Community Finding in Phylogenetic Networks

Luís Rita

Thesis to obtain the Master of Science Degree in

Biomedical Engineering

Examination Committee

Chairperson: Prof.

Supervisor: Prof. Alexandre Francisco

Co-Supervisor: Prof. João Carriço

Members of the Committee: Prof.

Month 2019

Abstract

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

The aim of this thesis was to develop an improved version of PHYLOViZ Online and INSaFLU by implementing community finding algorithms, as well as, adding new tools for data visualization.

Louvain, Infomap, Label Propagation and Layered Label Propagation were implemented, tested and benchmarked against multiple databases with ground-truth communities containing millions of nodes and edges. New cutting-edge interfaces were built using Cytoscape.js in order to display the results obtained after running these tools.

For the first time, these algorithms were implemented in JavaScript (next generation and scalable web programming language). Along with a new graph displaying tool – Cytoscape, intensive data analysis is now possible to be performed in any browser.

PHYLOViZ and INSaFLU, along with the developed tools, provide a brand-new insight into the field of microbiology. Based on microbial typing, they now allow us to determine how similar multiple strains are and to infer a possible evolutionary path. To trace the source of an outbreak, infer about the pathogenicity/virulence of a recently discovered strain or, based on serotype data (Fig. 1) or other phenotypic data, to determine the most suitable strains to be targeted for vaccination. These tools are intended to assist distinct health professionals, including doctors and bioinformaticians, and are designed to enable medical and research purposes.

.

Resumo

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

The aim of this thesis was to develop an improved version of PHYLOViZ Online and INSaFLU by implementing community finding algorithms, as well as, adding new tools for data visualization.

Louvain, Infomap, Label Propagation and Layered Label Propagation were implemented, tested and benchmarked against multiple databases with ground-truth communities containing millions of nodes and edges. New cutting-edge interfaces were built using Cytoscape.js in order to display the results obtained after running these tools.

For the first time, these algorithms were implemented in JavaScript (next generation and scalable web programming language). Along with a new graph displaying tool – Cytoscape, intensive data analysis is now possible to be performed in any browser.

PHYLOViZ and INSaFLU, along with the developed tools, now provide a brand-new insight into the field of microbiology. Based on microbial typing, they now allow us to determine how similar multiple strains are and to infer a possible evolutionary path. To trace the source of an outbreak, infer about the pathogenicity/virulence of a recently discovered strain or, based on serotype data (Fig. 1) or other phenotypic data, to determine the most suitable strains to be targeted for vaccination. These tools are intended to assist distinct health professionals, including doctors and bioinformaticians, and are designed to enable medical and research purposes.

Acknowledgments

Text

Table of Contents

List of figures

No table of figures entries found.

List of Tables

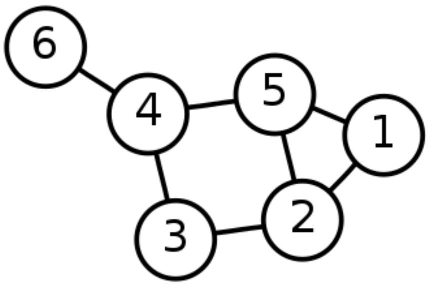
List of Acronyms

ABC	Activity-Based Costing

1. Introduction

Networks

These structures are ubiquitous in many phenomena of the present world. In an increasing order of magnitude (adopting an anthropocentric point of view): protein interactions, neurons organization/communication, human interaction and societal behavior are often explained by current Network Theory.



One way to represent and analyze these structures is using graph theory.

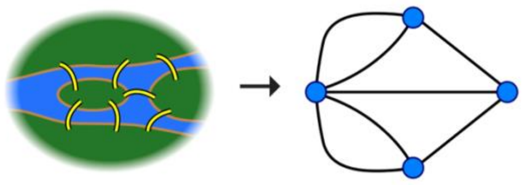
Graph Theory

The building blocks of graphs are nodes and edges.

Associated to these simple structures is a complex theory that has direct implications in areas like computer science and mathematics. Although, other studies are also common among physics, chemistry, biology and, even, linguistics or social sciences.

Seven Bridges of Konigsberg, wrote in 1736 by Leonhard Euler, is often regarded as the 1st paper in the history of graph theory. The city Konigsberg in Prussia (now Kaliningard, Russia), lay in both sides of Pregel River and includes 2 islands – Kneiphof and Lomse. These were connected to the 2 mainland portions by 7 bridges. The problem was to devise a walk such that one could cross every bridge exactly once. In fact, it retracts how graph theory may be used to solve many real-world problems.

In spite of the example discussed above, this problem could be easily solved by hand. In other words, we could easily find a solution to this and other similar complexity problems using a simple brute-force approach. By trying every single possibility, we could conclude the problem has no solution. Having said this, the power of this theory arises when one starts considering big numbers of possibilities/combinations, such that even the most powerful computers are not able to



solve in a short time scale. This leads us to a class of problems which are called NP problems.

NP, P, NP-hard…

Big O Notation

While analyzing algorithmic performance, one should consider both time and space complexity. This is especially important when there is limited computational power and when some of the problems require the analysis of big quantities of data. For the same problem, sometimes one may find its solution in a short time scale or, in others, it may take longer than a life-time. For this reason, it was important to develop a way of upper bond the time of execution of algorithms and to identify what is the expected value for this. This resulted in the common use of Big O notation.

Depending on the number of instructions the algorithm should execute and the input/output size values, it will have different run times. In fact, the time of execution is often correlated with the size of the data to analyze. This relation is often linear, logarithmic, quadratic, cubic or exponential. Here is an ordered list, in terms of time of execution, of the most common algorithms, along with some examples:

N – when it is required the analysis and manipulation of an array of data;

Log – often means divide-and-conquer algorithms were used. This happens when a main problem is decomposed in many smaller ones;

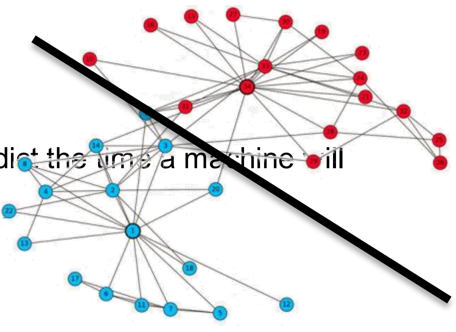
N^2 – adjacency matrices representations often imply the use of quadratic algorithms. Cubic or higher order degrees are valid for analogous data structures of increased complexity;

2^N – few algorithms are useful with this complexity. Nevertheless, situations where brute-force approaches are the only ones available, it may be considered (for limited datasets).

Definition

A function f(N) is said to be O(g(N)) if there exists constants k and N_0, such that $f(N) < k\, g(N_0)$ for all $N > N_0$.

After finding the complexity of a given algorithm, we may try to predict the time a machine will take to execute it based on its ROM and RAM memory parameters.



Community Finding

With the advent of new ways of data acquisition in areas like social or biological sciences, improved procedures to handling it are needed. Not only are needed new algorithms that scale well for big data, but also ones that can precisely enough identify similarity among sets of nodes (the fundamental units in networks).

First community finding experiments remote 1927, when Stuart Rice tried to identify communities in political bodies, using voting patterns. Although, it was only in 2002 when Girvan and Newman used the Zachary Club as benchmark test that community detection gained an increased interest among the scientific community. As soon as the paper was published, the number of citations in the theme exploded. Later on, Erdős and Rényi study of random networks and the discovery of the scale-free property in real ones, turn community finding a central problem in network theory.

Before start analyzing in detail community finding, it will be introduced a classical social and biological example.

Zachary's Karate Club

In the past, there was a Karate Club with 34 members. Among those, there was a president and an instructor. In figure _, each node represents a member and, when an edge is present, this means there is a relationship outside of the club between both members. There is a total of 78 pairwise links in the network.

The interest in this dataset is driven by a single event: a conflict between the president and the instructor give rise to a separation of the members in 2 groups. One supporting the president and the other the instructor. Today, community finding algorithms are often tested upon their ability to separate this 2 groups.

This case was documented by Wayne Zachary.

A community can be defined as a structure (present in a network) that rests in 4 hypotheses:

1. Fundamental Hypothesis;

It states that communities are uniquely encoded by their wiring diagram. The algorithms we will further present, pretend to detect those by analyzing the whole network.

2. Connectedness and Density Hypothesis;

A community can be described as a dense and connected subgraph. Depending on other criteria, we may use stricter or looser rules to defined them.

3. Random Hypothesis;

Random networks do not have communities.

4. Maximal Modularity Hypothesis.

A partition with higher modularity offers a better representation of the network's community structure. Mathematically, it is defined as: $M = \sum_{c=1}^{n_c} \frac{l_c}{L} - (\frac{k_c}{2L})^2$. This quality measure will be analyzed in the next pages.

