similarly to the Ravasz, Girvan-Newman algorithm does not predict the best partition. Again, modularity is used to determine the optimal cut in the dendrogram.

Regarding the complexity of the algorithm, the limiting step is the centrality calculation. If link betweenness is chosen, the complexity is $O(LN)$. The overall complexity is obtained by multiplying the previous by the number of times the centrality matrix has to be calculated – L (until all links are removed). This means $O(L^2N)$ or $O(N^3)$ (sparse network) is the final complexity.

Respecting Ravasz and GN algorithms, it is important to ask whether hierarchical structure is really present in real networks or if the algorithms are imposing it. Are there nested modules inside bigger ones? Is it possible to assess, *a priori*, if a network has this structure?

One way to check whether hierarchical modularity is present is by analyzing the clustering coefficient:

$$C(k) \sim k^{-1} \tag{50}$$

This dependence with the node's degree lets us identify whether such pattern is present. In many real networks this phenomenon is present: scientific collaboration, metabolic and citation networks. As expected, under degree-preserved randomization, community structure disappears. Resembling Erdős-Rényi random networks, where these structures are not present.

2.2.3.2 Modularity Maximization

Based on the hypothesis a random network does not have community structure, the local modularity concept was formulated [2]. It compares the partition of a given network with the analogous degree-preserved randomization.

Considering a network with N nodes and L links, and a partition of it with n_c communities, each with N_c nodes and L_c links:

$$c = 1, \ldots, n_c \tag{51}$$

Local modularity compares the number of links between the real network wiring of a subgraph C_c and the randomly rewired subgraph:

$$M_c = \frac{1}{2L} \sum_{(i,j) \in C_c} (A_{ij} - p_{ij}) \tag{52}$$

p_{ij} results from randomly rewiring the whole network, maintaining the expected degree of each node:

$$p_{ij} = \frac{k_i k_j}{2L} \tag{53}$$

If $M_c = 0$, then the subgraph is randomly wired (thus, no community structure). If $M_c < 0$, C_c has less edges than expected and should not be considered a community.

Manipulating (52), a simplified formula is obtained:

$$M_c = \frac{L_c}{L} - \left(\frac{k_c}{2L}\right)^2 \tag{54}$$

Generalizing local modularity to the whole network, the best partition of the graph is identified. This way, network's modularity becomes the sum over all communities of (54):

$$M = \sum_{c=1}^{n_c} \left[\frac{L_c}{L} - \left(\frac{k_c}{2L} \right)^2 \right] \tag{55}$$

Similarly, the higher the modularity, the higher the quality of the partition of the network. It can take positive, null or negative values. Whenever the whole network is defined as one community:

$$(k_c = 2L \wedge L_c = L) \Longrightarrow M = 0 \tag{56}$$

In the extreme case each node is a single community, $L_c = 0$ and the modularity becomes negative. Consequently, none of the previous structures can be classified as a community.

Several algorithms use modularity to partition a network.

Greedy Algorithm

Greedy algorithm maximizes modularity at each step [24]:

1. At the beginning, each node belongs to a different community;
2. The pair of nodes/communities that, joined, increase modularity the most, become part of the same community. Modularity is calculated for the full network;
3. Step 2 is executed until one community remains;
4. Network partition with the higher modularity is chosen.

In terms of computational complexity, since modularity variation can be calculated in constant time, step 2 requires $O(L)$ calculations. After merging communities, the adjacency matrix of the network is updated in a worst-case scenario of $O(N)$. Each merging event is repeated N-1 times. Thus, overall complexity is: $O\big((L+N)N\big)$ or $O(N^2)$, in a sparse graph.

Although modularity is computationally suited and an accurate way of community detection, two pitfalls should be highlighted: resolution limit and modularity maxima.

Resolution Limit

First, a way to calculate network's modularity variation when communities A and B are merged is introduced [2]:

$$\Delta M_{AB} = \frac{l_{AB}}{L} - \frac{k_A k_B}{2L^2} \tag{57}$$

This means two communities should be joined whenever:

$$\Delta M_{AB} > 0 \Leftrightarrow \frac{l_{AB}}{L} > \frac{k_A k_B}{2L^2} \tag{58}$$

Assuming there is one link between communities A and B:

$$1 > \frac{k_A k_B}{2L} \tag{59}$$

In other words, when communities A and B are connected even only by one link, they will be merged if their size is lower than a threshold. Becoming impossible to detect communities below a certain size.

Assuming,

$$k_A \approx k_B = k \tag{60}$$

$$1 > \frac{k^2}{2L} \Leftrightarrow k < \sqrt{2L} \tag{61}$$

The resolution limit depends on the size of the network. One way to overcome this limitation is by subdividing larger communities into smaller ones and partition them.

Modularity Maxima

The fourth hypothesis discussed in *Community Hypothesis*, rely on the assumption that higher modularity implies a better partition of the network. Although, in some graphs, significantly different partitions may have similar modularity. This becomes a relevant issue when the number of nodes in the network increases. Becoming harder to separate network's best partition from the lower quality ones. In the case of the algorithms that optimize modularity this is a central issue, once they iterate until its variation is lower than an input threshold.

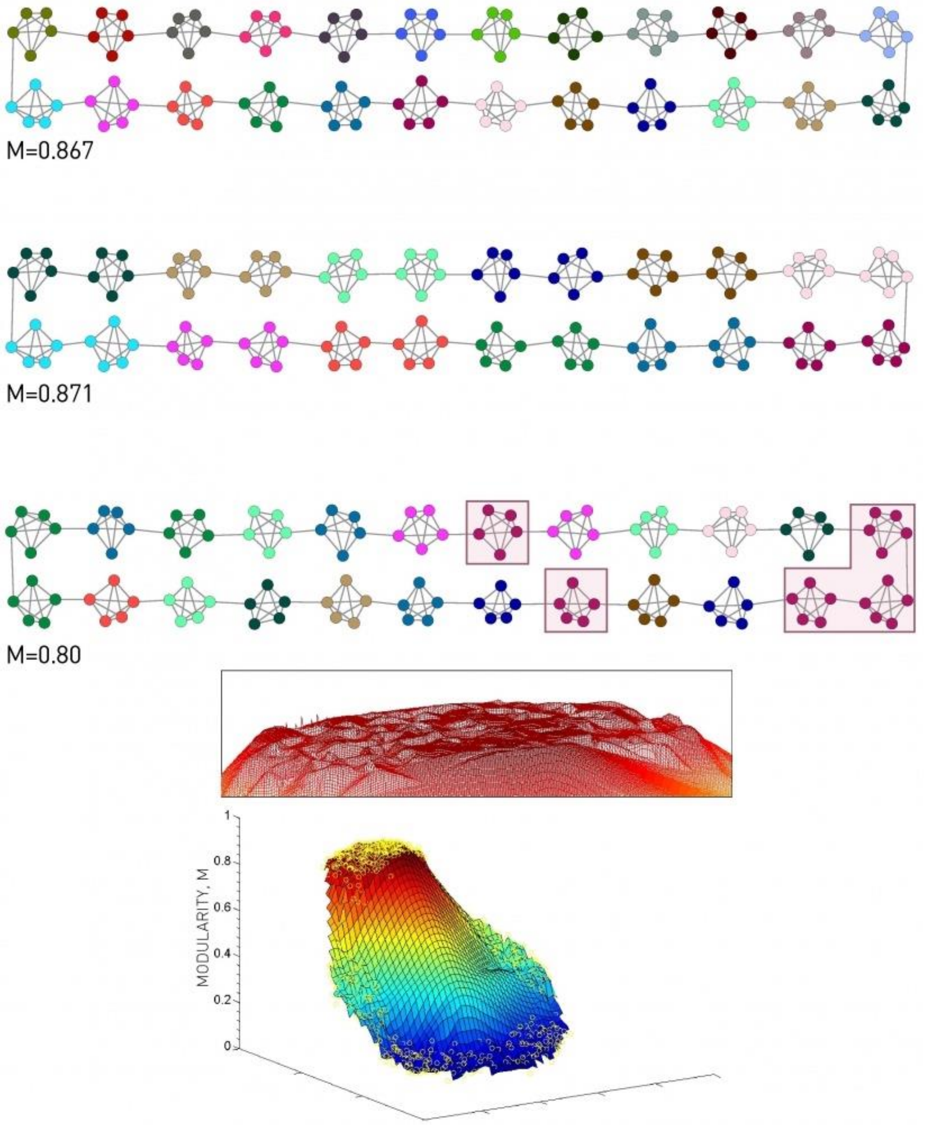


Figure 8 Significantly different partitions of the same network can have similar modularity [2].

Upon analyzes of the network in Figure 8, the number of links inside each cluster is approximated to:

$$k_c \approx \frac{2L}{n_c} \tag{62}$$

Considering $k_A = k_B = k_c$ and applying (57) to the previous network, modularity variation is calculated in terms of the number of nodes n_c:

$$\Delta M_{AB} = \frac{l_{AB}}{L} - \frac{k_A k_B}{2L^2} = \frac{l_{AB}}{L} - \frac{2}{n_c^2} \tag{63}$$

Merging two random clusters of the previous network in the same community will decrease modularity at most $\frac{2}{n_c^2}$. In the limit, this variation is undetectable:

$$\lim_{n_c \to \infty} \frac{2}{n_c^2} = 0 \tag{64}$$

Empirically, the best partition should be the one that groups each 5-node cluster in different communities. Although, modularity increases by 0.003 whenever two adjacent communities are merged. Moreover, if random 5-node clusters are assigned to a community, even if they are not directly connected, it results in a modularity variation close to 0 around the one detected for the optimal partition. This complies with the vision of the plateau in the modularity graph that may distort the choice of the best partition in Figure 8. This plateau explains why a large number of modularity maximization algorithms can quickly detect high modularity partitions – they are not unique.

Modularity optimization algorithms are part of a larger set of problems that are solved by optimizing a quality function. Next, clique-based and statistical inference algorithms are presented.

2.2.3.3 Clique-Based Methods

Some nodes, in real networks, are part of multiple communities. Consequently, they cannot be correctly categorized in only one. This led to the arousal of two algorithms: *clique percolation* [25] and *link clustering* [32].

Clique Percolation Algorithm

Also called *CFinder*, it identifies communities as sets of adjacent cliques. Proceeding the following way:

1. An $N_{clique} \times N_{clique}$ matrix is created. Each entry O_{ij} is the number of shared nodes between i and j k-cliques. k is picked such that we get the richest community structure (with widely distributed cluster sizes);
2. If the number of shared nodes between 2 cliques is $k - 1$, then they are joint. This step is performed in the whole network;
3. Step 2 is performed until no more adjacent cliques are possible to aggregate to the same community.

It is important to state that even random networks may contain big connected components detectable by clique-based algorithms. To assess if a community is present by chance, the analysis of the randomly rewired network should be done.

$$p_c(k) = \frac{1}{[(k-1)N]^{\frac{1}{k-1}}} \tag{65}$$

Whenever the probability of connection between nodes exceeds p_c (which varies with the size of the network and of the clique), it is expected to observe a big component.

Finding cliques in a network is a task that grows exponentially with its size. Although, once the goal is not finding maximum cliques, but k-cliques, they can be identified in polynomial time. However, if large cliques are expected to be present, $O(e^N)$ algorithms may perform better.

Link Clustering Algorithm

Instead of clustering nodes like Ravasz algorithm, Link Clustering uses hierarchical clustering to partition the network in communities accordingly to the similarity between edges. This way, when nodes can belong to more than one community, edges are restricted to a single one. This property is usually verified in real networks.

Before introducing the algorithm, a measure of proximity between pairs of links is defined. Each entry of the similarity matrix is calculated the following way:

$$S\big((i,k),(j,k)\big) = \frac{|n_+(i) \cap n_+(j)|}{|n_+(i) \cup n_+(j)|} \tag{66}$$

$n_+(i)$ represents the number of nodes to which i is connected (including itself). S is the fraction of nodes shared by the 2 non-common nodes present in each link, among all the possibilities.

The overall procedure can be resumed as:

1. Calculation of a similarity matrix;
2. Cluster links with the highest similarity;
3. Apply single-linkage and merge communities;
4. Cut the dendrogram using, for instance, modularity.

The algorithm is divided in 2 steps. First, the calculation of a similarity matrix and, second, hierarchical clustering. Each contributes differently to the overall complexity:

- The first is dependent on the number of links present in the nodes i and j of the pairs being compared. This requires $max(k_i, k_j)$ steps. In case of a scale-free network, the complexity is $O(N^{\frac{2}{\gamma-1}})$. This estimate is determined by the size of the largest node $k_{\max}$;
- Hierarchical clustering step does not run in $O(N^2)$ (like in the Ravasz algorithm), but in $O(L^2)$ because links are being compared instead of nodes. In sparse graphs, it can be approximated to $O(N^2)$.

From the combination of the 2 previous steps, final complexity is:

$$O\left(N^{\frac{2}{\gamma-1}}\right) + O(N^2) = O(N^2) \tag{67}$$

2.2.3.4 Statistical Inference

Three approaches for community finding were already described: hierarchical clustering, modularity maximization and clique-based methods. Statistical inference is the last class of algorithms still not presented. Similarly to clique-based methods, it allows community overlapping.

Yang et al [33] proposed in 2012 a new model-based approach called Community-Affiliation Graph Model (AGM). They defined a new way of representing network's structure by relating each node to the community they belong. Each community has an associated probability of two random contained nodes being connected. Based on this model, they proposed a community finding algorithm.

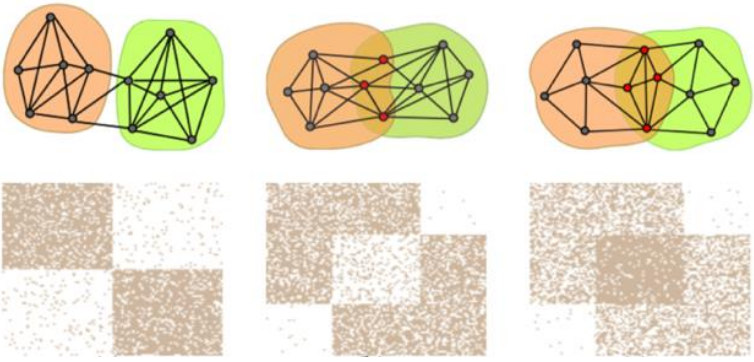


Figure 9 Networks (top) and respective adjacency matrices (bottom) [33].

Community-Affiliation Graph Model

It distinguishes of any other by considering community overlapping and assuming that nodes in those areas, are more likely to be connected (Figure 9). This is an intuitive consequence of their assumption: the probability of 2 nodes being connected is equal to the sum of the probabilities of the communities they share. Before going through the mathematical basis, critical notation is introduced:

Let $B(V, C, M)$ be the bipartite graph that associates each node to the set of communities it belongs. V is the set of nodes, C contains all the communities and M englobes all the edges present in the model, such as: $(u, c) \in M$. u belongs to V and c to the set of communities C. Finally, given $B(V, C, M)$ and $\{p_c\}$, the Community-Affiliation Graph Model generates a graph $G(V, E)$ by creating an edge between u and v, with probabilities $p(u, v)$ (Figure 10). This probability is defined as:

$$p(u, v) = 1 - \sum_{k \in C_{uv}} (1 - p_k) \tag{68}$$

As predicted, $p(u, v)$ changes with the number of communities the nodes share. As higher C_{uv}, the lower $\sum_{k \in C_{uv}} (1 - p_k)$ becomes and closer to 1 $p(u, v)$ gets. $C_{uv} \subset C$ and C_{uv} is the set of communities both nodes share:

$$C_{uv} = \{c | (u, c), (v, c) \in M\} \tag{69}$$

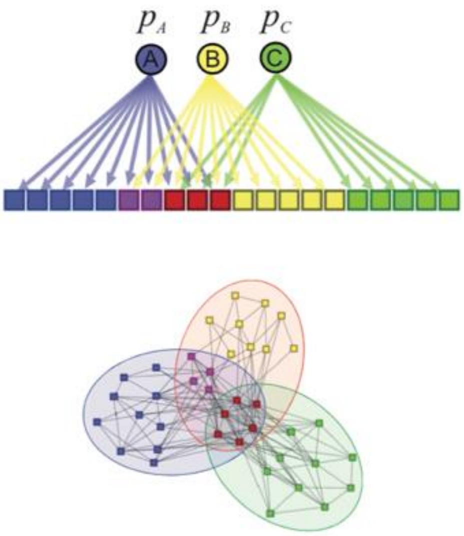


Figure 10 Community Affiliation Model (top) and the original network (bottom) [33].

Algorithms discussed in the previous sections, based on the underlying assumptions, would more likely identify a community in overlapping regions or join together both clusters as a single community [2].

As a consequence of the flexibility of the model, it is possible to represent a wide number of network topologies, with non-overlapping, overlapping and nested communities (Figure 11).

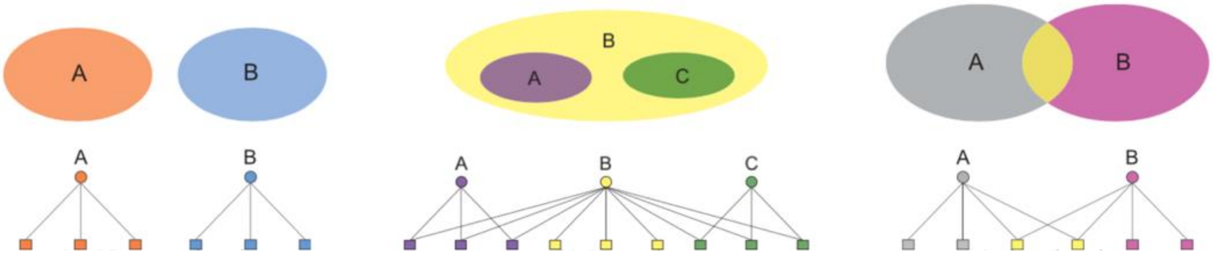


Figure 11 AGM is able to represent non-overlapped (left), nested (middle) and overlapped networks (right) [33].

Community Finding Algorithm

The goal is the find the best network partition by maximizing the likelihood $L(B,\{p_c\}) = p(G|B,\{p_c\})$ of the underlying graph G:

$$arg\max_{B,\{p_c\}} L(B,\{p_c\}) = \prod_{(u,v)\in E} p(u,v) \prod_{(u,v)\notin E} \big(1 - p(u,v)\big) \tag{70}$$

The problem is solved by employing coordinate ascent strategy, iterating over B and $\{p_c\}$. This way, when one of the variables is updated, the other is kept fixed.

While keeping B fixed and updating $\{p_c\}$, the following optimization problem is solved (considering $0 \le p_c \le 1$):

$$arg\max_{\{p_c\}} \prod_{(u,v)\in E} \left(1 - \prod_{k\in C_{uv}} (1-p_k)\right) \prod_{(u,v)\notin E} \left(\prod_{k\in C_{uv}} (1-p_k)\right) \tag{71}$$

Although this problem is not convex, it can be made convex by taking the logarithm of the likelihood and proceeding to the following variable substitution:

$$e^{-x_k} = 1 - p_k \tag{72}$$

The likelihood maximization problem has now a different shape:

$$\underset{\{x_c\}}{arg\,max} \sum_{(u,v)\in E} log(1 - e^{-\sum_{k\in C_{uv}} x_k}) - \sum_{(u,v)\notin E} \sum_{k\in C_{uv}} x_k \tag{73}$$

The resulting variable is constrained to:

$$x_c \geq 0 \tag{74}$$

A globally optimal solution can now be found applying Newton's method or gradient descent.

The last part of the optimization process focuses in finding B that maximizes the likelihood of the model. A different iterative approach is used, once B is not a quantifiable variable like $\{p_c\}$. A method called Metropolis-Hastings is used [34]. Stochastically, it updates B, whenever a certain transition maximizes the likelihood's ratio:

$$max\left(1, \frac{L(B', \{p_c\})}{L(B, \{p_c\})}\right) \tag{75}$$

Metropolis-Hastings randomly updates the bigraph using the following transitions (Figure 12):

- Leave – Considering Figure 12, node u is removed from the actual community. The final set of edges is: $M' = M\backslash\{(u,c)\}$. Assuming, $\{(u,c)\} \in M$;
- Join – Node u joins a community c. Updated version of M is: $M' = M \cup \{(u,c)\}$. Assuming, $\{(u,c)\} \notin M$;
- Switch – 1 node switches communities. This can be described as: $M' = (M\backslash\{(u,c_1)\}) \cup \{(u,c_2)\}$. Assuming, $\{(u,c_1)\} \in M$ and $\{(u,c_2)\} \notin M$.

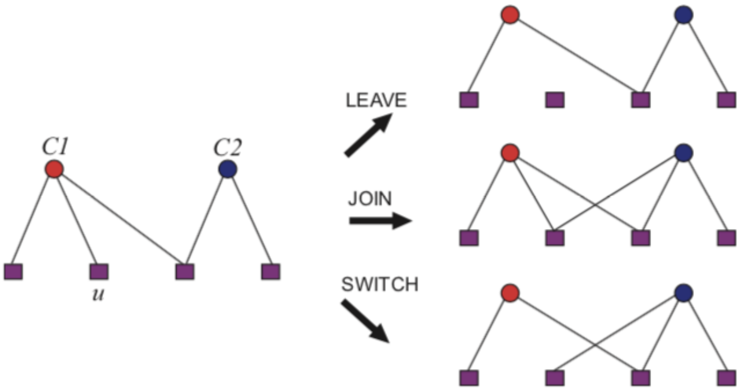


Figure 12 Metropolis-Hastings procedure to find the bigraph that maximizes the likelihood of the model [33].

Resuming, a random B is generated at first place. In each iteration i, the transition is validated ($B_{i+1} = B_i'$) based on the ratio of log-likelihoods. In case the condition is not satisfied, then no update is made ($B_{i+1} = B_i$). Being B_i' the bigraph after applying Metropolis-Hastings procedure at iterate i.

Regarding the computational complexity of AGM algorithm, it is not indicated for networks with millions of nodes. The time complexity of the algorithm is $O(N^2)$ [33]. Being N the number of nodes in the network.

2.2.4 Benchmark

An effective way to benchmark community finding algorithms is by running them in networks whose community structure is known. Different synthetic networks generative algorithms require the user to set different parameters. This, in turn, reflects their ability to approximate real networks. An important property because, at the end, they are intended to perform in real networks, as they do in the synthetic ones.

Community-Affiliation approach is detailed in the next sub-section. As well as different measures to assess algorithm's accuracy, by comparing the detected partitions with the ones present in the synthetic networks. At the end of the section, the speed of several community finding algorithms is compared.

GN and LFR generative methods were used to generate the benchmark networks of the thesis. Normalized Mutual Information (NMI) used to test the accuracy of the implemented algorithms. This way, they are described in the Methodology section.

2.2.4.1 Community-Affiliation Graph Model

AGM can be used as a community finding algorithm, but also as a benchmark network generator [33]. What distinguishes this method from the other two is the ability to generate graphs accounting to community overlap.

The user starts by deciding the number of nodes and communities in the network. Then, the probabilities of two nodes inside a given community being connected by an edge are defined. In order to create an even more realistic network, it can also consider an extra community designated ε-community which adds an extra probability for nodes from different communities to connect:

$$\varepsilon = \frac{2|E|}{|V|(|V|-1)} \tag{76}$$

Based on the parameters of the model, a synthetic network is generated.

Measures to assess algorithm's partitions correctness are introduced next.

2.2.4.2 Accuracy

Normalized Mutual Information [35], Average F1 Score [33], Omega Index [36] and Accuracy in the Number of Communities [33] are commonly used to test the accuracy of community finding algorithms respecting the ground-truth communities present in the network. The first method is described in the Methodology section and the last three are below.

Average F_1 Score

F_1 score is defined as the average of the F_1 score of the best matching ground-truth community for each detected one, and the F_1 score of the best estimated against each ground-truth community. This can be mathematically defined as:

$$F_1 = \frac{1}{2}(\bar{F}_g + \bar{F}_d) \tag{77}$$

Where:

$$\bar{F}_g = \frac{1}{|C^*|}\sum_i max_j F_1(C_i, \hat{C}_j) \tag{78}$$

$$\bar{F}_d = \frac{1}{|\hat{C}|}\sum_j max_j F_1(C_j, \hat{C}_i) \tag{79}$$

And $F_1(C_i, \hat{C}_j)$ is the harmonic mean of precision and recall of C_i and $\hat{C}_j$:

$$F_1(C_i, \hat{C}_j) = \left(\frac{recall^{-1} + precision^{-1}}{2}\right)^{-1} = 2 \times \frac{precision \times recall}{precision + recall} \tag{80}$$

Contingency tables are usually used to represent the number of true/false negatives/positives. From those, precision and recall are calculated. The first measure is given by the ratio of true positives over the sum of true and false positives. Recall is calculated dividing the number of true positives over the sum of those with the false negatives. This statistical definition can be extrapolated to community finding context by considering the community each node is assigned, against its ground-truth.

Omega Index

It estimates the accuracy of the detected number of shared communities by a pair of nodes versus its theoretical value. It is particularly relevant when working with covers in which community overlapping is present.

$$\frac{1}{|V|^2}\sum_{u,v \in V} \mathbf{1}\{|C_{uv}| = |\hat{C}_{uv}|\} \tag{81}$$

C_{uv} and $\hat{C}_{uv}$ are, respectively, the ground-truth and the detected communities in the network. V is the total set of vertices that u and v belong.

Accuracy in the Number of Communities

It determines how the number of detected communities relates to the one found in the ground-truth partition of the network.

$$1 - \frac{||C^*| - |\hat{C}||}{|C^*|} \tag{82}$$

2.2.4.3 Speed

In terms of speed, it is expected the overall time of execution of the algorithms to vary accordingly to the computing machine and size of the network. Nevertheless, computational complexity provides us an upper bound regarding the number of steps each algorithm will have to execute.

Table 2 compares all community finding algorithms in the thesis. Louvain and Infomap algorithms are the fastest. On the opposite side, *CFinder* has the worst performance.

Table 2 Community finding algorithms' time complexity. Some are only introduced in Methodology (*).

Algorithm	Nature	Time Complexity	Reference
Ravasz	Hierarchical Agglomerative	$O(N^2)$	[21]
Girvan-Newman	Hierarchical Divisive	$O(N^2)$	[22, 23]
Greedy Modularity	Modularity Optimization	$O(N^2)$	[24]
Clique Percolation (*CFinder*)	Overlapping Communities	$Exp(N)$	[25]
Link Clustering	Hierarchical Agglomerative Overlapping Communities	$O(N^2)$	[32]
AGM	Statistical Inference Overlapping Communities	$O(N^2)$	[33]
Louvain*	Modularity Optimization	$O(L)$	[37]
Infomap*	Flow Optimization	$O(NlogN)$	[38]
Label Propagation* (LP)	Semi-Supervised	$O(L)$	[39, 40]
Layered Label Propagation*	Semi-Supervised	$O(L)$	[41]

2.3 Phylogenetics

The term phylogeny was used for the first time by Haeckel in 1866 [42]. Phylogenetics aims to trace an evolutionary relationship between taxa [43]. Calling it taxa, we generalize the claim to all taxonomic groups, such as strains, species or populations.

Based on visual representations of descendance, named phylogenetic trees, Charles Darwin was the first to popularize these structures [44]. Regarding its shape, the most recent taxa are represented as leaves, ancestors as internal nodes and a common ancestor in the top of the tree occupying the position of the root.

In the 19th century, similarity between taxa was based only on morphological characteristics. Nowadays, the attempt to classify organisms based on their morphology is called phenetics (still important in cases where taxa's morphology is the only information available, e.g.: fossils) [45]. With the discovery of DNA and the advent of protein and nucleic acids analysis, a variety of sequencing methods was made available. Those led to a boom in the quantity and quality of data which is now available to infer more precise evolutionary relationships. A new field of study emerged – Molecular Phylogenetics [46].

2.3.1 Molecular Phylogenetics

Molecular phylogenetics uses DNA, RNA or protein sequences to infer similarity among taxa. This choice should be made based on the expected evolutionary time separating them. This way, it is used the structure best conserved over time but that still captures evolution. At the end, these evolutionary relationships are intended to be represented as a phylogenetic tree.

2.3.1.1 Sequencing Genomes

Data acquisition for phylogenetic analysis strongly relies on DNA sequencing. Frederick Sanger, a pioneer in protein and DNA sequencing (2 times Nobel Prize winner), developed a sequencing method that due to its reliability and easiness of execution, was used in the 1st generation of DNA sequencers. This would remain the standard sequencing method from the 1980s until middle 2000 [47].

With the development of shotgun sequencing, the development of Next Generation Sequencing alternatives was triggered. This would lead to a row of high-throughput methods which strongly decreased the costs and time of sequencing. This made possible analyzing longer pieces of DNA, but also multiple ones simultaneously. Completely sequenced chromosomes and genomes were the result of NGS methods [47].

All these developments were proved to be crucial in metagenomics, medicine, forensics, molecular and evolutionary biology. Once the microorganisms in the onset of most infections (bacteria and virus) have the shortest genomes, they were the first to be sequenced. This way, microbiology had its role emphasized and its development led to the creation of new labs and genetic databases [48].

Some standard and replicable typing methods are now widely used to infer phylogenetic relations. The most common is MLST. Instead of considering the complete DNA sequence of each strain, 7 housekeeping loci are usually chosen. This selection stems in the assumption that these loci are under moderate to strong purifying selection. Therefore, the resultant sequence variation is mostly neutral and approximately linear with time. This way, genetic distance tends to be proportional to the time of divergence between alleles. Each locus contains around 400-500 nucleotides and belong to a gene, a regulating or a non-coding region. Progressively, more housekeeping loci are being used. Each sequence is then compared to an allele database and a numerical identifier is assigned. The combination of alleles at each locus creates an allelic profile. Each profile is then converted in a Sequence Type (ST) that unambiguously identify each clone. The main advantage of this approach is to buffer the effect of recombination in taxa evolution by considering it a single genetic event, similarly to nucleotide mutation [49].

Core genome MLST (cgMLST), ribosomal MLST (rMLST) and whole genome MLST (wgMLST) and Single Nucleotide Polymorphism (SNP) are some other techniques used to assess taxa similarity [50]. cgMLST combines the discriminatory power of classical MLST with the additional data got from WGS. It considers hundreds of gene targets across the whole genome. Thus, it enhances discriminatory power, particularly relevant in outbreak events [51]. rMLST considers 53 genes encoding the bacterial ribosome protein subunits (*rps* genes). *rps* genes are ideal targets because they are present in all bacteria, distributed around the chromosome and encode proteins that are under stabilizing selection for functional conservation [52]. wgMLST is an approach that is able to capture many types of nucleotide differences for every opening frame of the genome. Thereby, it allows genome-wide comparisons. Another benefit is that by designating alleles for each ORF, it allows to reduce the analysis of millions of pair bases into a single character per gene [53]. Depending on the scope of the analysis and data availability, one inference method can be chosen over the other. Figure 13 highlights the variation of discriminatory power based on the method. For short-term outbreak situations, SNP analysis is usually chosen. It analyzes variations at a single position of the DNA sequence [54].

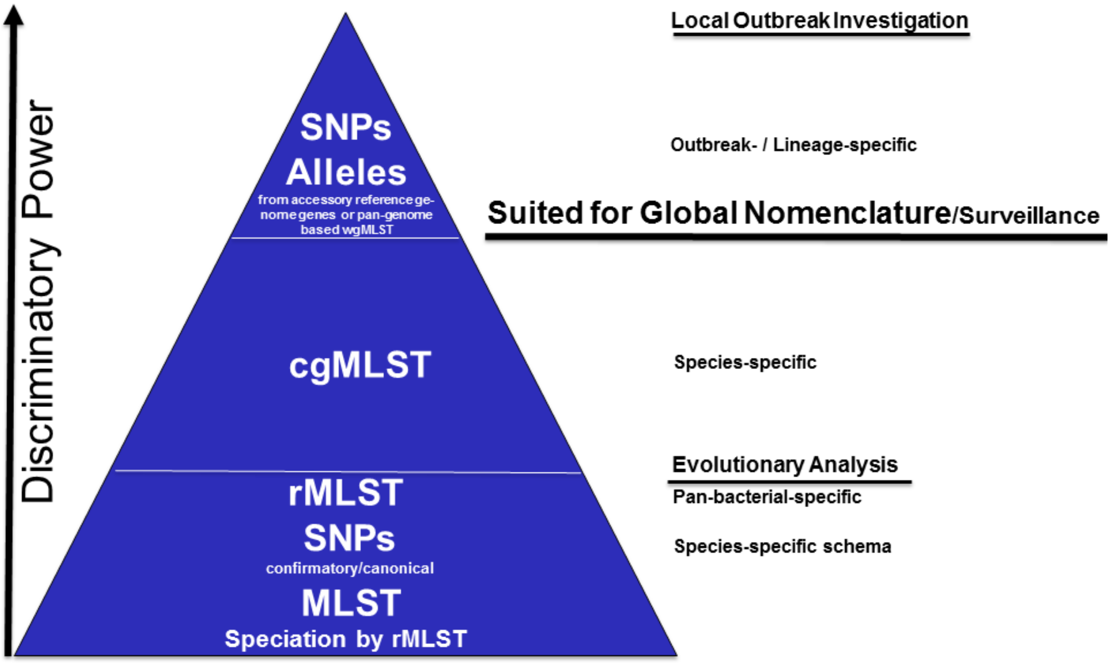


Figure 13 Discriminatory power of different typing methods [50].

2.3.1.2 Tree Building

Two types of reconstruction algorithms for phylogenetic trees are widely used: distance (UPGMA, WPGMA, Single Linkage, Complete Linkage, Neighbor-Joining and Fitch-Margoliash algorithms) and character-based (Maximum Parsimony, Maximum Likelihood and Bayesian Inference) methods [46]. The first calculates the distance between every pair of sequences and the resultant distance matrix is used for tree reconstruction. Character-based approach simultaneously compare all sequences in an alignment considering one character/residue at a time when calculating a score to the inferred tree. In theory, the tree with the best score is the one that best infers an evolutionary relationship among taxa. Many heuristic algorithms have been developed to overcome the unfeasibility of analyzing every possibility. These approaches often generate a starting tree using fast algorithms and optimize it using local arrangement in the attempt of improving tree score.

Phylogenetic trees represent the inferred distances among taxa, considering the ancestors are unknown. Although, in micro-evolution (short-term evolution), such as outbreaks, they are known, and their representation allows a more comprehensive analyzes. Moreover, phylogenetic trees only allow a rigid representation of the relations among nodes. This is particularly important when in the presence of reticulate events: hybridization, horizontal gene transfer, recombination, gene duplication or loss. In order to overcome these problems an alternative representation was proposed – phylogenetic networks.

eBURST is a widely used algorithm to infer phylogenetic networks [55]. goeBURST is a globally optimized version of eBURST that uses Kruskal algorithm to infer these representations (Figure 14) [49].