The previous hypothesis pretend to give you an intuition about what a community should look like. This means, they are not strict enough to allow one single definition to these structures. In fact, many were already proposed. From the more rigid definition to the looser, we present them next:

Maximum Clique

In the first references to communities, it was required that all nodes would need to be connected to all others inside the same community. A clique is synonymous of a complete subgraph.

This was later considered a too rigid definition, once most of the real networks present triangles, but larger cliques than that are rare. Moreover, none of the communities present in the previous networks could be classified that way.

Strong Community

To overcome the rigidity of cliques 2 other definitions of community were proposed: a strong and weak community.

Introducing first the definition of a node's inner and external degree, the former corresponds to the number of connections with nodes inside the community (k_i^{int}) and the latter outside of it (k_i^{ext}).

The definition of strong community states that the inner degree of every node should be greater than the corresponding external degree:

$k_i^{int}(C) > k_i^{ext}(C)$

Weak Community

On the other hand, a weak community relaxes the previous definition in the way it requires the sum of the internal degree of every node from the community to be greater than the corresponding external degree.

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

Other Definitions of Communities

In some cases, the frequency of ties among members of the same group is considered to define community. This states that every element in the group must have at least k connections to other elements of the group.

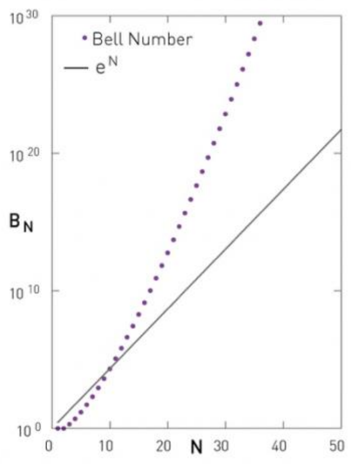
Another definition only considers nodes in the same group if they are, at most, at a distance n from other elements in the same group.

Source: http://snap.stanford.edu/class/cs224w-2015/slides/12-community.pdf

Before presenting community finding algorithms, one could reasonably ask how feasible would be to use a brute force approach to inspect every single possible community in a network. The answer is provided by the Bell Number:

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

As we can observe in figure _, the number of possible partitions in a network increases faster than exponentially as N increases.



The main problem resides in partitioning the original graph without having to define, a priori, the number of communities and their size. At first, we will be looking at algorithms whose mode of operation consists in attributing to each node 1 single community. Then, shared nodes will be allowed. Thus, retracting better the overlap existence present in most real networks. This motivates the introduction of Hierarchical Clustering.

Source: http://snap.stanford.edu/class/cs224w-2015/slides/12-community.pdf

Hierarchical Clustering

To uncover the community structure of a network, algorithms with, at most, polynomial running times are needed.

Inside the category Hierarchical Clustering, there are agglomerative (Ravasz Algorithm) and divisive procedures (Girvan-Newman Algorithm).

<u>Ravasz Algorithm</u>

This algorithm includes 4 steps that should be considered sequentially:

1. Definition of a Similarity Matrix

This matrix can be based on evolutionary distances among nodes or in the number of common neighbors and . Thus, in the latter case, we may define each entry as:

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

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

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

And the maximum value will be reached if there is a connection between both nodes and they have the same neighbors:

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

2. Group Similarity Criteria

After joining the first 2 most similar nodes in the network, it will be needed to compare this formed cluster with the remaining elements (or later, with other clusters). 3 different approaches may be considered:

- Single Linkage Clustering

Similarity between 2 communities is calculated using the 2 most similar elements.

- Complete Linkage Clustering

Analogous to the previous but using as reference the 2 most dissimilar nodes present in each cluster.

- Average Linkage Clustering

Considers the average distance of every possible node combination in the 2 clusters.

3. Hierarchical Clustering Procedure

Having defined an initial similarity matrix and a similarity criterion to compare different clusters during its execution, the algorithm will sequentially execute the following set of instruction:

- Assign a similarity value to every node present in the network;

- Find the most similar community pair and join both nodes. Then, similarity matrix is updated based on group similarity criteria;
- Last step is performed until all nodes belong to a single community.

4. Dendrogram Cut

At the end of the algorithm execution, we will end up with a single tree, joining every possible node. Although we are able to visualize which nodes the algorithm considered closer to each other, it does not tell us what is the best possible partition in the network.

In fact, one way to represent the communities better assigned in the dendrogram is cutting it at a certain level. The issue is that, until now, we have now clue about what the best partition is. To solve this problem, we will further introduce a new concept called modularity that pretend to overcome this problem.

Combining all the steps previously discussed, we can estimate the complexity of the algorithm:

- Step 1: every node will have its similarity calculated against every other node. Thus, complexity should be in the order of O(N^2), being N the number of elements in the network;
- Step 2: every community will be compared against the others and the overall complexity should also be O(N^2);
- Steps 3 & 4: using a convenient structure to represent data, the construction of the dendrogram, in a worst-case scenario, may be done in O(Nlog(N)) steps.

$$O(N^2) + O(N^2) + O(NlogN) = O(N^2)$$

Despite we will encounter, in the next chapters, faster algorithms, this one is already significantly faster than the brute force approach which would require O(e^N) operations.

<u>Girvan-Newman Algorithm</u>

The mode of operation of this algorithm is, in some way, the contrary to the previous presented. We start with a network which will see their edges removed based on centrality criteria, until none remains.

1. Definition of Centrality

In Girvan-Newman algorithm we pretend to iteratively identify the pair of nodes that is most probable to belong to different communities. Centrality matrix will reflect this. As higher the matrix entry, as probable it will be each node to make part of different communities. This matrix can be calculated using 2 different approaches:

- Link Betweenness

Modularity will be proportional to the number of shortest-paths between every node that include the link between node i and j.

Link betweenness complexity will be O(NL) or O(N^2), in case of sparse networks ($N \approx L$).

- Random-Walk Betweenness

After picking at random a node m and n, a random path between those will be traced. After picking every possible combination of nodes, the average number of times the link (i,j) was crossed is recorded. The x_{ij} will be proportional to this value.

The first step of this calculation will require an inversion of a N*N matrix, thus computational complexity will be O(N^3).

Secondly, the average flowing over all node pairs will require O(LN^2) steps.

In case of a sparse network, the overall complexity will be O(N^3).

2. Hierarchical Clustering Procedure

Upon picking one of the 2 centrality criteria, the following set of operations will be performed by the algorithm.

- It begins by calculating the centrality for every node pair;
- The link with highest centrality is removed from the network. In case of a tie, one of them is randomly picked;

- Network and the centrality matrix are recalculated;
- The 2 previous steps are repeated until no link is left in the network.

3. Dendrogram

Similarly to the Ravasz algorithm, Girvan-Newman algorithm does not predict what is the best partition among the ones deductible from the obtained dendrogram. Again, the modularity function may help us determining the optimal cut. This way, this value should be calculated for every possibility present in the tree.

Regarding the complexity of the algorithm, the limiting step is the centrality calculation. This should be performed in O(LN), in case we choose the most efficient method – link betweenness. The overall complexity obtains by multiplying the previous by the number of times the centrality matrix has to be calculated – L (all links will end up removed). This means O(L^2N) or O(N^3) is the total complexity for a sparse network.

As a final statement regarding these 2 methods, one may ask whether a hierarchical structure is really present in real networks or if the algorithms are imposing it. This means, are expecting to find nested modules inside bigger ones? Is it possible to assess if a network possesses this structure a priori?

One way to check hierarchical modularity is by using the clustering coefficient. In fact, it is dependence with node's degree, let us identify whether such pattern is present. In many real networks this phenomenon is present: scientific collaboration, metabolic and citation network. The following equation predicts its distribution: $C(k)\sim k^{-1}$. As expected, when randomized, this modularity structure disappears. This resembles the case of random networks (Erdős-Rényi or Barabási-Albert models), where such structure is not present.

This section finishes by stating once again that hierarchical clustering allowed us to join network members based on their similarity, but the corresponding algorithm did not give us any clue regarding which network partition may be the best. This will thoroughly analysed in the next section.

Modularity Maximization

Based on the hypothesis a random network should not display community structure, the modularity concept was built. In fact, it compares a partition of the network under study, with the same but, previously, randomly rewired.

Starting by introducing some notation to simplify the formal theory present next. We are considering a network with N nodes and L links, and a partition of it in n_c communites, each with N_c nodes and L_c links:

$$c = 1, \dots, n_c$$

Modularity measures the difference between the real network wiring of a subgraph C_c with the analogous but randomly wired:

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

p_{ij} results from randomly rewiring the whole network, although maintaining the expected degree of each node. Consequently, this probability is calculated the following way:

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

Analyzing equation _, we conclude that if $M_c = 0$, then the network is randomly wired (thus, not presenting any underlined structure). If $M_c < 0$, C_c does not form a community as well.