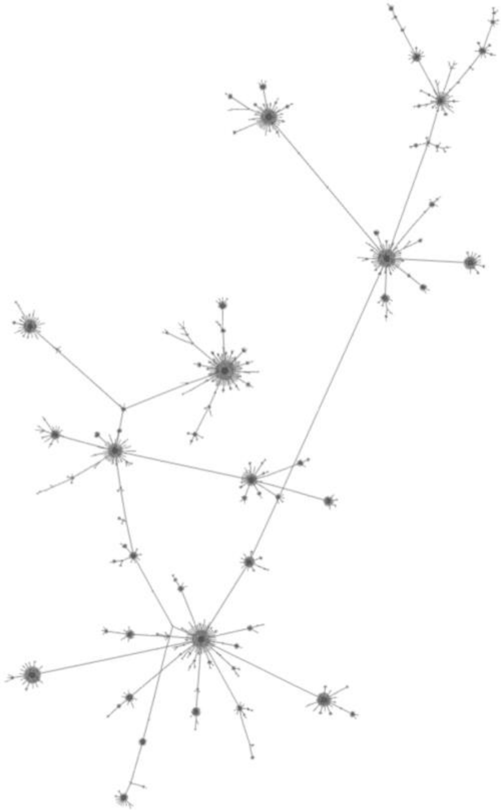


Figure 14 Phylogenetic tree generated using goeBURST implemented in PHYLOViZ Online.

2.3.2 PHYLOViZ

PHYLOViZ offers the user a scalable way of analyzing phylogenetic data. It is available as a cross-platform JAVA software [56] and a web application [57]. The latter includes a RESTful API that allows for programmatic access and analysis of data. This thesis focuses in the online version.

Regarding implementation, PHYLOViZ Online is a Node.js application widely supported by most browsers [57]. It uses goeBURST [49] to build Minimum Spanning Trees for different data inputs. Plus, it allows users to exchange visual representations of the generated data and provides access to a public database with hundreds of microbial genetic profiles and respective auxiliary data. Along with this user-friendly application, the RESTful API allows to retrieve, upload data and run available tree algorithms, from any browser or device.

Due to the number of databases and sequencing data formats available, it is fundamental to allow the input of the broadest range. PHYLOViZ Online supports [57]:

- A tab-delimited file format for MLST, cgMLST, wgMLST, multiple locus variable-number tandem repeat (MLVA) and SNP analyses;
- FASTA files with sequences of the same length or aligned to the same length;
- Newick files already containing the tree topology and branch lengths.

The user can also upload auxiliary data in a tab-delimited format, such as epidemiological, demographic, temporal or other metadata about the strain such as: antibiotic resistance, serotype, or other strain characterization or typing method result.

3. Methodology

Introduction covered all theoretical concepts important for this section. Although some community finding algorithms, benchmark network generator algorithms, real networks and visualization tools were presented before, the ones used in the thesis are introduced in the next sub-sections. Before presenting the methods, it is important to situate them in the state of the art. This way, research and review components are now identified.

Louvain, Infomap and LLP algorithms were presented for the first time by *Blondel et al* in 2008 [37], *Rosvall et al* in 2009 [38] and *Raghavan et al* in 2007 [39], respectively. Several implementations, in Python and C++, were made available online by the authors. The main goal of the thesis was to implement the three algorithms in JavaScript. Louvain's implementation was based on a version available in the NPM (though, it was adapted to convey phylogenetic data analysis, as well as to perform faster than the original one) [58]. Infomap and LLP were implemented in JavaScript. The first was obtained after modifying Louvain so that it could optimize a different quality function. LLP was implemented from zero.

An algorithm to generate GN synthetic networks was developed in JavaScript, in the thesis. The original version was part of a paper published by Girvan and Newman in 2002 [23]. In 2008, Lancichinetti, Fortunato and Radicchi, using C++ programming language, implemented a second version [59]. The same implementation was used to generate a different class of networks called LFR networks.

Amazon, Zachary's Karate Club and *S. aureus* datasets were obtained from *Stanford Large Network Dataset Collection* [60], Zachary's paper [61] and *PubMLST* [62], respectively. For the Zachary's Karate Club network, a JavaScript Object Notation (JSON) file was created in the thesis.

All networks can be visualized with the most up-to-date and stable JavaScript versions of D3 (version 4) and Cytoscape (version 3.2.22). Benchmark data was plotted using D3 (version 4).

JavaScript programming language was repeatedly emphasized before once it is becoming a web standard due to its flexibility, scalability and for being supported by most browsers and devices. Thus, it is fundamental for online data analysis and visualization.

Wrapping it all, an innovative web application was built. It allows the user to explore three community finding algorithms, two synthetic networks, three real networks and to visualize the results using one of three possible big-data friendly frameworks.

After clarifying the review and innovative components of the thesis, the structure of Methodology is outlined: first, community finding algorithms are presented; then, benchmark networks used to assess algorithm's speed and accuracy are introduced; real test networks; a measure to estimate algorithm's accuracy and, finally, different visualization frameworks implemented in the web application are detailed.

3.1 Community Finding

Louvain, Infomap and Layered Label Propagation algorithms implemented in JavaScript have their mathematical description, pseudocode and complexity detailed below.

3.1.1 Louvain Algorithm

Louvain is an unsupervised algorithm (does not require the input of the number of communities nor their sizes before execution) divided in 2 phases: Modularity Optimization and Community Aggregation [37]. After the first step is completed, the second follows. Both will be executed until there are no more changes in the network and maximum modularity is achieved.

$$M = \frac{1}{2m}\sum_{i,j}(A_{ij} - p_{ij})\delta(c_i, c_j) = \frac{1}{2m}\sum_{i,j}(A_{ij} - \frac{k_i k_j}{2m})\delta(c_i, c_j) \tag{83}$$

A_{ij} is the adjacency matrix entry representing the weight of the edge connecting nodes i and j, $k_i = \sum_j A_{ij}$ is the degree of node i, c_i is the community it belongs, δ-function $\delta(u, v)$ is 1 if $u = v$ and 0 otherwise. $m = \frac{1}{2}\sum_{ij} A_{ij}$ is the sum of the weights of all edges in the graph.

Modularity Optimization

Louvain will randomly order all nodes in the network in Modularity Optimization. Then, one by one, it will remove and insert each node in a different community C until no significant increase in modularity (input parameter) is verified:

$$\Delta M = \left[\frac{\Sigma_{in} + 2k_{i,in}}{2m} - \left(\frac{\Sigma_{tot} + k_i}{2m}\right)^2\right] - \left[\frac{\Sigma_{in}}{2m} - \left(\frac{\Sigma_{tot}}{2m}\right)^2 - \left(\frac{k_i}{2m}\right)^2\right] \tag{84}$$

Let Σ_{in} be the sum of the weights of the links inside C, Σ_{tot} the sum of the weights of all links to nodes in C, k_i the sum of the weights of all links incident in node i, $k_{i,in}$ the sum of the weights of links from node i to nodes in the community C and m is the sum of the weights of all edges in the graph.

One way to further improve the performance of the algorithm is by simplifying (84) and calculating ΔMm instead of the complete expression:

$$\Delta M = \frac{k_{i,in}}{m} - \frac{2\sum_{tot} k_i}{(2m)^2} \Leftrightarrow \Delta Mm = k_{i,in} - \frac{\sum_{tot} k_i}{2m} \tag{85}$$

While $k_{i,in}$ and Σ_{tot} need to be calculated for each trial community, $\frac{k_i}{2m}$ is specific of the node that is being analyzed. This way, the latter expression is only recalculated when a different node is considered in Modularity Optimization.

Community Aggregation

After finishing the first step, all nodes belonging to the same community are merged into a single giant node. Links connecting giant nodes are the sum of the ones previously connecting nodes from the same different communities. This step also generates self-loops which are the sum of all links inside a given community, before being collapsed into one node (Figure 15).

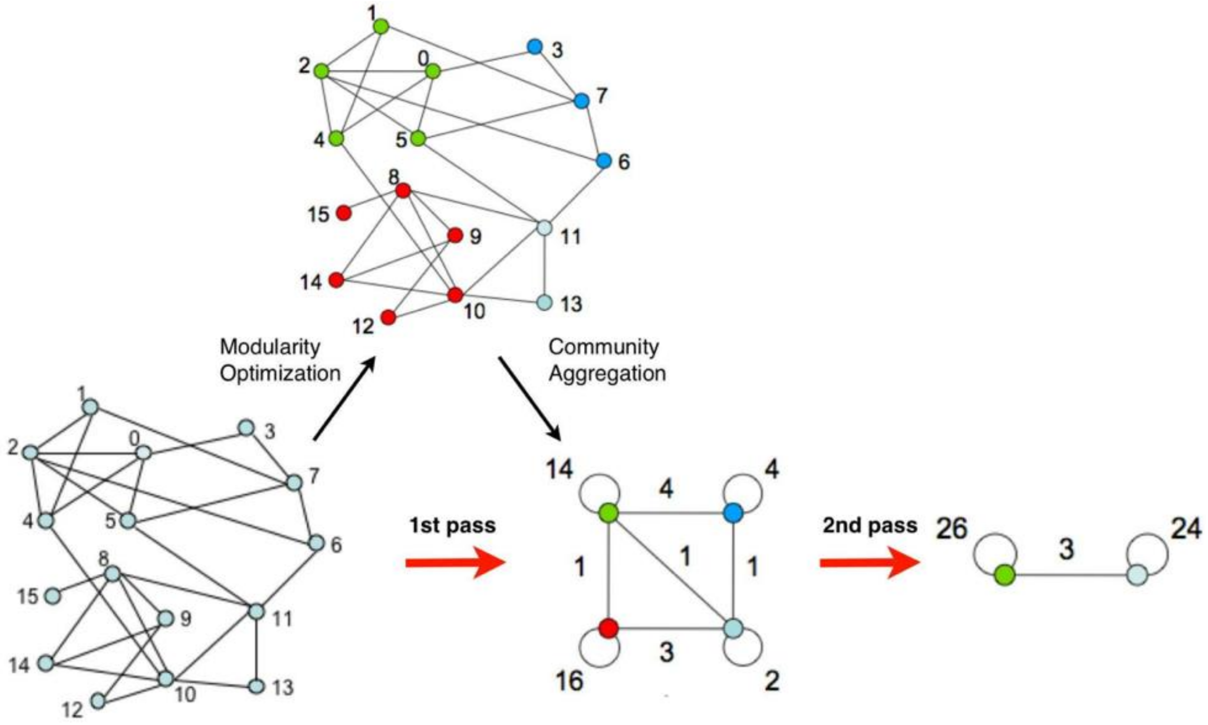


Figure 15 Sequence of steps followed by Louvain algorithm. Adapted from [37].

Thus, by clustering communities of communities after the first pass, it inherently considers the existence of a hierarchical organization in the network (Algorithm 1).

Algorithm 1 Louvain

```
Require:
$G^0 = (V^0, E^0)$: initial undirected graph. $V^0$ is the initial set of vertices. $E^0$ is the initial set of edges;
$\theta$: modularity improvement threshold.
Ensure:
$M$: resulting module;
$Mod$: resulting modularity;
$\delta Mod$: modularity variation using (85);
$k_i$: sum of the weight of all edges connecting node $i$;
$k_{i,j}$: weight of the edge connecting nodes $i$ and $j$.
$m = \sum k_{i,j}, (i,j) \in E^0$
$k = 0$ // Iteration number.
repeat
  // Attributing a different community to each node.
  for all $i \in V^k$ do
    $M_i^k = \{i\}$
  end for
  Compute $Mod_{new} = Mod(M)$ using (83)
  repeat
    $Mod = Mod_{new}$
    Randomize the order of vertices.
    for all $i \in V^k$ do
      $best\_community \ = \ M_i^k$
      $best\_increase \ = \ 0$
      for all $M' \in C^k$ do
        $M_i^k = M_i^k \backslash \{i\}$
        ${\Sigma_{tot}}^{M_i^k} = \sum_{\alpha \in M_i^k} k_\alpha - k_i; \, {\Sigma_{tot}}^{M'^k_i} = \sum_{\alpha \in M'^k_i} k_\alpha + k_i;$
        $\Sigma_{in} = {\Sigma_{tot}}^{M'^k_i} - \sum k_{i,j}\,, (i,j) \in E^k, i \in C_i^k \ and \ j \notin C_i^k$
        $k_{i,in} = \sum_{\alpha \in M'^k_i} k_{i,\alpha}$
        If $\delta Mod_{M_i^k \rightarrow M'^k_i} > best\_increase$ then
          $best_{increase} = \delta Mod_{M_i^k \rightarrow M'^k_i}$
          $best_{com} = M'^k_i$
          $M'^k_i = M'^k_i \cup \{i\}$
        else
        $M_i^k = M_i^k \cup \{i\}$
        end else
      end for
    end for
  Compute $Mod_{new} = Mod(M)$ using (83)
  until No vertex movement or $Mod_{new} - Mod < \theta$
// Calculate updated modularity
Compute $Mod_{new} = Mod(M)$ using (83)
If $Mod_{new} - Mod < \theta$ then
  break;
end if
$Mod = Mod_{new}$
// Merge communities into a new graph
$V^{k+1} \leftarrow C^k$
$E^{k+1} \leftarrow e(C_u^k, C_v^k)$
$G^{k+1} = (V^{k+1}, E^{k+1})$
$k = k + 1$
until break
```

3.1.2 Infomap Algorithm

In spite Louvain and Infomap algorithms optimize a quality function, the way the network is perceived differs. While the first focuses in the structural aspect, Infomap partitioning is based on the flow induced by the pattern of connections [38].

Considering a sender pretends to communicate a random path inside a network to a receiver, the following is assumed:

The size of this message is intended to be minimized. An immediate strategy would be to attribute to each node a different name (code) and send to the receiver the corresponding sequence. The latter would be able to decodify the message based on a proper codebook. Considering the path is described in binary language by N codes of the same size, the minimum length L of each code word is:

$$L = \log_2 N \tag{86}$$

Another way to describe the same path, in a more concise way, is using Huffman coding. This approach consists in associating codes of different lengths to each node, depending on the ergodic node visit frequencies of a random path (the average node visit frequencies of an infinite-length random walk). By itself, Huffman coding is already an efficient way to transmit sporadically or discontinuously separate nodes that constitute the random path we pretend to codify. In fact, Shannon's source coding theorem states that when we use n codewords to describe n states of a random variable X, the minimum description length (MDL) is given by the entropy of the random variable itself:

$$H(X) = -\sum_{1}^{n} p_i log(p_i) \tag{87}$$

It is possible to make the description more efficient if the goal is to transmit the full path or significant sequences of the belonging nodes. To achieve this, the network is partitioned in modules and, for each module, a different code book is defined. This allows different nodes to be given the same identifications. Which not only implies the definition of a module codebook for each module (including an exit code), but also an index codebook to flag anytime a random walker enters in a different module. In the case of these layered descriptions, the MDL is given by a weighted average of the frequency each module and index codebooks are used:

$$L(M) = qH(Q) + \sum_{m=1}^{n_m} p_{\odot}^{m} H(P_m) \tag{88}$$

Being $q = \sum_{j=1}^{m} q_j$ the sum of the exit probability for each community, $H(Q)$ the average code length of the movement between communities, $p_{\odot}^{i} = q_i + \sum_{\beta \in i}^{m} p_\beta$ the stay probability for a random walk in a community c and $H(P_m)$ the average code length of a module codebook m.

$$H(Q) = -\sum_{i=1}^{m} \frac{q_i}{\sum_{j=1}^{m} q_j} log\left(\frac{q_i}{\sum_{j=1}^{m} q_j}\right) \tag{89}$$

$$H(P_m) = -\frac{q_i}{q_i + \sum_{\beta \in i}^{m} p_\beta} \sum_{i=1}^{m} log\left(\frac{q_i}{q_i + \sum_{\beta \in i}^{m} p_\beta}\right) - \sum_{\alpha \in i} \frac{p_\alpha}{q_i + \sum_{\beta \in i}^{m} p_\beta} log\left(\frac{p_\alpha}{q_i + \sum_{\beta \in i}^{m} p_\beta}\right) \tag{90}$$

Inserting (89) and (90) in (88):

$$\begin{aligned} L(M) = &\left(\sum_{m \in M} q_m\right) log\left(\sum_{m \in M} q_m\right) - 2\sum_{m \in M} q_m \, log(q_m) - \sum_{\alpha \in V} p_\alpha \, log(p_\alpha) \\ &+ \sum_{m \in M} (q_m + \sum_{\alpha \in m} p_\alpha) \, log\left(q_m + \sum_{\alpha \in m} p_\alpha\right) \end{aligned} \tag{91}$$

Considering an unweighted and undirected network, q_m is the exit probability of a module m and p_α the relative weight w_α that is computed dividing the total weight of the edges connected to α by twice the total weight of all links in the graph.

This way, by compressing the description length of the network, this is partitioned in modules which reflect the dynamics of the network. This is particularly useful when the connections between nodes represent a flow of a given quantity and not only the similarity between those (Algorithm 2).

Infomap approach clarifies how community finding algorithms can be also used in compression problems.