Manipulating equation _, we end up with a simpler formula to cluster's modularity:

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

Generalizing these ideas to the complete network, we identify the set of communities that better categorize the graph. This way, modularity is the sum over all communities of the previous measure:

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

We now analyze modularity in terms of the values it can take. Just like before, we can extrapolate the assumption that higher the modularity, higher the quality of the partition of the whole network. Moreover, it can take positive, zero or negative values. Whenever we define the whole network as part of one single community:

$$(k_c = 2L \wedge L_c = L) \Rightarrow M = 0$$

In the extreme case each node is a single community, we will obtain $L_c = 0$ and the modularity will take negative values. Consequently, none of these previous cases identifies a community.

There are several algorithms that use modularity to define communities inside a network. The first one that will be presented next is a greedy approach of the maxima modularity problem.

Greedy Algorithm

This algorithm aims to maximize modularity at each step. It proceeds the following way:

1. At the beginning each node is considered a different community;
2. Every connected pair of nodes in the network is analyzed and the network's modularity determined (considering this change). The duo that maximizes modularity is chosen and joined as part of the same community. Modularity is recorded;
3. Previous step until only 1 community remains;
4. Network partition with higher modularity is chosen.

In terms of computational complexity, we should analyze the impact of each step in the overall execution. Since modularity difference can be calculated in constant time, step 2 requires O(L) calculations. After community merging, the whole the matrix representing the network is updated in a worst-case scenario of O(N). Each merging will take place N-1 times. Thus, overall complexity ends up: $O\big((L+N)N\big)$ or $O(N^2)$ on a sparse graph.

Although modularity is computationally suited and an accurate way of community detection, there are a couple of pitfalls that should be highlighted. Resolution limit and modularity maxima are 2 issues one should be aware when using these algorithms.

Resolution Limit

Before precising the fundamental points of this limit, it is necessary to introduce a new way to compare how modularity varies after merging of 2 communities:

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

The means that 2 communities should be united whenever:

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

Assuming the number of links between A and B is at least 1:

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

In other words, every time there is at least 1 link between 2 communities, they are going to be linked. Even this does not maximize modularity of the network. This issue takes place when the size of each community is lower than $\sqrt{2L}$.

Assuming $k_A \approx k_B = k$,

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

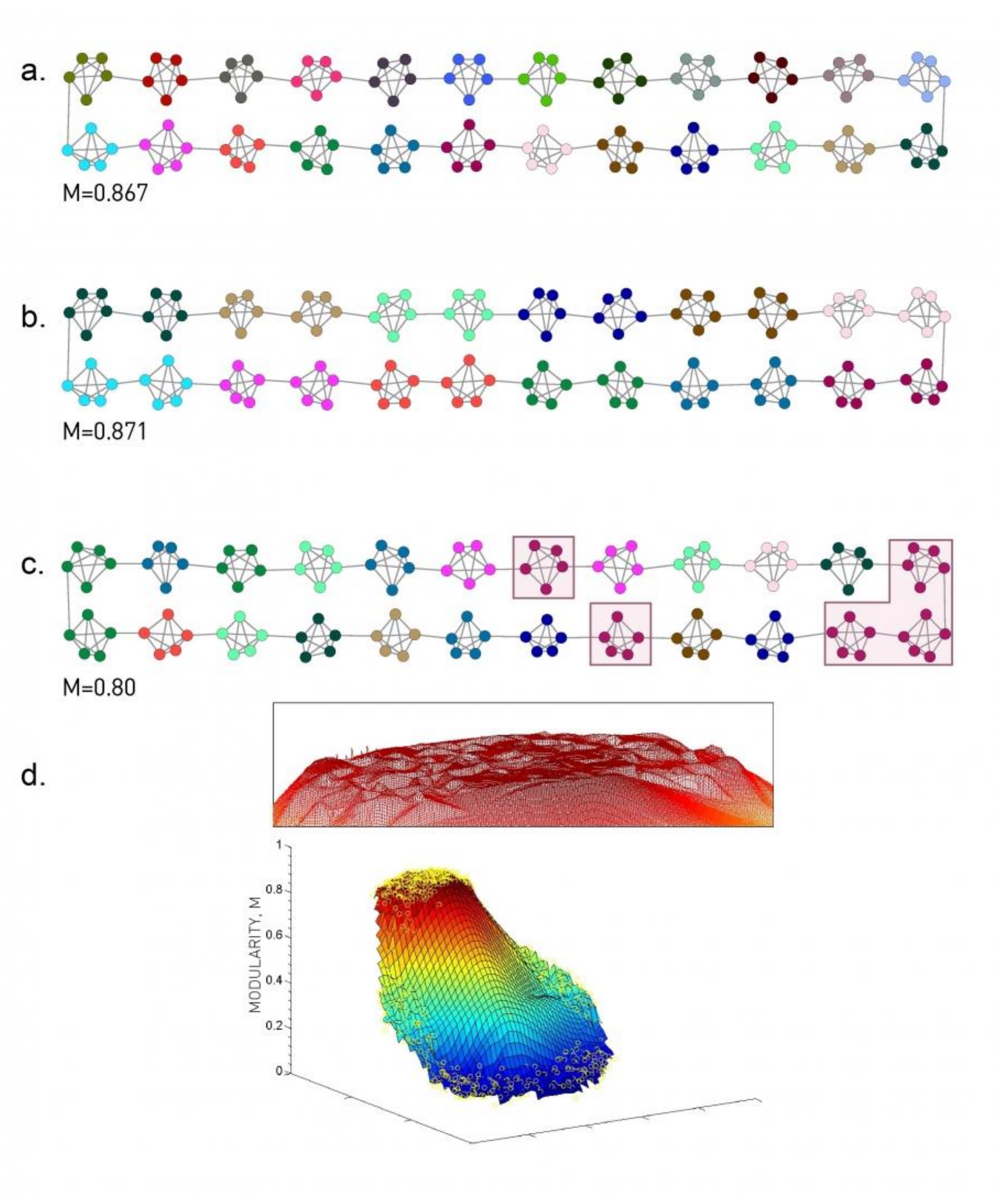
This stops us from indiscriminately using modularity in real networks because many communities may be lower than the resolution limit.

One way to overcome this limitation would be to subdivide large communities obtained by modularity optimization. Decreasing N would increase our resolution of the network.

Modularity Maxima

All algorithms presented until now, show that higher modularity correspond to a better partition of the network in communities. Although, in some graphs, completely different partitions may generate very similar modularity values. This may become a particularly relevant issue when the number of nodes increases. Turning progressively harder to separate network's best partition from the lower quality ones.

The next example clarifies this issue.



Intuitively, one says the best partition is the one that groups each cluster of 5 nodes in different communities. Although, modularity increases when 2 adjacent communities are merged. This complies with the vision the plateau in the modularity graph may distort our choice of the best partition.

Still regarding the network present in Fig. _, another important feature should be highlighted. In fact, if we assign random clusters in a communities, even the ones that are not directly connected may produce modularity values similar to the optimal partition.

Modularity optimization algorithms are part of a larger set of problems that are solved by maximization of quality functions. After introducing the previous algorithms which were part of the genesis of community finding, we now go through new up-to-date, accurate and scalable approaches.

Clique-Based Methods

Until this point, many algorithms intended to resolve communities in graphs were already discussed. One important issue that was not considered is that some nodes are part of different communities and, consequently, cannot be correctly categorized in only one. This led to the arousal of 2 algorithms that took this in consideration: clique percolation and link clustering.

Clique Percolation Algorithm

Also called CFinder, it aims to form communities based on adjacent cliques.

It proceeds the following way:

1. An N*N matrix is created. Each entry O_{ij} corresponds to the number of shared nodes between i and j k-cliques. k is picked such that we get the richest community structure (widely distributed cluster sizes);
2. If the number of shared nodes between 2 cliques is k-1, then they are joined. This step is performed globally to the whole network;
3. Last step is performed until no more adjacent cliques are possible to aggregate in the same community. In general terms, maximal cliques can be found using the following pseudo-function:

```
$function\ Expand(R, Q)\ \{$
$while\ (R\ \neq \{\})\ \{$
$p\ =\ vertex\ in\ R$
$Q_p\ =\ Q \cup \{p\}$
$R_p = R \cap \Gamma(p)$
$if\ (R_p \neq \{\})\ \{$
$Expand(R_p, Q_p)$
$\}\ else\ \{$
$return\ Q_p$
$\}$
$R = R - \{p\}$
$\}$
$\}$
```

Source: http://snap.stanford.edu/class/cs224w-2015/slides/12-community.pdf

Nevertheless, it is important to state that even random networks may present big connected components such those detected by the algorithm. To assess if a community is present by chance, the analysis of a random rewired network should be considered:

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

Thus, whenever the probability of connection between nodes exceeds p_c (which varies accordingly to the size of the network), it is expected to contain big components.

Generally, finding cliques in a network grows exponentially with N. Although, once we are not interested in finding maximum cliques, but k-cliques, we may identify those in polynomial time. If, however, large cliques are expected in the graph, O(e^N) algorithms may perform better.

Link Clustering Algorithm

Similarly to Rasvaz algorithm, we apply hierarchical clustering to partition the network in different communities but attending that each node can be present in more than 1. Apart from this, the other key dissimilarity is that instead of node clustering, links are grouped accordingly to their similarity.

One advantage of using a link clustering method is that, contrary to nodes, links usually are restricted to a single community.

Before introducing the algorithm, it is relevant to define a measure of proximity between each pair of links. So, each entry of the similarity matrix will be calculated using the following formula:

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

$n_+(i)$ represents the number of nodes to which i is connected (including itself). Therefore, S is a relative measure of the proportion of nodes shared by the 2 non-common nodes present in each link.

The overall procedure can be resumed in the following steps:

1. Calculation of a similarity matrix for each node pair;
2. Cluster links with higher similarity;
3. Apply single-linkage and merge communities with largest similarity link pairs;
4. Cut the generated dendrogram using, for instance, maxima modularity.

The algorithm is divided in 2 important steps. The first is the calculation of a similarity matrix and the second is hierarchical clustering. Each will contribute differently to the overall complexity:

- The first calculation is inherently dependent upon the number of links each node belonging to the pair under comparison has. This will require max(ki,kj) steps. In case we are dealing with a scale-free network, the calculation complexity of this phase will be $O(N^{\frac{2}{\gamma-1}})$. Note this estimate is determined by the size of the largest node k_{max};
- Analogously to Rasvaz algorithm, hierarchical clustering step will run not in $O(N^2)$, but in $O(L^2)$ because the links are the structures being compared. Although, in sparse graphs, we can substitute this by $O(N^2)$.

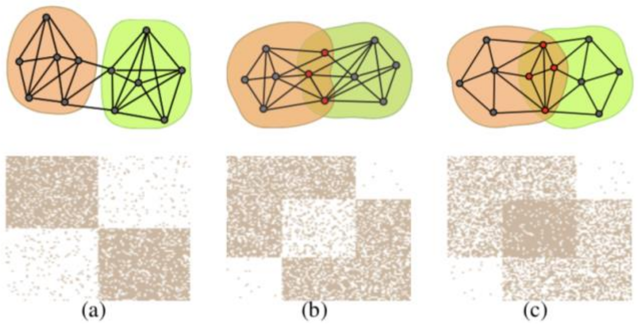
The final complexity results from the combination of the 2 previous steps:

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

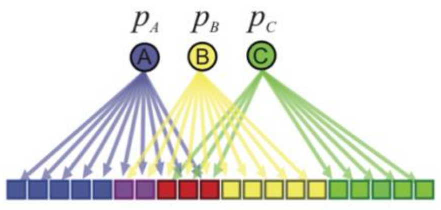
Statistical Inference

Community-Affiliation Graph Model

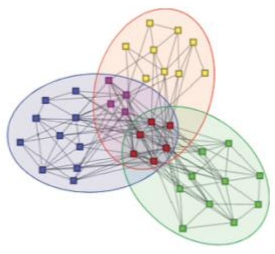
Until now, we explored 3 main approaches for community finding: hierarchical clustering, modularity maximization and clique-based methods. Statistical inference is the last class of algorithms still not presented.



Yang et al [?] proposed in 2012 a new model-based approach. They defined a new way of representing network's structure. This representation includes not only the nodes present in the original graph but the communities they should belong to. Each cluster (community) has a probability associated which translates in the probability of 2 nodes, sharing the same cluster, to be linked by an edge. After, they proposed an algorithm to identify such structure out of any input network. This would lead to a new community finding method.



(a) Community Affiliation Network


(b) Overlap

Model:

In fact, this is how Affiliation Graph Model distinguishes from any other. Not only it accepts community overlapping, but it also states that nodes present in those areas, are more likely connected. This is intuitively a consequence of their assumption: 2 nodes belonging to some community have a probability to connect defined by this same structure. Thus, sharing more communities implies a greater probability of connection. Before going through the mathematical basis, we should be aware of the most critical notation of the model:

Let $B(V, C, M)$ be the bipartite graph that related each node for a given number of communities. V is the set of nodes, C is the community set 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 $p(u, v)$. This probability is defined below:

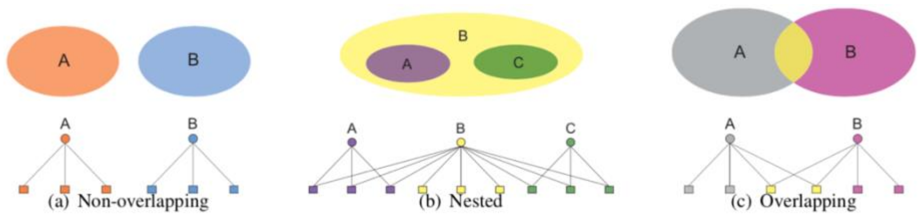
$$p(u, v) = 1 - \sum_{k \in C_{uv}} (1 - p_k)$$

As predicted, it is directly correlated with the number of shared communities. In fact, as higher is this number, lower is $\sum_{k \in C_{uv}} (1 - p_k)$ and closer to one the final probability. Formally:

$C_{uv} \subset C$, C_{uv} is the set of communities both nodes share.

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

A natural consequence of what we discussed above is that, contrarily to previously discussed algorithms' assumptions, regions of overlapping have an increased number of edges. This conclusion is verified by simply checking any kind of social or biological network. Algorithms discussed before would more likely identify a community in overlapping regions or join together both clusters as a single community, than identify this generalized particularity of most networks.



As a consequence of the flexibility of the model (in terms of node connections), we are able to use it to represent a wide number of network's topologies, ranging from one containing non-overlapping communities to nested or overlapping clusters. Graphical representation in Fig. _.

Algorithm:

Generally, taken an unknown network we pretend to detect its communities by identifying the pattern we identify before. Thus, the goal will be to maximize the likelihood $L(B,\{p_c\}) = P(G|B,\{p_c\})$ of the underlying and unlabeled graph:

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

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 and vice-versa.

[Updating $\{p_c\}$]

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

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

Although, this problem is not convex. This means one should transform it into a convex one. We do that by taking the logarithm of the likelihood and then proceeded to the following variable substitution:

$$e^{-x_k} = 1 - p_k$$

Consequently, the likelihood maximization problem has now a different shape:

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

And the constraint (?) is now:

$$x_c \geq 0$$

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

[Updating B]

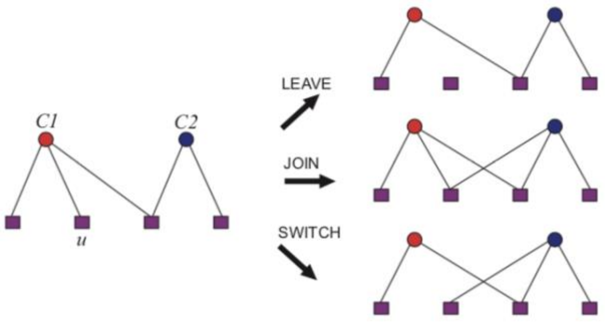
The remaining part of the optimization process focuses on finding the suitable B. For that, an iterative approach will be used again, but now in a different scope such as B is not a quantifiable property as $\{p_c\}$. A method called Metropolis-Hastings will handle this issue. Stochastically, it updates B, whenever a certain transition maximizes the likelihood’s ratio:

$$\max(1, \frac{L(B', \{p_c\})}{L(B, \{p_c\})})$$

Metropolis-Hastings will randomly update the digraph using the following type of transitions:

- Leave – Relating to Fig. _, node u is removed from the actual community. Thus, the final set of edges can be expressed as: $M' = M \backslash \{(u, c)\}$. Assuming, $\{(u, c)\} \in M$;
- Join – This action is the opposite of the previous. Node u joins community c. Update version of M, now, turns: $M' = M \cup \{(u, c)\}$. Assuming, $\{(u, c)\} \notin M$;
- Switch – 1 node changes its associated community, this can be described as: $M' = (M \backslash \{(u, c_1)\}) \cup \{(u, c_2)\}$. Assuming, $\{(u, c_1)\} \in M$ and $\{(u, c_2)\} \notin M$.

Resuming, a new B is first generated. Then, a large number of steps are performed. 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 submitted to the transition in iterate i.



Regarding the computational complexity of AGM algorithm. It is important to state that it is not suitable for networks with millions of nodes, just like clique-based methods presented before cannot. As stated by the authors, the convergence of the algorithm is $O(N^2)$ [?]. Being N the number of nodes in the network.

The authors still tested AGM algorithm against 3 other well-known methods for community detection: Link Clustering, Clique Percolation and Mixed-Membership Stochastic Block Model. The results are presented in the *Algorithm Benchmark* section.

Source: https://cs.stanford.edu/people/jure/pubs/agmfit-icdm12.pdf

Algorithm Benchmark

In this section, you will find tools to test the accuracy and to measure the speed of any community finding algorithm.

Synthetic Networks

Although a network structure can be completely defined by its wiring diagram, different algorithms (based in distinct assumptions) will partition the same graph in different ways. This way, it is important to understand what algorithms are the most reliable.