Algorithm 2 Infomap

```
Require:
$G^0 = (V^0, E^0)$: initial undirected graph. $V^0$ is the initial set of vertices. $E^0$ is the initial set of edges;
$\theta$: quality improvement threshold.
Ensure:
$M$: resulting module;
$L$: resulting MDL;
$\delta L$: MDL variation;
$w_u$: sum of the weight of all edges connecting node $u$;
$w_{u,v}$: weight of the edge connecting nodes $u$ and $v$.
$w = 0$ // Iteration number.
repeat
  for all $u \in V^k$ do
    $p_u = \frac{degree(u)}{|E^k|}$
  end for
// Attributing a different community to each node.
  for all $u \in V^k$ do
    $M_u^k = \{u\}$
    $p^{M_u^k} = \sum_{\alpha \in M_u^k} p_\alpha$
    $q^{M_u^k} = \sum w_{u,v}, (u,v) \in E^k, u \in C_u^k \; and \; v \notin C_u^k$
  end for
  Compute $L_{new} = L(M)$ using (91)
  Repeat
    $L = L_{new}$
    Randomize the order of vertices.
    for all $u \in V^k$ do
      If $M'^k_u = argmin\left(\delta L_{M_u^k \rightarrow M'^k_u}\right) < 0$ then
        $M_u^k = M_u^k \backslash \{u\}$; $M'^k_u = M'^k_u \cup \{u\}$
        $p^{M_u^k} = \sum_{\alpha \in M_u^k} p_\alpha - p_u$; $p^{M'^k_u} = \sum_{\alpha \in M'^k_u} p_\alpha + p_u$
        update $q^{M_u^k}$; update $q^{M'^k_u}$
      end if
    end for
    Compute $L_{new} = L(M)$ using (91)
  until No vertex movement or $L - L_{new} < \theta$
  // Calculate updated MDL.
  Compute $L_{new} = L(M)$ using (91)
```

```
    If $L - L_{new} < \theta$ then
        break;
    end if
    $L = L_{new}$
    // Merge communities into a new graph
    $V^{k+1} \leftarrow C^k$
    $E^{k+1} \leftarrow e(C_u^k, C_v^k)$
    $G^{k+1} = (V^{k+1}, E^{k+1})$
    $k = k + 1$
until break
```

3.1.3 Layered Label Propagation Algorithm

Layered Label Propagation algorithm [41] development was strongly based in the older Label Propagation [39]. The latter starts by assigning a different community to each node present in the network. Then, at the beginning of each round, the order of the nodes is randomized. In each round, every node will be assigned the community more represented among the immediate neighbors. LP finishes as soon as no changing in communities happens or after reaching a given number of iterations. It does not depend on the optimization of any objective function and it does not require prior information regarding the communities present in the network (unsupervised). Although, it can be input (semi-supervised) with an approximate set. By definition, this algorithm is inherently local, (approximately) linear in the number of edges and requires few passes in the graph. These are fundamental features to allow the analysis on real networks. The problem of finding an optimal partition has been proven to be equivalent to find local minima of the Hamiltonian for a kinetic Potts model [41]. Although the problem has a trivial globally optimal solution that considers the whole network as a single cluster, it is not of our interest. Label Propagation avoids such solution because of the inherent dynamics of local search. While this is normally considered a drawback, in this case it is a fundamental feature of LP algorithm to work.

In spite of using a similar approach, LLP not only takes into account the nodes in the neighborhood, but also the ones in the remaining network. The iterative process is the same, the difference resides in the value it optimizes. In this case, the assigned community will be the one that maximizes:

$$k_i - \gamma(v_i - k_i) \tag{92}$$

Being k_i the number of nodes with label λ_i in the neighborhood of a given node and v_i the total number in the whole graph labeled the same way. γ is the resolution parameter that buffers the number of nodes belonging to a given community in the neighborhood, by its usual presence across the whole network. Whenever it approaches 0, LLP reduces to LP. It is expected that for low values of γ, the algorithm will highlight few, big and sparse clusters. For higher values, clusters become smaller, denser and increase their number. It is relevant to say that there is no *a priori* measure to state which value could be better for the network under analysis (Algorithm 3).

Similarly to LLP, other algorithms have been proposed that execute the same sequence of steps but maximizing different values like modularity.

In all the previous community finding algorithms, parallel implementations, making using of multiple processors, can significantly speed up their execution. Nevertheless, this will not be considered along the thesis.

Algorithm 3 Layered Label Propagation

```
Require:
$G^0 = (V^0, E^0)$: initial undirected graph. $V^0$ is the initial set of vertices. $E^0$ is the initial set of edges;
$max_{iteration}$: maximum iteration number;
$\gamma$: resolution parameter.
Ensure:
$M$: resulting module;
$k_i$: number of neighbors having label $\lambda_i$;
$v_i$: overall number of nodes having label $\lambda_i$.
$k = 0$ // Iteration number.
// Attributing a different community to each node.
for all $i \in V^0$ do
  $M_i^0 = \{i\}$
end for
repeat
  Randomize the order of vertices.
  for all $u \in V^k$ do
    for all $i \in labels(u)$ do
      If $k_i(u) - \gamma(v_i(u) - k_i(u)) > k_{i-1}(u) - \gamma(v_{i-1}(u) - k_{i-1}(u))$ then
        $M_u^k = M_u^k \backslash \{u\}$; $M'^k_u = M'^k_u \cup \{u\}$
      end if
    end for
  end for
$k = k + 1$
until No vertex movement or $k < max_{iteration}$
```

3.2 Benchmark

Although a network structure can be completely defined by its wiring diagram, different community finding algorithms (based on distinct assumptions) part the same graph in different ways. Consequently, it is important to understand which algorithms are the most indicated.

An accurate way to gain this insight is by partitioning networks whose community structure is known. GN and LFR benchmark networks are presented in the next sub-sections, as well as NMI which estimates the similarity between the original and the detected partitions.

Community finding algorithms were not only tested in terms of accuracy, but also speed. All benchmark tests were repeated 10 times. The mean values and corresponding confidence intervals (CI) at 95% were included in Results.

3.2.1 Girvan-Newman Network

Historically, GN network was the first synthetic benchmark network to be designed. In their paper, Girvan and Newman described it as a set of 128 nodes divided in 4 groups, each one with exactly 32 nodes [23]. Then, a mixing parameter, chosen by the user, defines the probability of nodes from different communities being connected. This probability is given by:

$$\mu = \frac{k^{ext}}{k^{int} + k^{ext}} \tag{93}$$

Moreover, each node is connected to, exactly, 16 different ones (Figure 16).

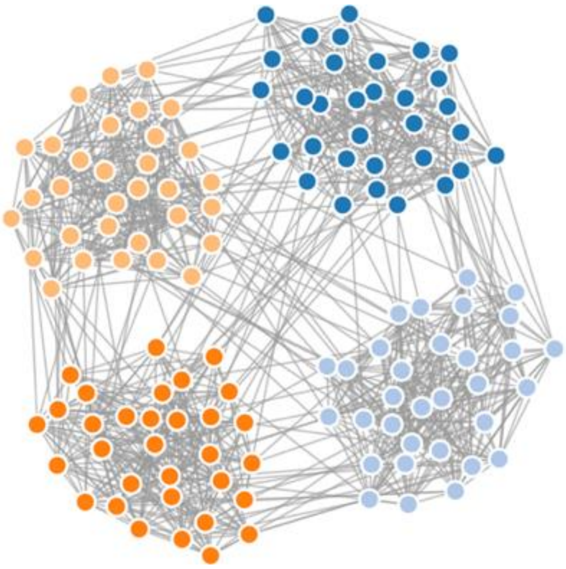


Figure 16 GN synthetic network. $N = 128$, $\mu = 0.1$ and $k = 16$. Represented using D3.js and SVG.

The algorithm hereby implemented not only allows the mixing parameter to vary, but also the average degree (Algorithm 4). This led to an increased number of benchmark tests using these networks.

Algorithm 4 GN Benchmark

```
Require:
$mix$: mixing parameter.
$deg$: degree of each node.
Ensure:
$E$: resulting set of edges.
$V$: resulting set of nodes.
$A$: adjacency matrix.
$T$: resulting set of temporary selectable nodes.
// Generating nodes of GN network.
$k = 0$; // Iteration number.
$V = \{\}$;
repeat
  If $k < 32$ then
    $V = V \cup \{id: k, group: 1\}$
  else if $k < 64$
    $V = V \cup \{id: k, group: 2\}$
  else if $k < 96$
    $V = V \cup \{id: k, group: 3\}$
  else
    $V = V \cup \{id: k, group: 4\}$
  end if
  $k = k + 1$
until $k < 128$
```

```
// Generating internal edges to each group of GN network.
Randomize the order of the set of vertices $V$ maintaining them ordered by blocks of ground-truth communities.
$i = 0$;
$A^{initial} = [0]_{128\times128}$
$E^{initial} = \{\}$;
repeat
   $T = \{\}$;
   $j = 0$;
   repeat
      If $V_j.group === V_i.group$ then
         $T = T \cup \{T_j\}$
      end if
      $j = j + 1$
   until $j < 128$
   $j = 0$;
   loop:
   repeat
      $j = round(rand * (T.length - 1))$ // Generates random position in $T$ array.
      repeat
         $T = T \backslash \{T_j\}$
         If $T === \{\}$ then
            break loop;
         end if
         $j = round(rand * (T.length - 1))$
      until $A^{initial}_{(V_i.id)(T_j.id)} === 1$ or $degree(T_j) \geq deg \times (1 - mix)$
      $E^{initial} = E^{initial} \cup \{source{:}\, V_i.id, target{:}\, T_j.id, value{:}\, 1\}$
      $A_{(V_i.id)(T_j.id)} = 1$; $A_{(T_j.id)(V_i.id)} = 1$
   until $degree(V_i) < deg \times (1 - mix)$
   If $(i + 1)\%32 === 0$ then
      $aux = 0$; // Auxiliary variable.
      $k = 0$;
      repeat
         If $V_i.group === V_k.group$ then
            $aux = aux + (deg \times (1 - mix) - degree(V_k))$;
         end if
         $k = k + 1$
      until $k < 128$
      If $aux > 0$ then
         $i = i - 32$;
      else
         $E^{final} = E^{final} \cup E^{initial}$; $A^{final}_{(V_i.id)(T_j.id)} = A^{final}_{(V_i.id)(T_j.id)} + A^{initial}_{(V_i.id)(T_j.id)}$
      end if
      $E^{initial} = \{\}$;
      $A^{initial}_{(V_i.id)(T_j.id)} = [0]_{128\times128}$;
   end if
   $i = i + 1$;
until $index < 128$
// Generating external edges to each group of GN network.
Randomize the order of the set of vertices $V$.
```

```
$i = 0;$
$A^{initial2} = [0]_{128\times128}$
$E^{initial2} = \{\};$
repeat
    $T = \{\};$
    $j = 0;$
    repeat
        If $V_j.group$ ! == $V_i.group$ then
            $T = T \cup \{T_j\}$
        end if
        $j = j + 1$
    until $j < 128$
    $j = 0;$
    loop:
    repeat
        $j = round(rand * (T.length - 1))$ // Generates random position in $T$ array.
        repeat
            $T = T \backslash \{T_j\}$
            If $T$ === {} then
                $i = -1;$
                $A^{initial2} = [0]_{128\times128}$
                $E^{initial2} = \{\};$
                break loop;
            end if
            $j = round(rand * (T.length - 1))$
        until $A^{initial2}_{(V_i.id)(T_j.id)}$ === 1 or $degree(T_j) \geq deg \times mix$
        $E^{initial2} = E^{initial2} \cup \{source: V_i.id, target: T_j.id, value: 1\}$
        $A_{(V_i.id)(T_j.id)} = 1;\ A_{(T_j.id)(V_i.id)} = 1$
    until $degree(V_i) < deg \times mix$
    $A^{final}_{(V_i.id)(T_j.id)} = A^{final}_{(V_i.id)(T_j.id)} + A^{initial2}_{(V_i.id)(T_j.id)}$
    $E^{final} = E^{final} \cup E^{initial2}$
    $i = i + 1;$
until $index < 128$
```

3.2.2 Lancichinetti-Fortunato-Radicchi Network

LFR benchmark networks [63, 23] try to approximate the real ones. This means that not only they consider a power-law distribution of community sizes with coefficient ζ (94), but also a power-law distribution of node degrees with coefficient γ (95):

$$p_{N_c} \sim {N_c}^{-\zeta} \tag{94}$$

$$p_k \sim k^{-\gamma} \tag{95}$$

This technique starts by generating N isolated nodes. Then, each one is assigned to a different community such that following the correct size distribution of the network. Based on the nodes' degree distribution, a degree k is attributed to each node. Next, each one will be assigned with $(1-\mu)k_i$ internal edges and the remaining of the connections will be linked to nodes from outside the respective cluster. Finally, all links (based on nodes' internal and external degrees) are determined randomly (Figure 17).

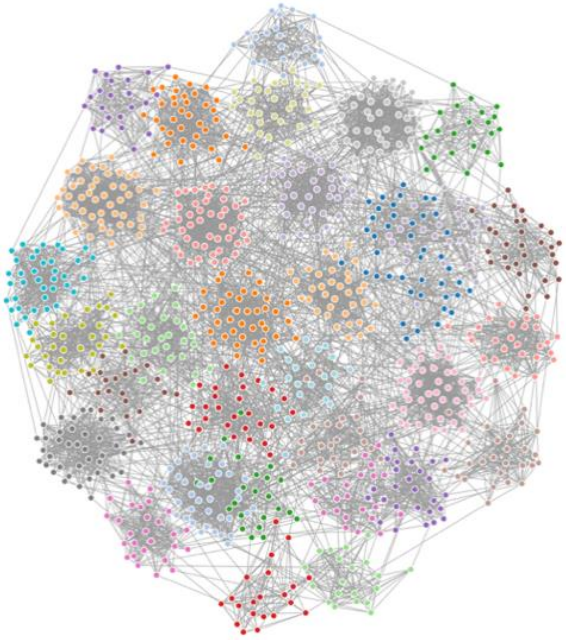


Figure 17 LFR synthetic network. $N = 1000$, $\mu = 0.1$, $k_{avg} = 15$, $k_{max} = 50$, $c_{min} = 20$ and $c_{max} = 50$. Represented using D3.js and SVG.

3.2.3 Normalized Mutual Information

NMI was the measure chosen to evaluate network partitioning performed by each community finding algorithm. It was considered due to its comprehensive meaning, the fact of allowing the comparison of two partitions even when a different number of clusters was identified (detailed below) and because it is already widely used to compare community detection methods [64].

NMI is a variant of a common measure in information theory called mutual information. Mutual information accounts to the "amount of information" one can extract from a distribution regarding a second one. In the case of discrete distributions, mutual information of 2 jointly random variable X and Y is calculated as a double sum:

$$I(X;Y) = \sum_{y\in Y}\sum_{x\in X} p_{(X,Y)}(x,y)\log\left(\frac{p_{(X,Y)}(x,y)}{p_X(x)p_Y(y)}\right) \tag{96}$$

Upon observation of (96), if X and Y are independent random variables, then:

$$\frac{p_{(X,Y)}(x,y)}{p_X(x)p_Y(y)} = 1 \Longrightarrow \log\left(\frac{p_{(X,Y)}(x,y)}{p_X(x)p_Y(y)}\right) = 0 \Longrightarrow I = 0 \tag{97}$$

A set of properties of mutual information result from definition (96).

Nonnegativity

Using Jensen's inequality one can show [65]:

$$I(X,Y) \geq 0 \tag{98}$$

Symmetry

By definition, $p_{(X,Y)}(x,y)$ is symmetrical. This implies:

$$I(X,Y) = I(Y,X) \tag{99}$$

Conditional and Joint Entropy Dependence

$$\begin{aligned} I(X;Y) &= \sum_{x\in X, y\in Y} p_{(X,Y)}(x,y) \log \frac{p_{(X,Y)}(x,y)}{p_X(x)p_Y(y)} \\ &= \sum_{x\in X, y\in Y} p_{(X,Y)}(x,y) \log \frac{p_{(X,Y)}(x,y)}{p_X(x)} - \sum_{x\in X, y\in Y} p_{(X,Y)}(x,y) \log p_Y(y) \\ &= \sum_{x\in X, y\in Y} p_X(x,y) p_{Y|X=x}(y) \log p_{Y|X=x}(y) - \sum_{x\in X, y\in Y} p_{(X,Y)}(x,y) \log p_Y(y) \\ &= \sum_{x\in X} p_X(x,y) \left(\sum_{y\in Y} p_{Y|X=x}(y) \log p_{Y|X=x}(y) \right) \\ &\quad - \sum_{y\in Y} \left(\sum_{x\in X} p_{(X,Y)}(x,y) \right) \log p_Y(y) \\ &= -\sum_{x\in X} p_X(x) H(Y|X=x) - \sum_{y\in Y} p_Y(y) \log p_Y(y) = -H(Y|X) + H(Y) \\ &= H(Y) - H(Y|X) \end{aligned} \tag{100}$$

See Figure 18.

Clustering quality of the community finding algorithms was tested using a normalized measure of mutual information – NMI [35]. An algorithm was developed, in JavaScript, in the thesis and tested in several networks. It receives an input of 2 arrays, with the same length, and returns the corresponding normalized mutual information.

$$NMI(Y,C) = \frac{2 \times I(Y;C)}{H(Y) + H(C)} \tag{101}$$

NMI not only depends on the mutual information I, but also on the entropy of the labeled $\mathrm{H(Y)}$ and clustered set $\mathrm{H(C)}$.

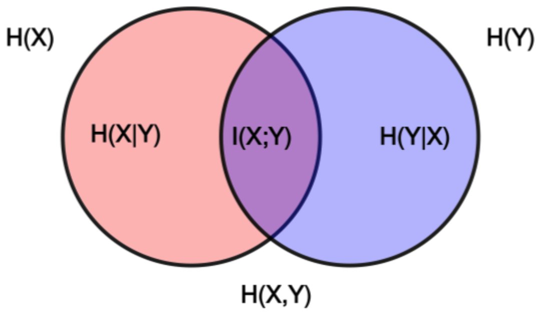


Figure 18 Venn diagram depicting the relation between different measures of entropy and mutual information [66].