An accurate way to provide this insight is by partitioning networks whose community structure is already known. Girvan-Newman and Lancichinetti-Fortunato-Radicchi (LFR) benchmarking techniques gave an important contribution in this area.

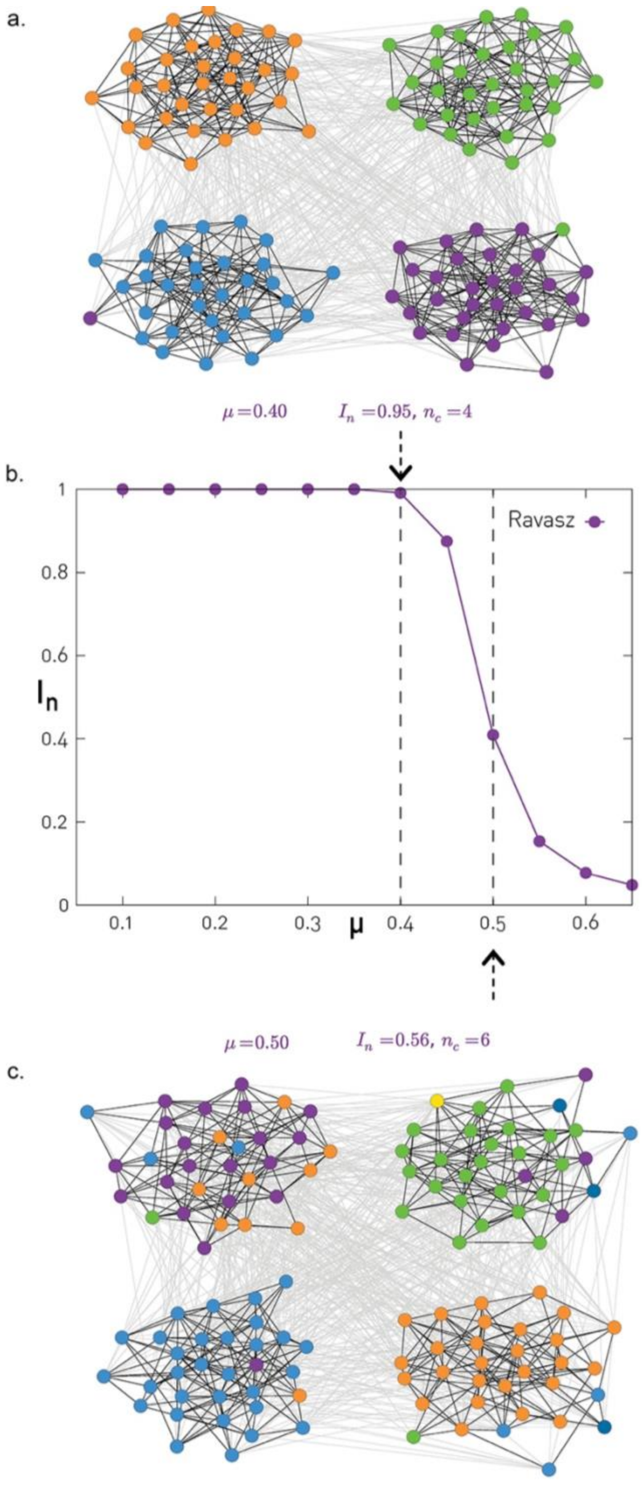
1. Girvan-Newman Benchmark

The first technique consists in generating a network with 128 nodes and subdivide it in 4 communities with the same number of elements (32 nodes). Next, each node is connected to the others Nc-1 (from the same community) with probability p_{int} and to the 3Nc external ones with probability p_{ext}.

A control parameter that captures the relation between community density and the one from the rest of the network is derived:

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

This allow us to trace a relation between community detection algorithm assertiveness and the control parameter (that reflects how differentiated are the communities inside the network). On Fig. _ is shown a representation of the performance of Ravasz algorithm for differently pseudo-randomly wired networks.



Although the previous analysis was done to a non-overlapping communities network, it can be extended to node sharing communities

2. Lancichinetti-Fortunato-Radicchi (LFR) Benchmark

LFR approach constructs a testing network which is closer to a real one. Here, the size of the communities and nodes' degree follow a power law distribution described, respectively, by coefficients ζ and γ:

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

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

The technique starts by creating N isolated nodes. Then, each one is assigned to a different community such that following the previous size distribution. Based on the node degree distribution, a degree k is attributed to each node. After that, each one will have assigned a $(1-\mu)k_i$ internal degree and the remaining of the connections will link nodes from outside its cluster. Finally, all possible links (based on nodes' internal and external degrees) between all single elements of the network are established in a random way.

3. Community-Affiliation Graph Model

AGM besides having a community finding algorithm associates created by the same authors, it can also provide us a way to build our own benchmark network. What distinguishes method from others is the capability of generating such graph accounting for community overlap.

This method was already explained before, consequently, I will avoid a detailed description. In this case, the user should decide the number of nodes and communities (s)he desires. And to attribute some probabilities to edge formation among pairs of nodes sharing the same community. In order to create even a more realistic network, one may also consider an extra community designated ε-community which pretend to add an extra probability for nodes from different communities to connect:

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

In spite of not being the best criteria in some cases, generally, it works well.

After introducing 3 methods for creating synthetic benchmarking networks, measures to assess community-detection algorithm's correctness are introduced.

<u>Accuracy</u>

Although we have already defined 2 benchmarking networks, it lacks a definition of a function able to compare the random distribution with the one present in our real graph. The function below is the key element that was missing:

$$I_n = \frac{\sum_{C_1,C_2} p(C_1,C_2)\log_2 \frac{p(C_1,C_2)}{p(C_1)p(C_2)}}{\frac{1}{2}H(\{p(C_1)\}) + \frac{1}{2}H(\{p(C_2)\})}$$

[Average F1 Score]

We define F1 score as the average of the F1-Score of the best matching ground-truth community for each detected one, and the F1-score of the best estimated again each ground-truth community. This can be mathematically defined as:

$$F1 = \frac{1}{2}(\bar{F}_g + \bar{F}_d)$$

Where:

$$\bar{F}_g = \frac{1}{|C^*|}\sum_i max_j F1(C_i, \hat{C}_j)$$

$$\bar{F}_d = \frac{1}{|\hat{C}|}\sum_j max_j F1(C_j, \hat{C}_i)$$

And $F1(C_i, \hat{C}_j)$ is the harmonic mean of precision of C_i and $\hat{C}_j$.

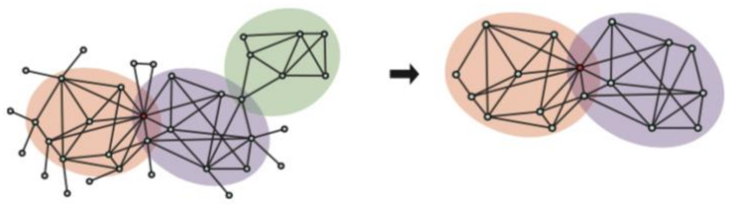
[Omega Index]

It estimates the accuracy of the detected number of shared communities by a pair of nodes versus its theoretical value.

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

[Normalized Mutual Information]

This method adopts a strategy used in information theory to compare ground-truth communities with the ones detected. Refer to [??] for more details.



[Accuracy in the Number of Communities]

This relative accuracy measure determines how the number of detected communities relates to the theoretical value.

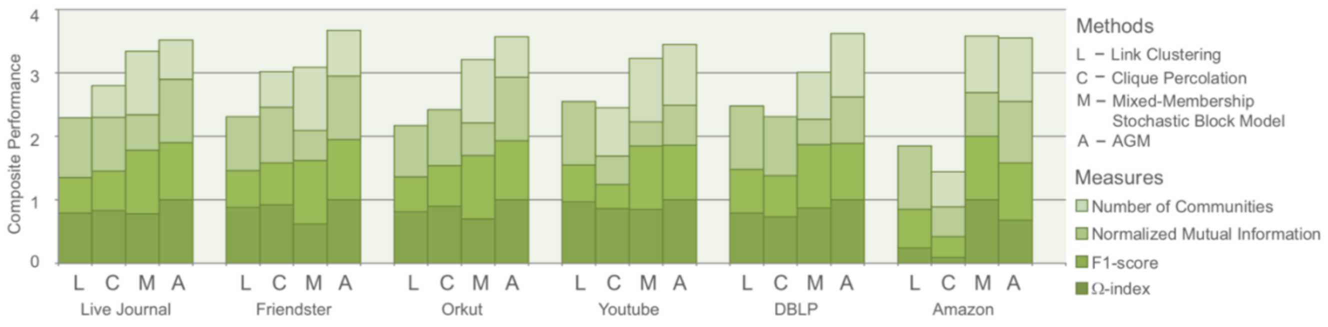
$$1 - \frac{||C^*| - |\hat{C}||}{|C^*|}$$

[Example]

Based on the 4 previous quality estimates, it was possible to represent the accuracy of 3 previously presented algorithms in 6 different ground-truth networks obtained online. Specifically, Link Clustering, Clique Percolation, Mixed-Membership Stochastic Block Model and AGM were tested, using Live Journal, Friendster, Orkut, Youtube, DBLP and Amazon. Although, these networks have millions of nodes and these algorithms' performance is not adjusted for such high number of elements. So, the strategy was to obtain a very large number of subnetworks with overlapping structure. To obtain 1 such network, one should pick randomly a node shared by 2 communities and expand the subgraph such as including connected nodes around sharing at least one of those. This process is shown in Fig. _. At the end, 500 subnetworks were obtained for each of the 6 datasets.