3.3 Real Test Data

The higher the number of tests performed using the implemented algorithms, the higher the certainty they are working properly for the broadest number of networks. This way, disconnected networks (sampled from a real

Amazon network), small sized ones (Zachary's Karate Club) and a biological/phylogenetic network (*S. aureus*) were additionally considered to validate algorithm's correctness. Nevertheless, no benchmark tests were performed using those.

In the next subsections, Amazon, Zachary's Karate Club and *S. aureus* networks are described in terms of their topology (number of nodes and links), meaning of the pattern of connections and how they were obtained.

3.3.1 Amazon Network

This network was obtained after crawling Amazon website [60]. Each node represents a given product available in the store and a connected pair means they are frequently bought together. The original network contains 334 863 nodes and 925 872 links. Only the first 5 000 links along the respective nodes were used in the tests (Figure 19).

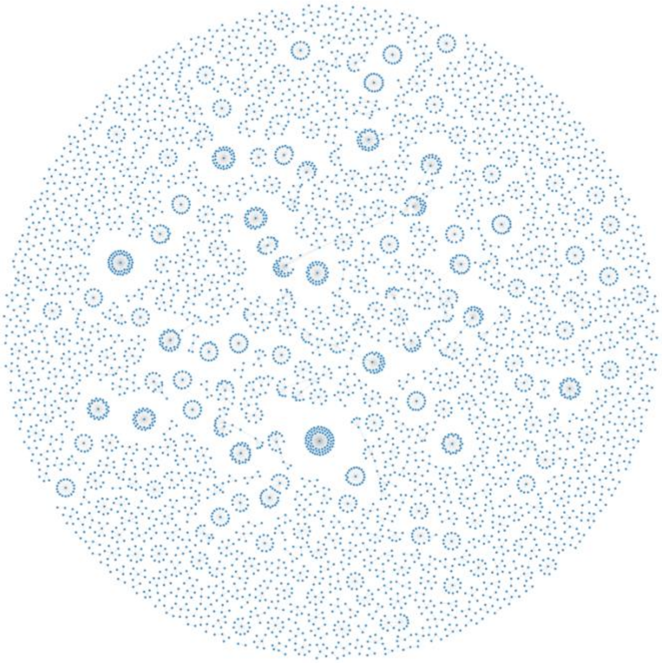


Figure 19 Amazon product co-purchasing network sampled from a 5 000 links graph. Represented using D3.js and SVG.

3.3.2 Zachary's Karate Club Network

One of the most popular networks in network science is the Zachary's Karate Club [61]. It became popular after have being used, in 2002, in a paper of Michelle Girvan and Mark Newman [23]. This is a social network representing the 77 relations between 34 individuals from a university karate club.

Wayne W. Zachary studied this network for 3 years, from 1970 to 1972. During their study, a conflict arose and the club split in 2. Half the members formed a new a club with the instructor and the other half found a new instructor or gave up karate.

Each node is a member. Node 1 is the instructor and 34 is the president (Figure 20). Dark and light blue nodes represent the split inside the club that was created due to a conflict between them. Each link connects 2 members when they meet regularly outside the club, before splitting.

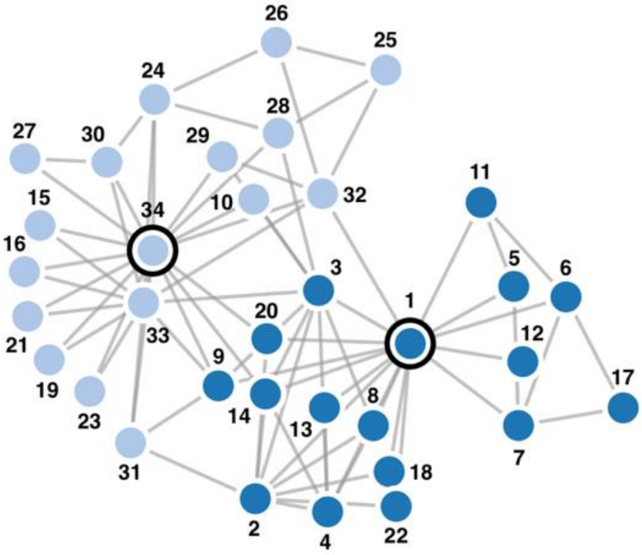
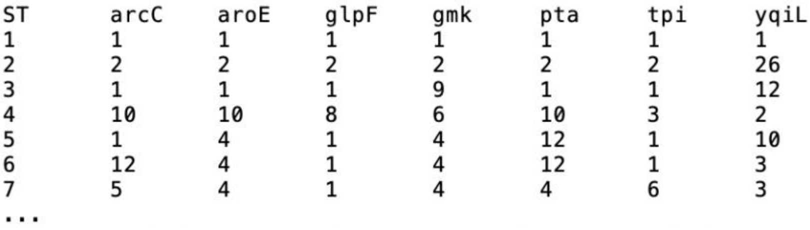


Figure 20 Zachary's karate club network. Represented using D3.js and SVG.

3.3.3 *Staphylococcus aureus*

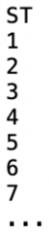
To test the application of the community finding algorithms in a real biological network, an MLST database of allelic profiles for *S. aureus* was used (Figure 21) [62]. The MLST profiles contain information about 7 *loci* (*arcC*, *aroE*, *glpF*, *gmk*, *pta*, *tpi* and *yqiL*) and, at the time of download, it had 5199 MLST profiles (ST). A graph was created by linking each ST to all of its Single Locus Variants (SLVs) and the largest component was used for the benchmark of the community finding algorithms. The choice of this test network was based on the fact it can emulate similar patterns to the ones that are created using cgMLST or wgMLST profiles, which have many more loci than MLST.



```
ST      arcC    aroE    glpF    gmk     pta     tpi     yqiL
1       1       1       1       1       1       1       1
2       2       2       2       2       2       2       26
3       1       1       1       9       1       1       12
4       10      10      8       6       10      3       2
5       1       4       1       4       12      1       10
6       12      4       1       4       12      1       3
7       5       4       1       4       4       6       3
...
```

Figure 21 *S. aureus* MLST profile.

1. PHYLOViZ 2 was firstly installed. A MLST profile of *S. aureus*, along with the corresponding metadata file (Figure 22), was then included in the platform.



```
ST
1
2
3
4
5
6
7
...
```

Figure 22 Isolate data metafile.

A minimum spanning tree (Figure 23) was generated using *goeBURST* and following all tiebreak rules. PHYLOViZ 2 prints a set of clonal complexes (CCs) which were defined based on the similarity among STs;

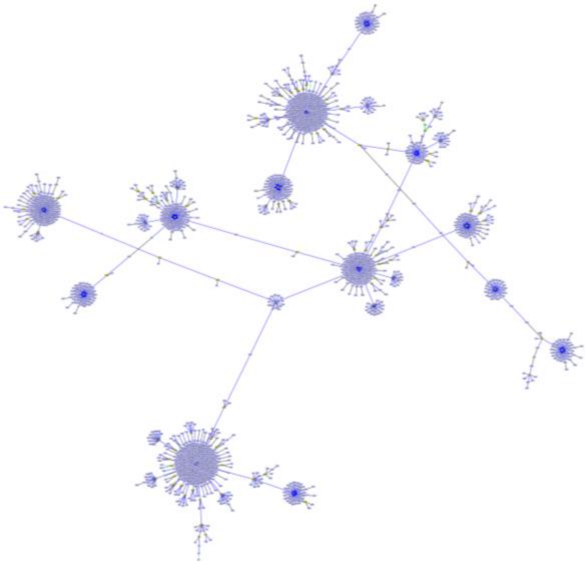
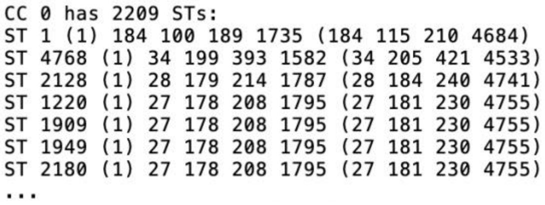


Figure 23 Minimum spanning tree of *S. aureus* generated using PHYLOViZ 2.

2. Copy-paste operation was manually performed to send clonal complex data from PHYLOViZ 2 to a text file. Data format (Figure 24) is the same as provided by the Java tool;



```
CC 0 has 2209 STs:
ST 1 (1) 184 100 189 1735 (184 115 210 4684)
ST 4768 (1) 34 199 393 1582 (34 205 421 4533)
ST 2128 (1) 28 179 214 1787 (28 184 240 4741)
ST 1220 (1) 27 178 208 1795 (27 181 230 4755)
ST 1909 (1) 27 178 208 1795 (27 181 230 4755)
ST 1949 (1) 27 178 208 1795 (27 181 230 4755)
ST 2180 (1) 27 178 208 1795 (27 181 230 4755)
...
```

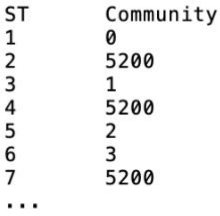
Figure 24 Data from the CC 0 of the *S. aureus* network obtained using PHYLOViZ 2.

3. This file was included in Phyl. Original MLST data profile was also provided;

4. The web application built an SLV network with all the strains present in the CC under study and considering *loci* information in the profile file;

5. Community finding algorithms were executed in the previous network;

6. The input metadata file is now returned with an extra column where every determined community is represented (Figure 25). For all nodes that were not part of the CC being studied, the same random number was attributed (accounting that it should be different than any other already used for strains in the CC).



```
ST      Community
1       0
2       5200
3       1
4       5200
5       2
6       3
7       5200
...
```

Figure 25 Final text file generated upon execution of Phyl. It contains an additional Community column, not present in the initial metadata file.

3.4 Visualization Frameworks

The most notable examples of older browser visualization frameworks are Prefuse, Flare and Protovis toolkits. The first was created in 2005 and required the use of Java, as well as a Java plug-in to allow rendering in the browser. Later, in 2007, Flare start becoming the web standard. It used ActionScript and required a Flash plugin for rendering. In 2009, a group from Stanford University created Protovis. This JavaScript library was able to generate SVGs from data sets. The same team, in 2011, interrupted Protovis development and focused in a new project called D3.js [67]. Their project, a few years later, become the web standard to dynamically and efficiently represent data in every modern browser. D3.js not only avoids a monolithic implementation, turning data representation more flexible, but presents enhanced performances by making use of the most modern browser technologies: Hypertext Markup Language (HTML), Cascading Style Sheets (CSS) and JavaScript. Comparing with DOM standards, it uses a less verbose language and still allows the user to seamlessly interact with every element present in the webpage.

In 2015, a JavaScript library called Cytoscape.js was launched [68]. It was the predecessor of the Java based Cytoscape, released in 2002. Cytoscape.js focuses not only in representing graphs but also in providing the user the tools to do analyze them.

Along with D3.js, Cytoscape.js performs seamlessly in the most modern browsers and in all devices. It does support a wide variety of graph properties: directed, undirected and loops. Although, it is not possible to use it for other applications apart from analyzing or representing networks.

In the visualization application, the user can opt by visualizing every network using D3 or Cytoscape JavaScript libraries. In the case of the first framework, there is the possibility of visualizing data through a Canvas or an SVG element. All networks were plotted using a force directed implementation. This mode requires the developer to set features like the gravity, repulsion or spring constant which will affect how the network is displayed. Depending on those parameters, node and link overlapping should be reduced in order to get an enhanced view of the graph (Figure 26).

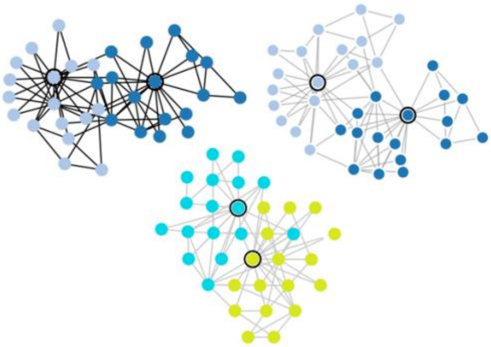


Figure 26 Zachary's Karate Club network represented using D3.js Canvas (top-left), D3.js SVG (top-right) and Cytoscape.js (bottom).

Regarding benchmark plotting, D3.js was used. Its flexibility in terms of visual representation, performance and the number of practical examples available on the web, determined this choice.

4. Results

In this section, the web application available online is first described. Then, the results from benchmarking the community finding algorithms against GN and LFR synthetic networks are presented. Next, benchmark network generator algorithms were analyzed based on the size, average node degree and mixing of the network. After, two JavaScript visualization frameworks are discussed in terms of their pros and cons. Finally, all community finding algorithms (with varying input) are run against a real network – *S. aureus* MLST SLV network, to assess which one performs the best in detecting the direct descendants of a given ancestor.

4.1 Web Application

In order to facilitate the analysis and visualization of the results after the execution of the previous algorithms, a web application was implemented (mscthesis.herokuapp.com) (Figure 27).

Its backend was conceived using Node.js and it runs in a Heroku server which is linked to a GitHub repository (github.com/warcraft12321/Thesis) containing all the implementations and documents related to the thesis. A Digital Object Identifier (DOI) was attributed to this repository using Zenodo (zenodo.org/badge/latestdoi/162063699). An image of the app is available at Docker Hub (cloud.docker.com/u/warcraft12321/repository/docker/warcraft12321/thesis). The following actions are strictly executed in the cloud:

- Generation of GN and LFR benchmark networks. The latter are generated using a binary from a C++ implementation [59]. This was possible after importing a package from NPM which simulates the command line in the server;

- Amazon [60], Zachary's Karate Club [61] and *S. aureus* testing networks had their data properly processed in the cloud, so that it could be sent and displayed in the frontend;

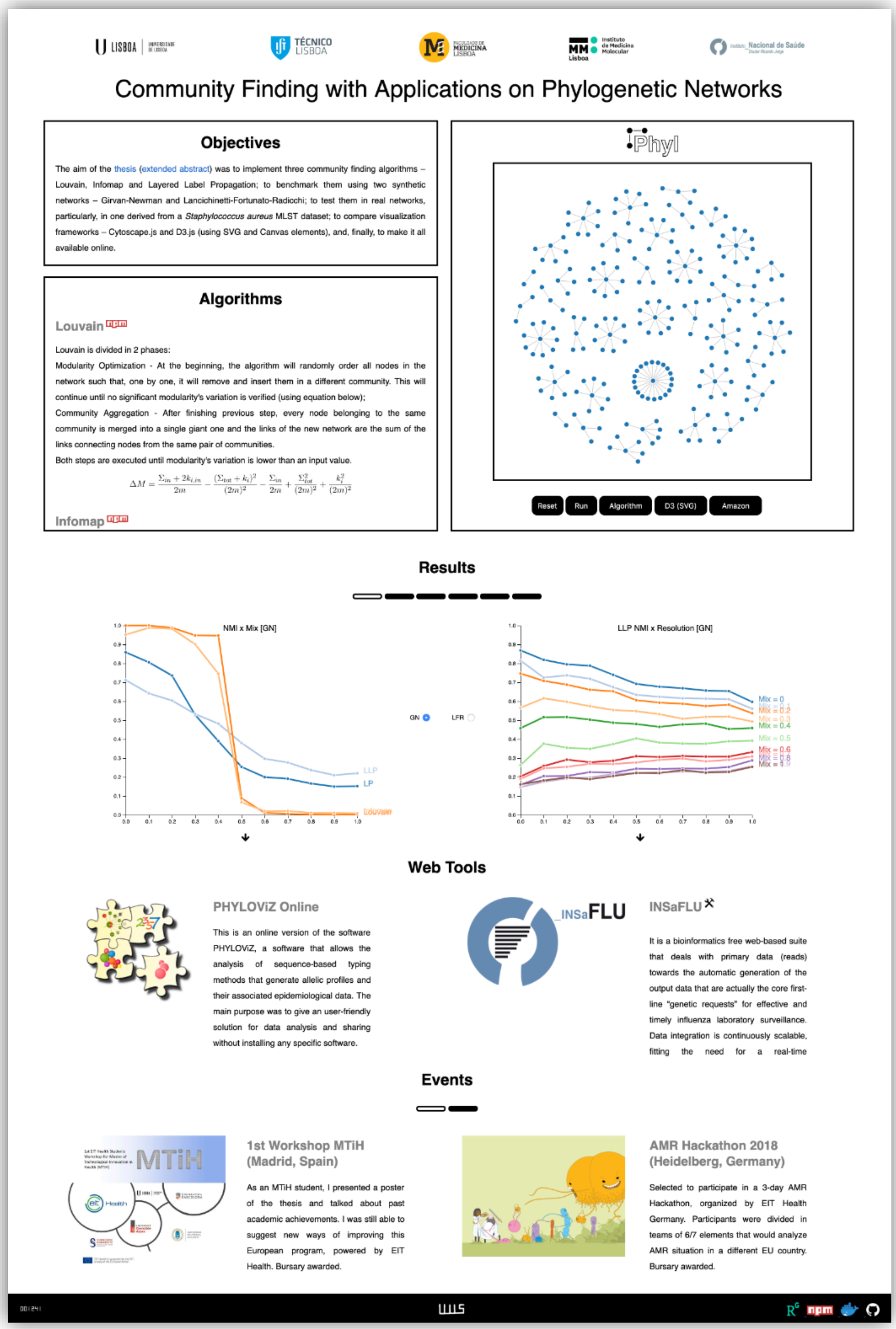
- Once the execution of Louvain, Infomap and LLP algorithms is not performed in the user's browser, this may raise some privacy issues. Although, it was not feasible to run Infomap in the user side and still keep the web application functional in all devices;

- The communication between the app and NPM was established in the server, so that it is possible to import and plot the statistics from each package uploaded to NPM in the application;

- Finally, all benchmarking data was obtained after running 10 times, for each configuration, the algorithms in the cloud.

The frontend is both static and dynamic. The static counterpart uses HTML, CSS and JavaScript. The dynamic execution is managed by Node.js, which runs in the server-side. Several sections were individualized in the interface:

- *Objectives* contains a concise description of the goals of the thesis;
- *Algorithms* shortly explains the core principles of Louvain, Infomap and LLP;
- *Phyl* allows the user to actively test all the community finding algorithms against 5 different networks – GN, LFR, Amazon, Zachary's Karate Club or *S. aureus*. Using any of 3 available visualization options – D3.js (Canvas), D3.js (SVG) and Cytoscape.js;
- *Results* joins in a single section all data obtained from a thorough analysis of every implemented algorithm. It dynamically represents data so that one can analyze it individually and in a clearer way: considering one line at a time with CI 95% error bars. The same plot can be visualized considering GN or LFR benchmark networks by selecting the respective radio button. On the top of each plot, both axles are identified following the rule: (y-axis) x (x-axis). At the end of each line, additional information is provided in order to identify each curve. Under each x-axis, the user has the possibility to download the Comma-Separated Values (CSV) file with the data used in each plot;
- *Web tools* – Tree representations of the data returned by the community finding algorithms, implemented in the thesis, can be visualized in PHYLOViZ Online. New visualization tools are being developed in the influenza-oriented web application – INSaFLU;
- *Events* describes 4 contests/workshops which were crucial for the development of the thesis.



Community Finding with Applications on Phylogenetic Networks

Objectives

The aim of the thesis (extended abstract) was to implement three community finding algorithms – Louvain, Infomap and Layered Label Propagation; to benchmark them using two synthetic networks – Girvan-Newman and Lancichinetti-Fortunato-Radicchi; to test them in real networks, particularly, in one derived from a *Staphylococcus aureus* MLST dataset; to compare visualization frameworks – Cytoscape.js and D3.js (using SVG and Canvas elements), and, finally, to make it all available online.

Algorithms

Louvain npm

Louvain is divided in 2 phases:

Modularity Optimization - At the beginning, the algorithm will randomly order all nodes in the network such that, one by one, it will remove and insert them in a different community. This will continue until no significant modularity's variation is verified (using equation below);

Community Aggregation - After finishing previous step, every node belonging to the same community is merged into a single giant one and the links of the new network are the sum of the links connecting nodes from the same pair of communities.

Both steps are executed until modularity's variation is lower than an input value.

$$\Delta M = \frac{\Sigma_{in} + 2k_{i,in}}{2m} - \frac{(\Sigma_{tot} + k_i)^2}{(2m)^2} - \frac{\Sigma_{in}}{2m} + \frac{\Sigma_{tot}^2}{(2m)^2} + \frac{k_i^2}{(2m)^2}$$

Infomap npm

Results

Web Tools

PHYLOViZ Online

This is an online version of the software PHYLOViZ, a software that allows the analysis of sequence-based typing methods that generate allelic profiles and their associated epidemiological data. The main purpose was to give an user-friendly solution for data analysis and sharing without installing any specific software.

INSaFLU

It is a bioinformatics free web-based suite that deals with primary data (reads) towards the automatic generation of the output data that are actually the core first-line "genetic requests" for effective and timely influenza laboratory surveillance. Data integration is continuously scalable, fitting the need for a real-time

Events

1st EIT Health Students Workshop for Master of Technological Innovation in Health (MTiH)
MTiH
Health

1st Workshop MTiH (Madrid, Spain)

As an MTiH student, I presented a poster of the thesis and talked about past academic achievements. I was still able to suggest new ways of improving this European program, powered by EIT Health. Bursary awarded.

AMR Hackathon 2018 (Heidelberg, Germany)

Selected to participate in a 3-day AMR Hackathon, organized by EIT Health Germany. Participants were divided in teams of 6/7 elements that would analyze AMR situation in a different EU country. Bursary awarded.

Figure 27 Web application: Objectives, Algorithms, Phyl, Results, Web Tools and Events.

4.2 Community Finding Algorithms

In terms of accuracy, Louvain outperformed all others for weakly mixed networks ($\mu < 0.4$). It is followed by Infomap, LP and LLP (γ $= 0.5$) for $\mu < 0.3$ or Infomap, LLP and LP (γ $= 0.5$) for $0.3 \leq \mu \leq 0.4$. By increasing μ, LLP, which is able to detect communities based on the panorama of the whole network, consistently becomes the most effective for $\mu \geq 0.5$. Comparing Louvain and Infomap when $\mu \geq 0.5$, their performance is similar. Based on the CI 95%, it is not possible to state one performs better than the other. Another point to highlight is that Louvain and Infomap are very sensitive, consequently, unreliable detecting communities in networks with $\mu \approx 0.5$. LLP and LP do not present any specific value in which community detection is steeply affected. These conclusions are all valid to GN and LFR networks (Figure 28).

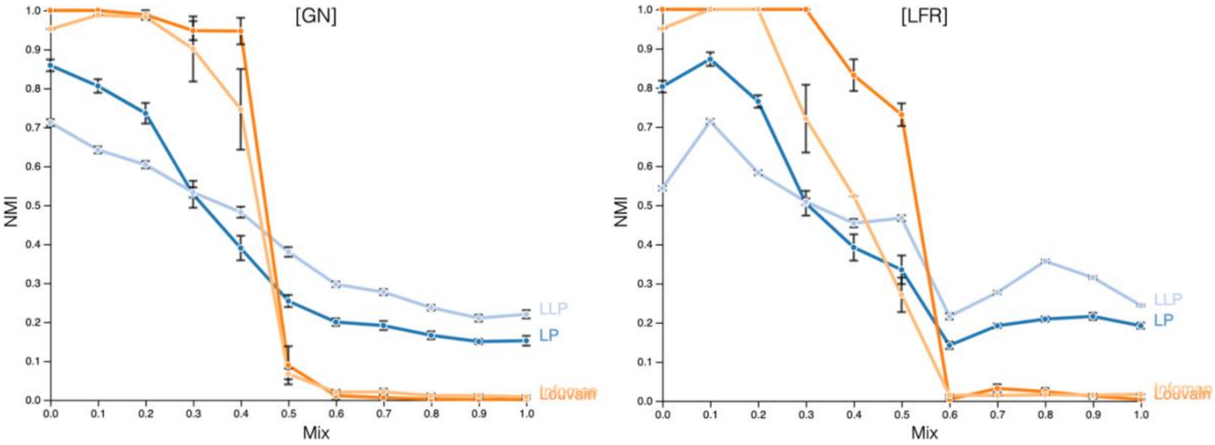


Figure 28 Clustering quality of each algorithm: Louvain (orange), Infomap (light-orange), LP (blue) and LLP (light-blue). Analyzed GN and LFR networks have the same parameters as in Figure 16 and Figure 17, respectively (excluding mixing). Some error bars may not be totally visible due to their small values.

Having considered only two gamma values in LLP, a more precise analysis was needed to check whether there could be others achieving better results.

In the case of GN network, it is possible to detect communities with higher accuracy in networks less mixed. The performance decreases whenever we increase γ for $\mu < 0.5$. When $\mu \geq 0.5$, the NMI between the detected partition and the one in the original network is higher as long as we keep increasing γ. Another point to highlight is the possibility of detecting communities with higher precision in networks with higher μ using appropriate γ than in others with lower μ but with a non-optimal γ. This is valid for networks with $\mu > 0.5$ (Figure 30).

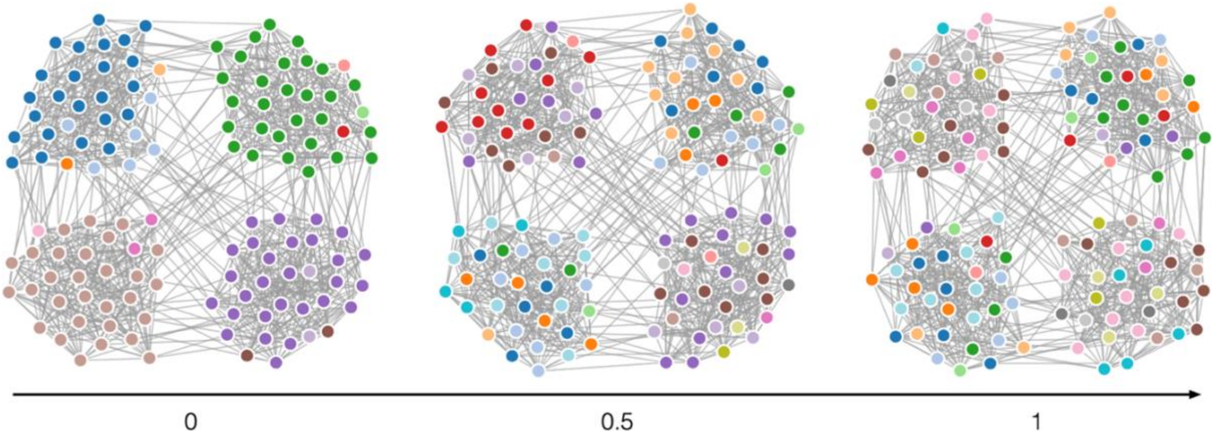


Figure 29 GN network (Figure 16) after running LLP with resolutions: 0 (left), 0.5 (center) and 1 (right). When γ is closer to 0, it highlights coarse structure of the network with few, big and sparse communities. As γ grows, fine-grained structure is unveiled. Communities become smaller and denser. Represented using D3.js and SVG.

In LFR networks, increasing gamma always leads to a decrease in the NMI (Figure 30).

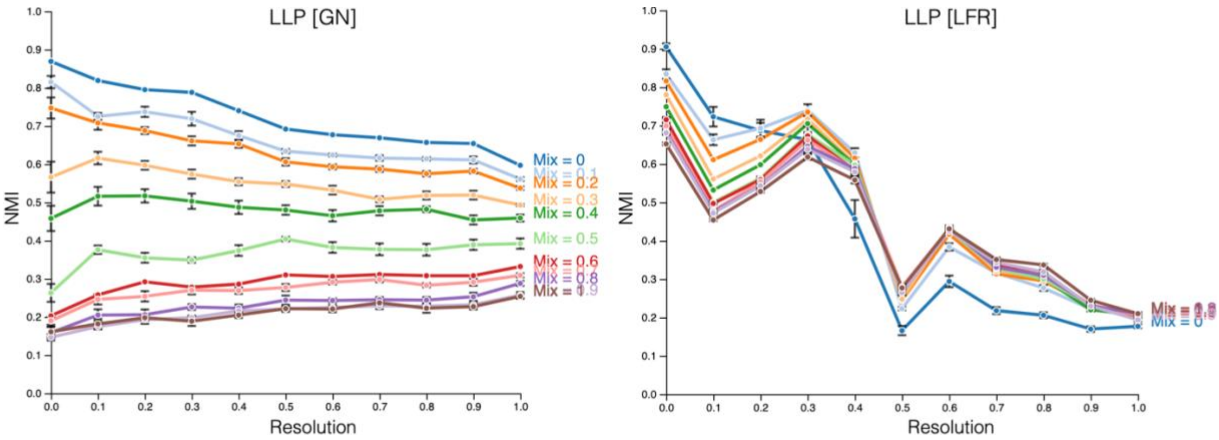


Figure 30 LLP algorithm accuracy in terms of the resolution parameter. Analysis performed for mixing parameters 0 – 1 (blue-brown) in GN and LFR networks, with the remaining properties as in Figure 16 and Figure 17, respectively. Some error bars may not be totally visible due to their small values.

The influence of the average node degree was tested to check whether it affects the capacity of each algorithm to identify communities.

In the case of GN networks, for $\mu \leq 0.5$, an increased average degree enhances the capacity of Louvain, Infomap, LP and LLP to differentiate communities. For $\mu > 0.5$, NMI clustering quality is lower as higher is the average node degree. Assuming it is not expected the algorithms to identify the original communities for $\mu \approx 1$, this suggests that higher average degrees tends to decrease detection by chance. This observation can be applied to GN and LFR networks (Figure 31, Figure 32, Figure 33 and Figure 34).

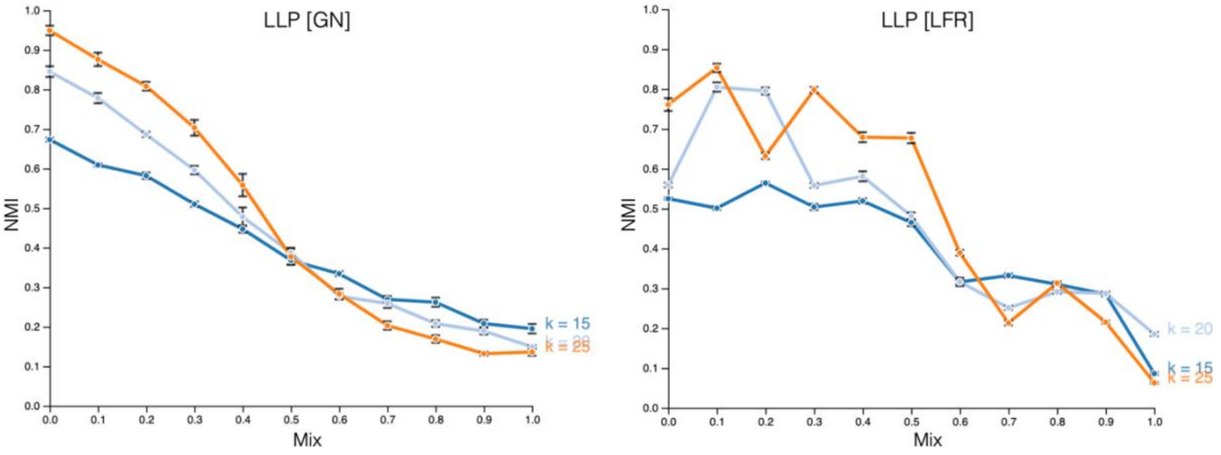


Figure 31 LLP algorithm accuracy in terms of the mixing parameter. Analysis performed in GN and LFR networks with average node degrees: 15 (blue), 20 (light-blue) and 25 (orange). The remaining properties are similar to the ones in Figure 16 (GN network) and Figure 17 (LFR network). Some error bars may not be totally visible due to their small values.

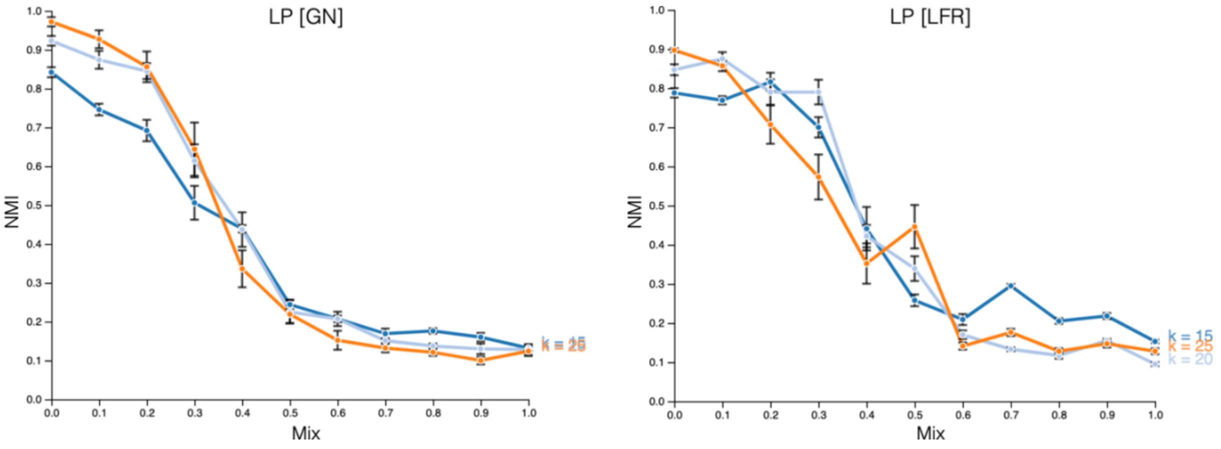


Figure 32 Label Propagation algorithm accuracy in terms of the mixing parameter. Analysis performed in GN and LFR networks with average node degrees: 15 (blue), 20 (light-blue) and 25 (orange). The remaining properties are similar to the ones in Figure 16 (GN network) and Figure 17 (LFR network). Some error bars may not be totally visible due to their small values.

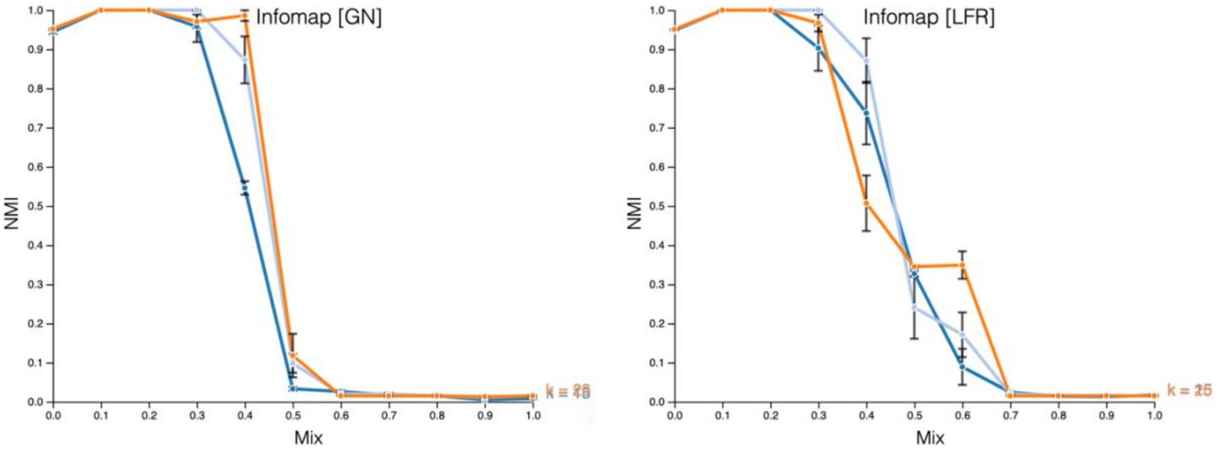


Figure 33 Infomap algorithm accuracy in terms of the mixing parameter. Analysis performed in GN and LFR networks with nodes with average node degrees: 15 (blue), 20 (light-blue) and 25 (orange). The remaining properties are similar to the ones in Figure 16 (GN network) and Figure 17 (LFR network). Some error bars may not be totally visible due to their small values.