Examining Fig. _, we quickly verify most of the times AGM presents enhanced accuracy as predicted before.

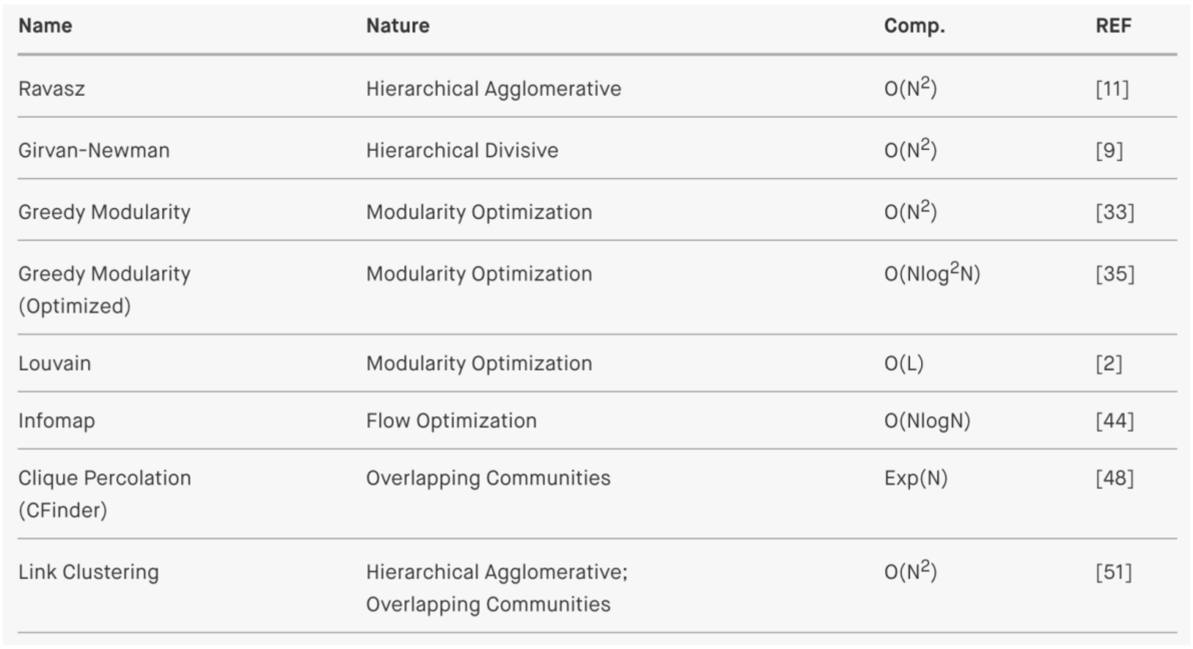
Source: http://networksciencebook.com/chapter/9#hierarchical, A. Lancichinetti and S. Fortunato. Community detection algorithms: A comparative analysis. *Phys. Rev. E*, 2009.



Speed

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

Table below compares the complete set of community finding algorithms that were introduced until this point. Note that considering sparse graphs, the following relation is acceptable: $N \sim L$. Analyzing Table _, it should become clear the fastest algorithms are Louvain and Infomap. On the opposite side, CFinder is the least efficient.



Name	Nature	Comp.	REF
Ravasz	Hierarchical Agglomerative	$O(N^2)$	[11]
Girvan-Newman	Hierarchical Divisive	$O(N^2)$	[9]
Greedy Modularity	Modularity Optimization	$O(N^2)$	[33]
Greedy Modularity (Optimized)	Modularity Optimization	$O(N\log^2 N)$	[35]
Louvain	Modularity Optimization	$O(L)$	[2]
Infomap	Flow Optimization	$O(N\log N)$	[44]
Clique Percolation (CFinder)	Overlapping Communities	Exp(N)	[48]
Link Clustering	Hierarchical Agglomerative; Overlapping Communities	$O(N^2)$	[51]

Community Characterization

As we have already seen, most real networks follow power law distributions regarding their node degrees. The same is verified in community size. A long fat tail of big communities is often present in real graphs.

All previous algorithms were applied to networks without weighted links. Although, when present, they can be exploited to graph partitioning in communities. Depending on the type of network being study strong links may be in the center or connecting different communities. When these weights are evolutionary distances, one may partition the network in such way that in the middle of each community it is likely to reside a stronger link (short distance).

Every network representing real phenomena is constantly evolving. Consequently, their wiring diagram and community structure are as well. Depending on the stability of each community, they may tend to grow, contract, merge, split or disappear. The formation of new ones is also a possibility to consider.

At the end, if we can separate the network in communities and further track their evolution over time, this can strongly contribute for the understanding of many biological phenomena. In this context, the most relevant is the microbiological (namely, in outbreak containment).

Below is the example of 2 MST greedy algorithms.

Source: https://en.wikipedia.org/wiki/Community_structure, http://networksciencebook.com/chapter/9#hierarchical

Prim's Algorithm

The way of functioning of this algorithm may be described in 3 steps:

- → Initialize a tree T with a random vertex from the initial graph;
- → Grow the tree by connecting the previous vertex to another linked by an edge with the lower weight;
- → Previous step is repeated until all vertices are present in T.

In case the initial graph is represented by an adjacency matrix, time complexity of Prim's algorithm will be $O(|V^2|)$.

Source: https://en.wikipedia.org/wiki/Prim%27s_algorithm

Kruskal's Algorithm

One major difference between Kruskal's and Prim's algorithm is that the former allows us to handle a disconnected graph. Based on the number of disconnected components, it will find a minimum spanning tree for each. Consequently, resulting a forest with the sum of each edge weight minimized.

In a concise way, this is how the algorithm works:

- → Create a forest F where each node is considered an isolated tree;
- → Create a list S with all possible edges in the network and with the respective weights associated;
- → While S is not empty:
    - Remove the edge with minimum weight from S;
    - Add it to forest F if it connects 2 different trees.

In terms of its time complexity, these two algorithms are equally fast for sparse graphs, but slower than other more sophisticated ones.

Source: https://en.wikipedia.org/wiki/Kruskal%27s_algorithm

Phylogenetics

The term phylogeny was used for the first time by Haeckel in 1866. Since the beginning, the goal of phylogenetics was to establish an evolutionary relationship between taxa. Calling it taxa, we generalize the previous claim to all kind of taxonomic groups, such as: species and populations. Based on visual representations of descendance, named phylogenetic trees, Charles Darwin was the first one to popularize these structures.

At the time, in 19th century, when they started to be used, the relations among taxa were established based in morphological characteristics. Nowadays, the attempt to classify organisms that way is called phenetics. The structure of these trees was already very similar to the one we are used. More 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.

With the discovery of DNA and the advent of protein and nucleic acids analysis, a big variety of sequencing methods arose. Those lead to a boom in the quantity and quality of data available to create more precise evolutionary relations. A new field of study has emerged – Molecular Phylogenetics.

Molecular Phylogenetics

Molecular phylogenetics uses DNA, RNA or protein sequences to draw relationships among taxa. This is done based on the expected evolutionary time separating them. Some structures are better conserved than others over time, this will determine the scope of analysis of the bioinformatician.

Following what was discussed above, one may ask which phenomena may be in the origin of those evolutionary patterns. In fact, the resulting tree bifurcation is accurately explained by 2 well-documented events:

Founder effect: the migration of organisms to new and isolated regions;

Vicariance: abrupt events may be in the origin of the separation of organism belonging to the same region. Leading, just like the above, to the isolation of lineages of species, resulting in speciation.

At DNA level, taxa divergence is explained by 2 phenomena: mutation and recombination events.

Many algorithms were developed over time to represent and analyze increasingly amounts of evolutionary data. In spite of the previous reference to a standard type of phylogenetic tree – with leaves, nodes and a root – there are a couple of alternative representations. These arose along with the algorithmic development which tries to reconstruct the most probable evolutionary history.

Data Acquisition

Data acquisition for phylogenetic analysis strongly relies in DNA sequencing. Frederick Sanger, a pioneer in protein and DNA sequencing (2 Nobel prizes attributed due to each achievement), developed a sequence method that due to its reliability and comparative ease with the methods at the time, was used in the 1st generation of DNA sequencers. This would prevail from the 1980s until middle 2000.