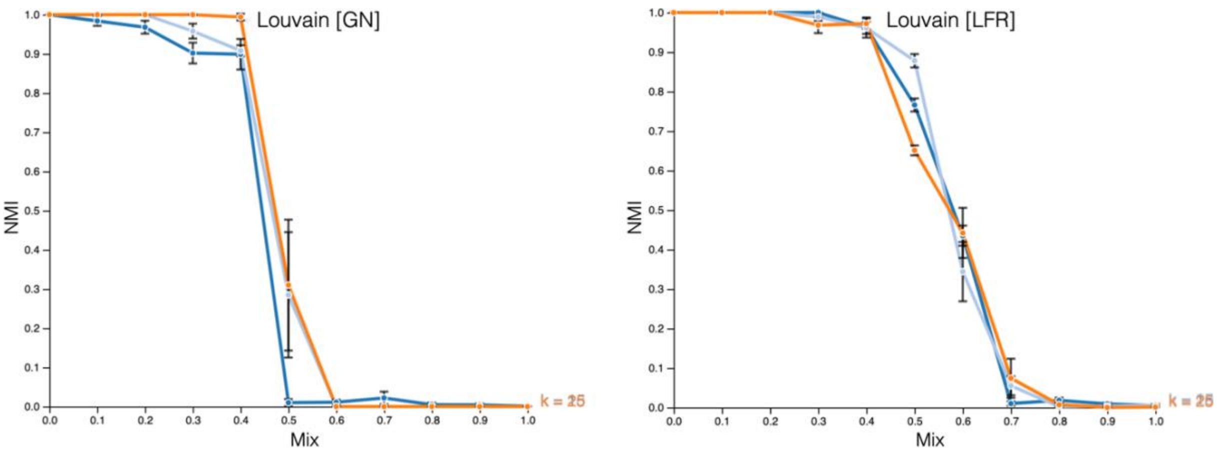


Figure 34 Louvain algorithm accuracy in terms of the mixing parameter. Analysis performed in GN and LFR networks with nodes with average node degrees: 15 (blue), 20 (light-blue) and 25 (orange). The remaining properties are similar to the ones in Figure 16 (GN network) and Figure 17 (LFR network). Some error bars may not be totally visible due to their small values.

In terms of <u>speed</u>, Louvain algorithm performed significantly better than any other. One important property that contributes to its efficiency is that it only calculates modularity variation at each iteration of *Modularity Optimization* phase (85). In Infomap, minimum description length is recalculated at the end of each iteration of the optimization step (91). Due to this computationally intensive step, Infomap presented the poorest performance in the benchmark tests. Based on Table 2, it was expected the difference of execution time between Infomap and all others to be logarithmic. Due to inefficient steps in the implementation, this was not achieved. Comparing LLP and LP, the fastest was the one that needed to optimize the simplest equation – LP (Figure 35). Although, time complexity appears to be the same, as predicted in Table 2. Again, using the same table, It was expected LLP and LP to execute about as fast as Louvain. The small difference observed in Figure 35 might be explained by some non-optimized step in the implementation of LLP/LP.

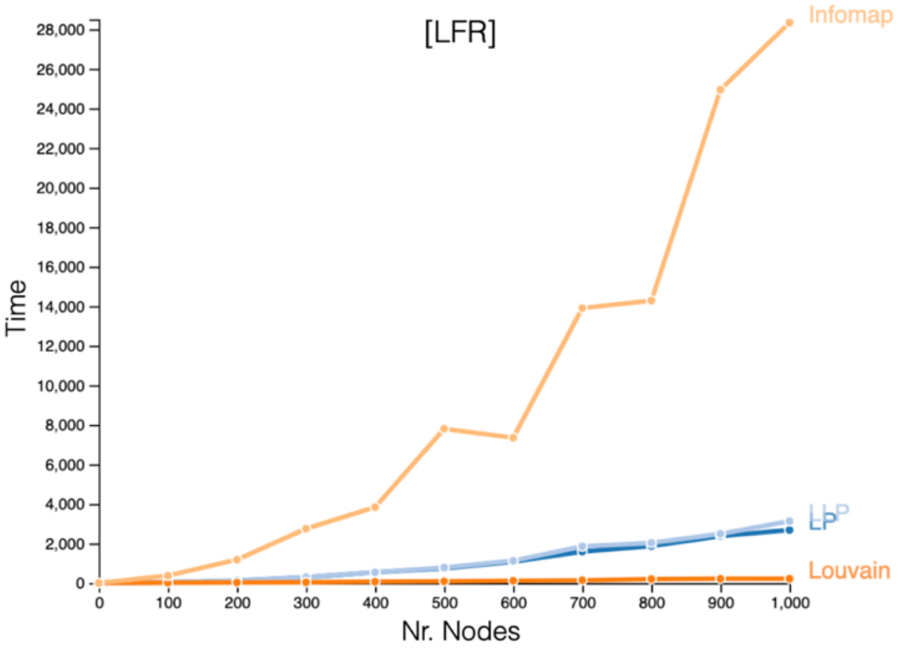


Figure 35 Time that Louvain (orange), Infomap (light-orange), LLP (light-blue) and LP (blue) took to finalize the analysis in terms of the size of the LFR network. The remaining properties are similar to the ones in Figure 17. Some error bars may not be totally visible due to their small values.

4.3 Benchmark Networks Algorithms

Previous GN and LFR networks were generated so the previous tests could be performed. The algorithm to create the former was implemented in the thesis. The time it takes to generate such networks increases exponentially with the average node degree and the mixing parameter. Becoming unfeasible to execute it for average degrees higher than 20, in an ordinary computer. Its performance was compared to the Fortunato's implementation which is highly scalable and allows the generation of networks with much higher number of links in shorter times. It was verified an approximately linear increase in the time of generation of networks with progressively higher mixing parameters, contrarily to the one implemented in the thesis. Just like before, generating networks with more edges, requires additional computational power (Figure 36).

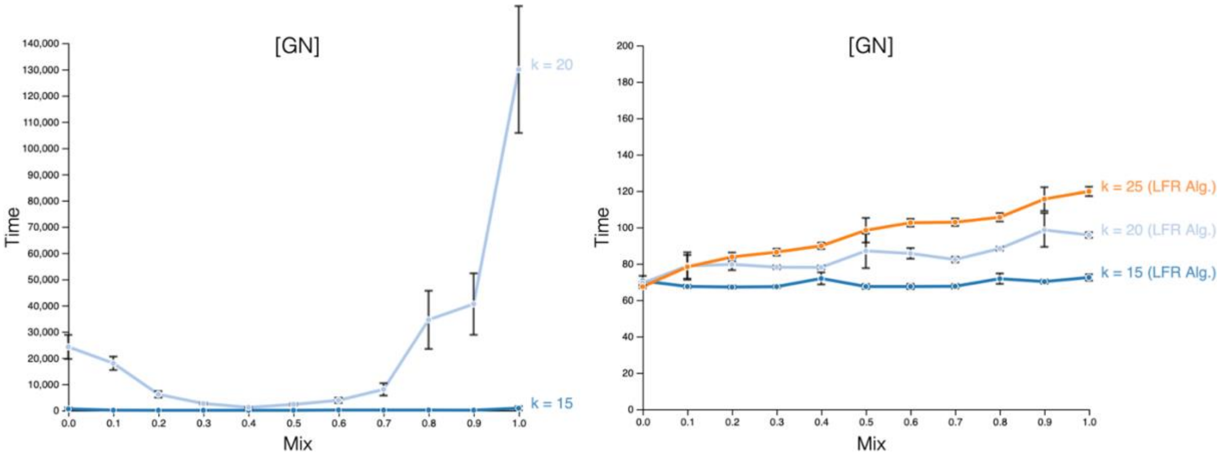


Figure 36 Time needed to generate GN networks (using thesis and LFR implementation) in terms of the mixing parameter. Analysis performed in networks average node degrees: 15 (blue), 20 (light-blue) and 25 (orange). The remaining properties are similar to the ones in Figure 16. Some error bars may not be totally visible due to their small values.

Additional tests were performed in LFR networks. The execution time was tested against the number of nodes and the mixing parameter of the network. In both cases, it was verified an approximately linear relation (Figure 37). The number of nodes was not considered in GN networks, once they have a fixed number of elements per community and of communities. In spite only the mixing parameter and the number of nodes were analyzed, the user can still choose to adjust nodes maximum degree, the exponents of the nodes degree distribution and communities size distribution, the maximum/minimum number of nodes per community, the number of overlapping nodes per community (covers were not considered in the thesis) and the number of memberships for the overlapping nodes [59].

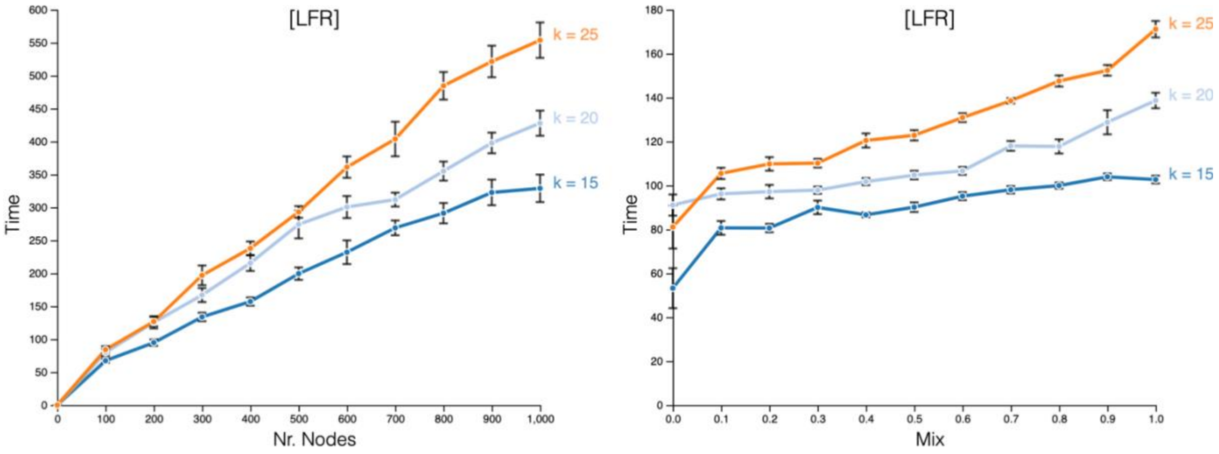


Figure 37 Time needed to generate LFR benchmark networks in terms of the number of nodes (left) and mixing parameter (right). Analysis performed in networks with average node degrees: 15 (blue), 20 (light-blue) and 25 (orange). The remaining properties are similar to the ones in Figure 17. Some error bars may not be totally visible due to their small values.

Every community finding, benchmark network generator, accuracy measure and related algorithms were made available in NPM (Table 3). This way, they can be used in any user-specific application.

Table 3 Algorithms implemented in the thesis were made available in NPM.

Algorithm	Package Name	URL
Louvain	*louvain-algorithm*	npmjs.com/package/louvain-algorithm
Infomap	*infomap*	npmjs.com/package/infomap
LLP	*layered-label-propagation*	npmjs.com/package/layered-label-propagation
GN Network Generator	*girvan-newman-benchmark*	npmjs.com/package/girvan-newman-benchmark
NMI	*normalized-mutual-information*	npmjs.com/package/normalized-mutual-information
Hamming Distance	*hamming-dist*	npmjs.com/package/hamming-dist

4.4 Visualization Frameworks

D3.js (SVG) allowed the user to visualize all networks used in the web application, as well as to represent the benchmark plots with the highest resolution possible (limited by density of pixels in the user's device). In terms of performance, it is the second fastest.

D3.js (Canvas) is the fastest among the analyzed frameworks. Nevertheless, when compared to the previous, the nodes and links from the network lose resolution.

Cytoscape.js is advisable when network analysis will be performed along with its representation. There is a considerable number of algorithms, including community finding ones, that can facilitate the study of the graph. The lack of working examples and the slow processing of network data made Cytoscape.js the less efficient. Representation of *S. aureus* SLV network and others with similar or higher number of nodes/edges overflows the web application.

4.5 *Staphylococcus aureus*

Louvain, Infomap and LLP algorithms were executed in the biggest component of the SLV graph obtained from the MLST dataset of *S. aureus*.

Louvain algorithm was executed using three modularity variation thresholds: 0.1, 0.02 and 0.01. It is possible to verify that high modularity partitions are rapidly achieved (2 passes needed to obtain final result), as predicted in Introduction. This characteristic makes modularity-based algorithms fast.

Using Infomap algorithm, several minimum description length thresholds were considered: 0.1, 0.02 and 0.01. For all values, every small dense group of nodes descending from the same ancestor was separated in different communities. Thus, it was not further considered.

LLP algorithm considered three values for the resolution parameter: 0, 0.5 and 1. Similarly to Infomap, it partitioned the network in several small and dense communities.

From the benchmark analyses of Louvain, Infomap and LLP against GN and LFR networks (Figure 28), it was predictable Louvain would perform better than the others, once it run in a network with well-defined communities. After adjusting the stopping parameter of Louvain, such that the communities identified were the most pertinent – nodes from densely connected regions descendent from the same ancestor belonging to the same community, a final partition was obtained (Figure 38). It is important to note that some clusters analogous to the one highlighted with a square were subdivided by the algorithm. One explanation is the presence of high recombination rates between strains of different sub-groups.

By uploading to PHYLOViZ Online the generated metadata file with the nodes labeled accordingly to the inferred communities, one may observe the original phylogenetic tree with the same communities identified.

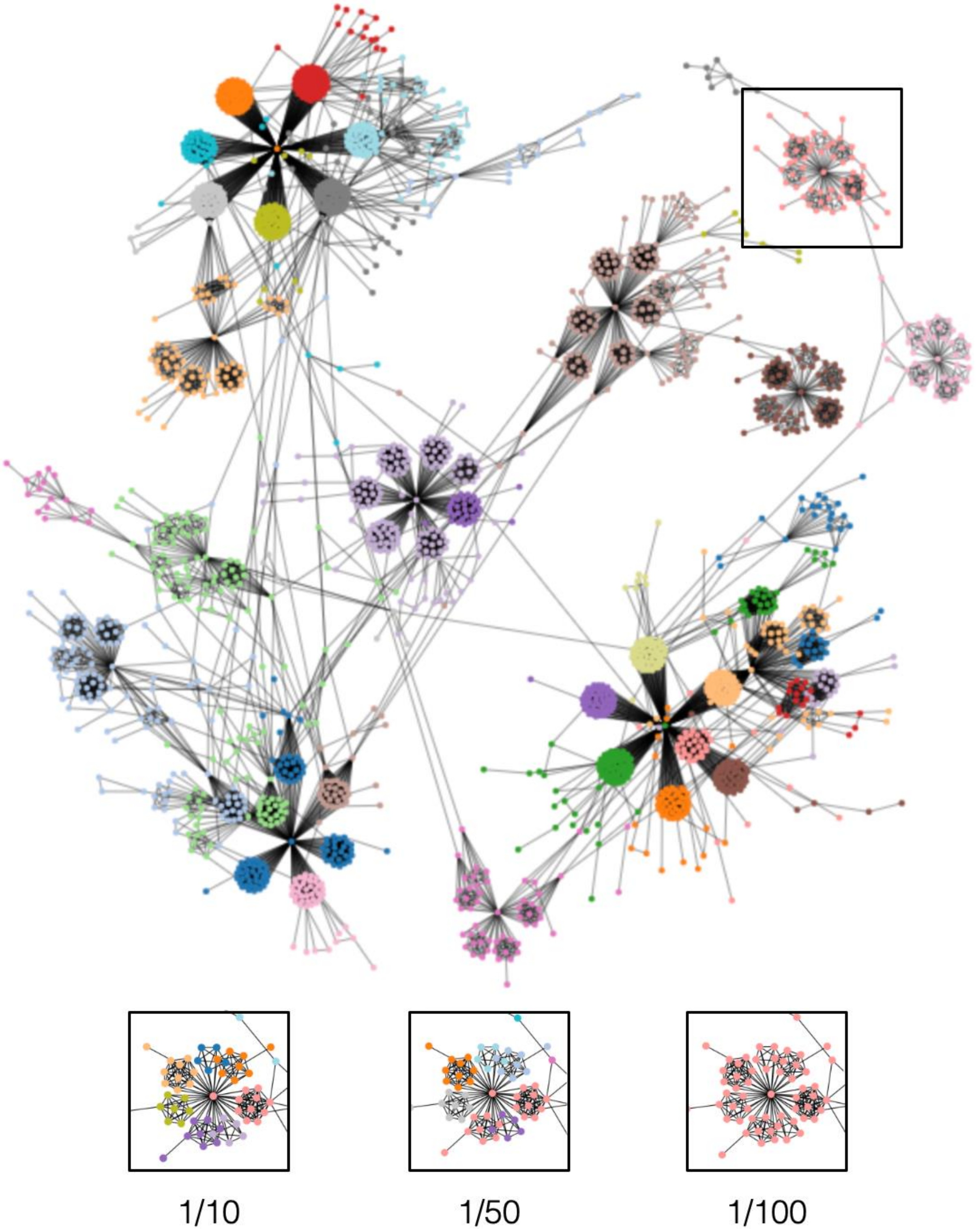


Figure 38 Partition obtained after running Louvain algorithm (considering different thresholds for the modularity variation) on CC 0 of *S. aureus* MLST SLV network. Represented using D3.js and Canvas.