With the development of shot-gun sequencing and similar methods, the development of Next Generation Sequencing methods was triggered. This would lead to a row of high-throughput methods who strongly decreased the costs and time of sequencing. This was the consequence of the possibility of analyzing longer pieces of DNA, but also multiple ones at the same time. Chromosome and genome sequencing were the ultimate products of this revolution.

All these developments were proved to have a crucial role in metagenomics, medicine, forensics and, mainly, in molecular and evolutionary biology.

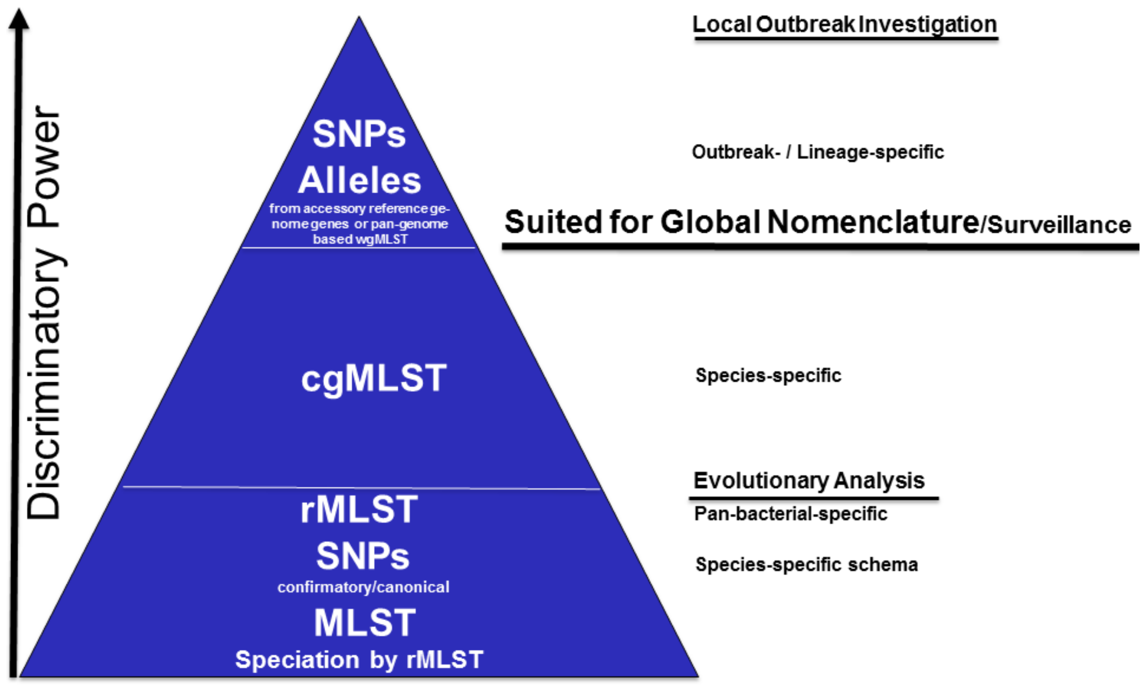
Infection since always have been a concern to human health. Plus, disease causing microorganisms such as bacteria and virus have shortest genomes. This lead the way for them being the first sequenced organisms. By the time, microbiology would see its role emphasized and the importance of the development of it would give rise to new labs and genetic databases.

There are now, a couple of standard and replicable typing methods used worldwide to infer phylogenetic relations. The most traditional is Multi Locus Sequence Typing. Instead of using the complete DNA sequence of each strain, usually 7 housekeeping loci are determined. This DNA sequences are usually strands of 400-500 nucleotides. They may be part of a gene, a regulating or a non-coding region. Progressively, more housekeeping loci are being used. The main advantage of using MLST is that it allows to buffer the effect of recombination in taxa evolution. Inference algorithms related to this method don't analyze the number of differences between segments of different strains, but only return a binary output of equal or different. Other similar techniques are cgMLST, rMLST and wgMLST. SNP is another technique which is becoming more and more used with the widespread of new high-throughput sequencing methods.

Depending on the scope of the analysis and the availability of data, one of those will be picked. Figure _ highlights how discriminatory power varies with the use of different typing methods. For short-term outbreak situations, it is wise to opt by a SNP analysis.

In spite of all the data we have now and the powerful sequencing techniques giving available, there is a lack of software to analyze many of this data.

Source: https://en.wikipedia.org/wiki/DNA_sequencing, https://www.ridom.de/seqsphere/cgmlst/, https://academic.oup.com/nar/article/44/W1/W246/2499344,



Algorithms for Tree Construction

Based on a distance matrix, one can use hierarchical clustering methods, Neighboor-Joining or Fitch-Margoliash methods to categorize data. The former includes: UPGMA, WPGMA, Single Linkage and Complete Linkage algorithms.

Moreover, Maximum Parsimony, eBurst, goeBurst, Maximum Likelihood and Bayesian Inference are some of the other popular methods among phylogenetics.

In the next sections I will go through each one, explaining the algorithm and the respective pros and cons accordingly to the data we have to analyze.

<u>UPGMA & WPGMA</u>

Single & Complete Linkage

Maximum Parsimony

eBurst & goeBurst

Maximum Likelihood

Bayesian Inference

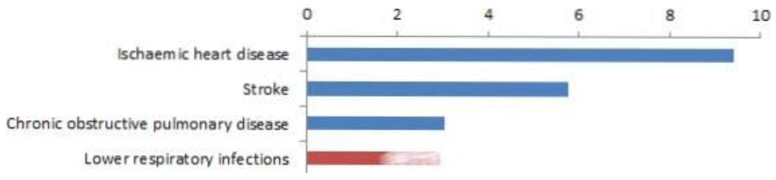
Source: https://en.wikipedia.org/wiki/Phylogenetics; https://en.wikipedia.org/wiki/Phenetics;

Bio-BC slides;

In order to represent these structures, one can use the Minimum Spanning Tree approach. There are 3 algorithms allows to

Medical Relevance

We may trace the source of an outbreak, infer about the pathogenicity/virulence of a recently discovered strain or, based on serotype data or other phenotypic data, to determine the most suitable strains to be targeted for vaccination.



PHYLOViZ

With the advent of high-throughput sequencing methods, new ways for visualizing and analysing increasingly amounts of data are needed. Although some software already exist, they do not scale well for big data or require advanced technical knowledge to be useful in phylogenetics. PHYLOViZ, along with the tools under development, will provide a brand-new insight into the field of microbiology.

PHYLOViZ is available both as a cross-platform JAVA software and online. The latter offers a RESTful API, allowing for programmatic access to data and analysis, from any Internet enable computer. 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 associated auxiliary data. This thesis will focus on the online version.

Source: https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-10-152#MOESM1,

Input Data Types

Due to the number of different databases and sequence data formats available, it was fundamental to allow the input of as many as possible. PHYLOViZ supports profile data:

- → In a tab-delimited file format, thus allowing for traditional multilocus sequence typing (MLST), cgMLST, wgMLST, MLVA (multilocus variable-number tandem repeat analysis) and single nucleotide polymorphism (SNP) analyses;
- → FASTA files with sequences of the same length or aligned to the same length;
- → And Newick format files, already containing a tree topology and branch lengths.

Along with this data formats, the user may also upload auxiliary data in tab-delimited format, such as epidemiological, demographic or temporal. Some examples of those are antibiotic resistance, serotype… In the case of tab-delimited files, identical column headers in profile data and auxiliary files will relate both. And for FASTA and Newick formats, identifiers from both files will be searched in the 1st column of the auxiliary data.

Implementation

PHYLOViZ Online is a Node.js application widely supported by most browsers. It uses goeBURST to trace Minimum Spanning Trees for different data type inputs.

Along with this user-friendly application, a RESTful API was also created. It allows to retrieve, upload data and run available tree algorithms on it.

Data visualization

Regarding data visualization, the user has 4 options. Tree visualization, table visualization, interactive distance matrix and sequence viewer. The first mode is provided by VivaGraphJS JavaScript library. It allows visualization of thousands of nodes by taking advantage of the machine's GPU hardware acceleration. Rendering is in this case assigned to WebGL JavaScript API. DataTables library is responsible for matrix and pie charts by Data Driven Documents (D3.js) and BioJS MSA Viewer assures the representation of multi sequence alignment for the FASTA input.

Source: https://academic.oup.com/nar/article/44/W1/W246/2499344,

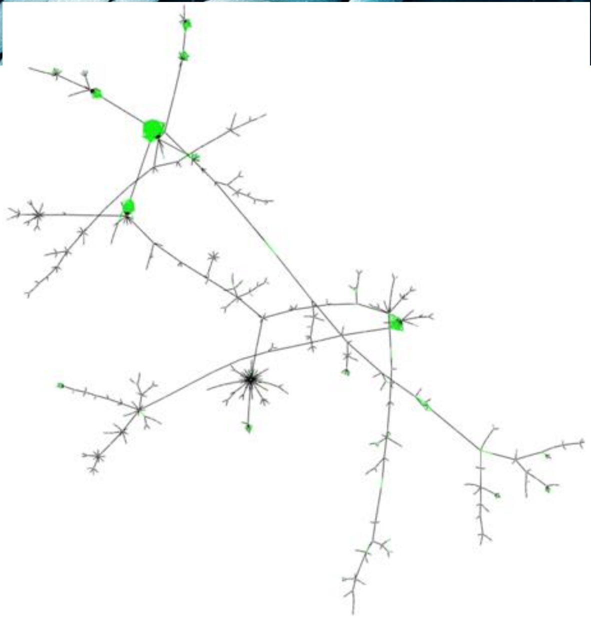
Cronobacter Spp.