5. Conclusion

The aim of the thesis was to implement three community finding algorithms in JavaScript – Louvain, Infomap and LLP. To benchmark them, in terms of accuracy and speed, against two synthetic networks – GN and LFR networks. Perform additional testing using an Amazon (disconnected), Zachary's Karate Club (small sized) and *S. aureus* MLST SLV (biological) network. Compare different visualization frameworks in terms of their tools and robustness to big data: D3.js (using SVG and Canvas elements) and Cytoscape.js. And, finally, it was intended to allow network visualization (before and after running community finding algorithms) and to plot all data obtained from benchmark tests in a reusable way.

Louvain, Infomap and LLP algorithms were implemented in JavaScript. Unless otherwise stated, next conclusions are valid for GN and LFR networks. In terms of speed, Louvain outperformed all others. In terms of accuracy, in networks with well-defined communities, Louvain was the most accurate. For higher mixing, LLP was the best. Contrarily to weakly mixed, it is advantageous to increase the resolution parameter in highly mixed GN networks. In LFR networks, higher resolution decreases the accuracy of detection, independently of the mixing parameter. The increase of the average node degree enhanced partitioning accuracy, suggesting detection by chance was minimized. It is computationally more intensive to generate GN networks with higher mixing or average degree, using the algorithm developed in the thesis or the LFR implementation. In the *S. aureus* MLST SLV network, Louvain was the fastest and the most accurate in detecting the clusters of 7 groups of strains evolved from the same ancestor. The lower the modularity variation threshold, the better the detection.

Along with the three community finding algorithms, GN benchmark network generator, NMI and hamming distance algorithms were made available in NPM with the names: *louvain-algorithm* (Louvain), *infomap* (Infomap), *layered-label-propagation* (LLP), *girvan-newman-benchmark* (GN benchmark), *normalized-mutual-information* (NMI) and *hamming-dist* (Hamming distance). The web application is hosted in the Heroku cloud platform. An image containing all the required modules to run the app in a local machine can be download from Docker Hub. Finally, all the implementations and a complete roadmap of the thesis was made available in GitHub. A Digital Object Identifier (DOI) was attributed to this repository using Zenodo.

One of the main difficulties encountered was running community finding algorithms in the user-side. A feature that would rise less privacy concerns. It was necessary to run the algorithms in the server, so that the application would not crash. It was not possible to achieve in Infomap implementation the same computational efficiency as predicted by its time complexity. The online visualization of networks with more than 10 000 nodes was unfeasible, even using the framework that showed the best performance – D3.js (using Canvas element).

During the next months, some of the tools developed in this thesis will be implemented in INSaFLU. New visualization frameworks for displaying phylogenetic trees, along with the uploaded metadata, will enhance the traceability of the evolutionary path among influenza strains. A geographical map dynamically varying with time will allow the user to precise the moment and location at which each sample was obtained and enhance flu surveillance during the critical season. An interactive histogram displaying each nucleotide belonging to a consensus sequence obtained from a sample of influenza virus with the SNPs highlighted, as well as with their effects in the phenotype of the strain specified. New flu surveillance dynamic reports will allow the visualization of increased quantities of data in a minimalistic way.

As future improvements, the implemented algorithms can be optimized in terms of memory and speed. Network and chart visualization may run faster in a more efficient implementation. The previous updates will make the web application to load quicker and run smoother. To benchmark community finding algorithms with a more representative set of networks (spanning a wider range of topological properties) would make possible to infer, with a higher certainty, if any of the community finding algorithms consistently performs better in phylogenetic networks. The integration of community finding functionalities developed in the thesis in a phylogenetic/surveillance-oriented analysis web application like PHYLOViZ Online or INSaFLU would be beneficial.

References

[1] "The world's most valuable resource is no longer oil, but data," The Economist, 6 May 2017. [Online]. Available: https://www.economist.com/leaders/2017/05/06/the-worlds-most-valuable-resource-is-no-longer-oil-but-data. [Accessed 28 May 2019].

[2] A.-L. Barabási, "Network Science Book," [Online]. Available: http://networksciencebook.com. [Accessed 15 May 2019].

[3] H. G. S. Patil, A. N. Babu and P. S. Ramkumar, "Non-invasive data acquisition and measurement in bio-medical technology: An overview," in *Maximizing Healthcare Delivery and Management through Technology Integration*, IGI Global, 2016.

[4] "Health," EUROPEAN DATA PROTECTION SUPERVISOR, [Online]. Available: https://edps.europa.eu/data-protection/our-work/subjects/health_en. [Accessed 8 June 2019].

[5] "Ten threats to global health in 2019," World Health Organization, [Online]. Available: https://www.who.int/emergencies/ten-threats-to-global-health-in-2019. [Accessed 4 June 2019].

[6] "Antimicrobial resistance," World Health Organization, 15 February 2018. [Online]. Available: https://www.who.int/en/news-room/fact-sheets/detail/antimicrobial-resistance. [Accessed 29 May 2019].

[7] Z. A. Memish, S. Venkatesh and A. M. Shibl, "Impact of travel on international spread of antimicrobial resistance," *International Journal of Antimicrobial Agents,* vol. 21, no. 2, pp. 135-142, 2003.

[8] "Top 10 Leading Causes of Death Globally," [Online]. Available: https://www.theatlas.com/charts/HkLaDreuW. [Accessed 12 May 2019].

[9] B. Ribeiro-Gonçalves, A. P. Francisco, C. Vaz, M. Ramirez and J. A. Carriço, "PHYLOViZ Online: web-based tool for visualization, phylogenetic inference, analysis and sharing of minimum spanning trees," *Nucleic Acids Research,* vol. 44, no. 1, pp. 246-251, 2016.

[10] Y. Motroa and J. Moran-Gilad, "Next-generation sequencing applications in clinical bacteriology," *Biomolecular Detection and Quantification,* vol. 14, pp. 1-6, 2017.

[11] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science,* vol. 286, p. 509–512, 1999.

[12] "Graph theory," Wikipedia, 21 May 2019. [Online]. Available: https://en.wikipedia.org/wiki/Graph_theory. [Accessed 29 May 2019].

[13] P. Erdős and A. Rényi, "On random graphs I.," *Publicationes Mathematicae (Debrecen),* vol. 6, pp. 290-297, 1959.

[14] E. N. Gilbert, "Random graphs," *The Annals of Mathematical Statistics,* vol. 30, no. 4, pp. 1141-1144, 1959.

[15] S. A. Cook, "The complexity of theorem proving procedures," in *Proceedings of the 3rd Annual ACM Symposium on the Theory of Computing (STOC'71)*, 1971.

[16] L. Fortnow, "The Status of the P versus NP Problem," *Communications of the ACM,* vol. 52, no. 9, p. 78–86, 2009.

[17] C. M. Institute, "P vs NP Problem," [Online]. Available: https://www.claymath.org/millennium-problems/p-vs-np-problem. [Accessed 15 May 2019].

[18] N. Viswarupan, "P vs NP Problem," Medium, 17 August 2017. [Online]. Available: https://medium.com/@niruhan/p-vs-np-problem-8d2b6fc2b697. [Accessed 15 May 2019].

[19] "P versus NP problem," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/P_versus_NP_problem. [Accessed 12 May 2019].

[20] J. Rosenberger, "P vs. NP poll results," *Communications of the ACM,* vol. 55, no. 5, p. 10, 2012.

[21] E. Ravasz, A. L. Somera, D. A. Mongru, Z. N. Oltvai and A. L. Barabási, "Hierarchical Organization of Modularity in Metabolic Networks," *Science,* vol. 297, no. 5586, pp. 1551-1555, 2002.

[22] M. E.J. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical review. E, Statistical, nonlinear, and soft matter physics,* vol. 69, 2004.

[23] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," *Proceedings of the National Academy of Sciences of the United States of America,* vol. 99, no. 12, pp. 7821-7826, 2002.

[24] M. E. J. Newman, "Fast algorithm for detecting community structure in networks," *Physical Review E,* vol. 69, no. 6, 2004.

[25] G. Palla, I. Derényi, I. Farkas and T. Vicsek, "Uncovering the overlapping community structure of complex networks in nature and society," *Nature,* vol. 435, p. 814–818, 2005.

[26] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, Introduction to Algorithms, Cambridge, Massachusetts: MIT Press , 2003.

[27] S. A. Rice, "The identification of blocs in small political bodies," *American Political Science Review,* vol. 21, p. 619–627, 1927.

[28] G. D. Bader and C. W. V. Hogue, "An automated method for finding molecular complexes in large protein interaction networks," *BMC Bioinformatics,* vol. 4, no. 2, 2003.

[29] G. Ouyang, D. Dey and P. Zhang, "Clique-Based Method for Social Network Clustering," *Journal of Classification,* 2017.

[30] R. D. Luce, "Connectivity and generalized cliques in sociometric group structure," *Psychometrika,* vol. 15, pp. 159-190, 1950.

[31] A. Kahng, J. Lienig, I. Markov and J. Hu, VLSI Physical Design: From Graph Partitioning to Timing Closure, Springer, 2011.

[32] Y.-Y. Ahn, J. P. Bagrow and S. Lehmann, "Link communities reveal multiscale complexity in networks," *Nature,* vol. 466, pp. 761-764, 2010.

[33] J. Yang and J. Leskovec, "Community-Affiliation Graph Model for Overlapping Network Community Detection," in *2012 IEEE 12th International Conference on Data Mining*, Brussels, Belgium, 2012.

[34] W. K. Hastings, "Monte Carlo Sampling Methods Using Markov Chains and Their Applications," *Biometrika,* vol. 57, no. 1, pp. 97-109, 1970.

[35] A. Lancichinetti, S. Fortunato and J. Kertesz, "Detecting the overlapping and hierarchical community structure of complex networks," *New Journal of Physics,* vol. 11, 2009.

[36] S. Gregory, "Fuzzy overlapping communities in networks," *Journal of Statistical Mechanics: Theory and Experiment,* vol. 2011, no. 02, 2011.

[37] V. D. Blondel, J.-L. Guillaume, R. Lambiotte and E. Lefebvre, "Fast unfolding of communities in large networks," *J. Stat. Mech. (2008) P10008,* p. 12, 2008.

[38] M. Rosvall, D. Axelsson and C. T. Bergstrom, "The map equation," *The European Physical Journal Special Topics,* vol. 178, no. 1, pp. 13-23, 2009.

[39] N. Raghavan, R. Albert and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," *Physical Review E 25th Anniversary Milestones,* vol. 76, no. 3, 2007.

[40] L. Šubelj, "Label propagation for clustering," in *Advances in Network Clustering and Blockmodeling*, New York, Wiley, 2018.

[41] P. Boldi, M. Rosa, M. Santini and S. Vigna, "Layered label propagation: a multiresolution coordinate-free ordering for compressing social networks," in *WWW '11 Proceedings of the 20th international conference on World wide web*, 2011.

[42] D. Harper, "Phylogeny," Online Etymology Dictionary, 2010. [Online]. Available: https://www.etymonline.com/search?q=Phylogeny. [Accessed 27 May 2019].

[43] "Phylogenetics: An introduction," EMBL-EBI Hinxton, [Online]. Available: https://www.ebi.ac.uk/training/online/course/introduction-phylogenetics/what-phylogenetics. [Accessed 27 May 2019].

[44] "Darwin's Tree of Life," Wayback Machine, [Online]. Available: https://web.archive.org/web/20140313124644/http://www.nhm.ac.uk/nature-online/evolution/tree-of-life/darwin-tree/. [Accessed 27 May 2019].

[45] R. R. Sakal, "PHENETIC TAXONOMY: Theory and Methods," *Annual Review of Ecology and Systematics,* vol. 17, no. 1, pp. 423-442, 1986.

[46] Z. Yang and B. Rannala, "Molecular phylogenetics: principles and practice," *Nature Reviews Genetics,* vol. 13, no. 5, p. 303–314, 2012.

[47] J. Adams, "DNA sequencing technologies," *Nature Education,* vol. 1, no. 1, p. 193, 2008.

[48] B. T. B. Staff, "10 Sequencing Moments That Mattered In 2015," Thermo Fisher Scientific, 21 December 2015. [Online]. Available: https://www.thermofisher.com/blog/behindthebench/sequencing-moments-that-mattered-2015/?socid=social_bitesizebio. [Accessed 27 May 2019].

[49] A. P. Francisco, M. Bugalho, M. Ramirez and J. Carrico, "Global optimal eBURST analysis of multilocus typing data using a graphic matroid approach," *BMC Bioinformatics,* vol. 10, no. 152, 2009.

[50] "cgMLST - Genome-wide gene by gene microbial typing," Ridom Bioinformatics, [Online]. Available: https://www.ridom.de/seqsphere/cgmlst/. [Accessed 12 May 2019].

[51] B. Neumann, K. Prior, J. K. Bender, D. Harmsen, I. Klare, S. Fuchs, A. Bethe, D. Zühlke, A. Göhler, S. Schwarz, K. Schaffer, K. Riedel, L. H. Wieler and G. Werner, "A Core Genome Multilocus Sequence Typing Scheme for Enterococcus faecalis," *Journal of Clinical Microbiology,* vol. 57, no. 3, 2019.

[52] K. A. Jolley, C. M. Bliss, J. S. Bennett, H. B. Bratcher, C. Brehony, F. M. Colles, H. Wimalarathna, O. B. Harrison, S. K. Sheppard, A. J. Cody and M. C. Maiden, "Ribosomal multilocus sequence typing: universal characterization of bacteria from domain to strain," *Microbiology,* vol. 158, no. 4, pp. 1005-1015, 2012.

[53] L. C. Kingry, L. A. Rowe, L. B. Respicio-Kingry, C. B. Beard, M. E. Schriefer and J. M. Petersen, "Whole genome multilocus sequence typing as an epidemiologic tool for Yersinia pestis," *Diagnostic microbiology and infectious disease,* vol. 84, no. 4, pp. 275-280, 2016.

[54] "SNP," Nature Education, 2014. [Online]. Available: https://www.nature.com/scitable/definition/single-nucleotide-polymorphism-snp-295. [Accessed 28 May 2019].

[55] E. J. Feil, B. C. Li, D. M. Aanensen, W. P. Hanage and B. G. Spratt, "eBURST: Inferring Patterns of Evolutionary Descent among Clusters of Related Bacterial Genotypes from Multilocus Sequence Typing Data," *Journal of Bacteriology,* vol. 186, no. 5, p. 1518–1530, 2004.

[56] A. P. Francisco, C. Vaz, P. T. Monteiro, J. Melo-Cristino, M. Ramirez and J. A. Carriço, "PHYLOViZ: phylogenetic inference and data visualization for sequence based typing methods," *BMC Bioinformatics,* vol. 13, no. 87, 2012.

[57] B. Ribeiro-Gonçalves, A. P. Francisco, C. Vaz, M. Ramirez and J. Carriço, "PHYLOViZ Online: web-based tool for visualization, phylogenetic inference, analysis and sharing of minimum spanning trees," *Nucleic Acids Research,* vol. 44, no. W1, pp. 246-251, 2016.

[58] M. PANAHI, "Louvain," Node.js Package Manager, [Online]. Available: https://www.npmjs.com/package/louvain. [Accessed 5 June 2019].

[59] A. Lancichinetti and S. Fortunato, "Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities," *Physical review. E, Statistical, nonlinear, and soft matter physics.,* vol. 80, 2009.

[60] J. Yang and J. Leskovec, "Defining and Evaluating Network Communities based on Ground-truth," in *Proceedings of 2012 IEEE International Conference on Data Mining (ICDM)*, 2012.

[61] W. Zachary, "An Information Flow Model for Conflict and Fission in Small Groups," *Journal of anthropological research,* vol. 33, 1976.

[62] "Staphylococcus aureus MLST Databases," PubMLST, 5 June 2019. [Online]. Available: https://pubmlst.org/saureus/. [Accessed 5 June 2019].

[63] A. Lancichinetti, S. Fortunato and F. Radicchi, "Benchmark graphs for testing community detection algorithms," *Physical review. E, Statistical, nonlinear, and soft matter physics.,* vol. 78, no. 4, 2008.

[64] A. Amelio and C. Pizzuti, "Is Normalized Mutual Information a Fair Measure for Comparing Community Detection Methods?," in *Proceedings of the IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, Paris, 2015.

[65] T. M. Cover and J. A. Thomas, Elements of Information Theory, Second Edition, New Jersey, USA: John Wiley & Sons, 2005.

[66] "Mutual information," Wikipedia, 26 May 2019. [Online]. Available: https://en.wikipedia.org/wiki/Mutual_information. [Accessed 27 May 2019].

[67] M. Bostock, "D3 Data-Driven Documents," D3 Data-Driven Documents, [Online]. Available: https://d3js.org/. [Accessed 6 June 2019].

[68] "Cytoscape.js," Cytoscape, [Online]. Available: http://js.cytoscape.org/. [Accessed 6 June 2019].