In order to test some of the functionalities of PHYLOViZ Online, a case study regarding *Cronobacter spp.* is presented. This bacterium was chosen based on its clinical importance and because the corresponding database, in PubMLST, was recently updated (2018-09-29). This resulted in an analysis of an increased number of STs.

Cronobacter is a gram-negative bacterium which is commonly found in dry environments. Powdered infant formula, skimmed milk powder, teas, starches and dry fruits are a few examples of environments where this microorganism live.

Mostly, new-born/infants, elders and people in immunosuppressed conditions (e.g. Cancer or HIV patients) are at greater risk of contracting this infection.

This infection is usually constrained to a certain body system such as the urinary or to the meninges. Although, It may also spread to the bloodstream.



Despite this is not a very common pathology (in the USA), it has a high-mortality rate. For instance, when the meninges are infected by Cronobacter, almost half of the patients are expected not to survive.

Source: https://www.cdc.gov/cronobacter/technical.html

After uploading to PHYLOViZ the 672 genetic profiles present in the database, a phylogenetic tree was created.

Upon adjustment of several displaying parameters – node size, edge length, drag coefficient, spring coefficient, gravity, theta and mass ratio - one could significantly change its appearance (although maintaining the same linking pattern). Along with the tree, we may opt to display strain ST (adjustable font size).

Moreover, 2 other operations may be activated such that the user can limit the representation of the edges until a certain NLV: tree cut-off. Or to further connect all nodes that present an NLV until a pre-determined value: NLV graph.

Auxiliary data can also be presented along the tree in the form of colored nodes. This was not done due to the lack of such information to this organism.

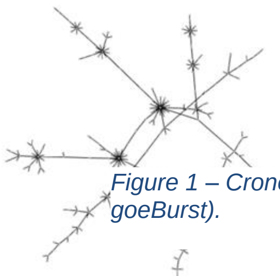


Figure 1 – Cronobacter hylogentic tree (using goeBurst).

Figure 2 - Cronobacter phylogenetic tree: NLV graph = 1.

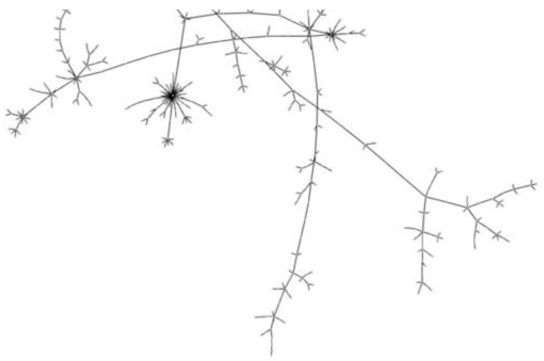


Figure 3 - Cronobacter phylogenetic tree: tree cut-off = 3. Clonal complexes were now set visible.

1.1. Context

Antimicrobial resistance is not a recent issue. In fact, the problem remotes the time drugs started to be used to treat a broad range of diseases caused by pathogenic organisms (e.g., viruses, bacteria, fungi and other parasites). Although this is a natural evolutionary process, in which the most competent strains get selected over time, the frequency people are using antimicrobial drugs is accelerating the resistance. This is a particularly relevant issue not only among people (who are being mistreated and disease prevented using these drugs inappropriately), but also in farms where cattle are being given preventive doses of antibiotics. Moreover, the frequency people are travelling around the world is facilitating the spread of contagious diseases. All of this makes surveillance and, consequently, quick outbreak tackling, a progressively fundamental issue.

1.2. Problem definition

[Relevance]

1.3. Solution objectives

Text

1.4. Research methodology

Text

1.5. Document structure

In order to guide the reader, after introduction this document will firstly explore related work that was already developed in the area. Then, all the results will be successively presented starting by the algorithms that were developed to perform community finding in general networks (with predefined input format); the benchmark techniques that were used to assess algorithms' accuracy and speed. And, finally, the different approaches used to display the results to the user.

2. Related work

Regarding the community finding algorithms and accordingly to my research online, Infomap and Layered Label Propagation algorithms were implemented for the first time in JavaScript. There was already available, on the web, a version of Louvain. In spite of the one present in the thesis being based on this, it was modified accordingly to the goals of the project.

Relatively to the benchmark analysis, two algorithms in JavaScript were implemented, for the first time, to generate both Girvan-Newman and Lancichinetti-Fortunato-Radicchi benchmark networks. Apart from the analysis in synthetic networks, real ones (spanning different network properties) were also considered. Those essentially belong to the social and web domains. One of those was Zachary's karate club network, to which a JSON file was created for the first time.

After testing the algorithms, two interfaces were used to visualize the results presented next. With this in mind, the most recent versions available (at the time) of D3.js and Cytoscape.js were used. Again, the representation of data was highly personalized, based on the tools these frameworks allow the user to explore.

3. Results

Wrapping up what was previously discussed, Louvain, Infomap and Layered Label Propagation were implemented with the intention of performing community finding in significantly large networks. With these in mind, some benchmark techniques were carefully used, and the results displayed with scalable online frameworks.

A comparison of the speed and accuracy of the 3 algorithms was realized using real and synthetic networks. After, Moreover, the performance of the different D3.js using SVG and Canvas, and Cytoscape was also considered. Both the assessments were done with real and synthetic networks.

3.1. Algorithms

3.1.1. Louvain Algorithm

Considering complexity of the greedy algorithm (O(N^2)), we fast understand it does not scales well for big networks. This led to the arousal of improved methods of graph partitioning.

Louvain algorithm not only assures the same accuracy as the ones presented before, but also it is suitable to be used in graphs with a high number of edges.

After assigning, at the beginning, a different community to every node in the network, the algorithm runs the following way:

1. The algorithm should compare every node with every existing community and the modularity variation of the network calculated using the formula below:

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

   Σ_{in} is the sum of the weights of the links inside community j; $k_{i,in}$ is the sum of the weights of the links connecting node i to community j nodes; W is the total weight of all links from the graph and Σ_{tot} sum of the weights from all links associated to nodes in C.

   Whenever modularity variation is positive and maximized, node i is joined to community C;

2. After associating all nodes to a community, each community is then represented by a single node. In such way the final network ends up with as many links as communities

in the older. Moreover, the updated network will contain nodes with self-loops and links to other nodes. The weight of the first ones will be the sum of the weights of the links inside the correspondent former cluster. And the second will represent the sum of the weights of the links connecting nodes from the 2 different communities before.

These 2 last steps should be performed until network modularity is maximized. Each iteration containing both is called a path.

Regarding the computational complexity, its complexity is limited not for their computational time, but storage demands. Predictably, the most demanding step is the 1st one which scales with L. At the end, the algorithm time complexity is O(N), allowing analysis in networks with millions of nodes.

Algorithm:

3.1.2. Infomap Algorithm

Similarly to Louvain Algorithm, network partitioning is done using a quality function. In this case, we pretend to minimize *map equation*:

$$L = qH(Q) + \sum_{c=1}^{n_c} p_{\odot}^{c} H(P_c)$$

Algorithm:

3.1.3. Layered Label Propagation Algorithm

Layered Label Propagation algorithm development was strongly based in the olded Label Propagation. The latter starts by assigning a different community to each node present in the network and, then, iteratively, it modifies the belonging community based on the communities of the nodes in the neighborhood. This means, to each node will be assigned the community the most represented by its immediate neighbors. In each iterate, this step will be executed for each node, in a random order. In other words, at the end of each iterate, the sequence at which will be picked is randomized. At the end, it is expected that the regions of the network strongly connected will preferentially be grouped in a single community.

Label Propagation Algorithm:

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 pretends to optimize. In this case, community will be the one that maximizes:

This approach pretends to buffer the counting of the nodes belonging to a certain community (in the neighborhood), by its usual presence across the whole network.

Layered Label Propagation Algorithm:

3.2. Benchmark Networks

3.2.1. Synthetic Networks

Two types of network generating algorithms were implemented to generate both the **GN** and **LFR**.

The first one was historically the first one to be appear as a result of an effort of Girvan and Newman. In their paper, where it was described for the first time, they considered a network of 128 nodes divided in 4 different groups, each one with exactly the same number of nodes (32). Then, a mixing parameter chose by the user would iteratively be used to connect different pairs of nodes with different probabilities. Precisely, nodes from different groups would connect with a probability given by that value (u) and the ones present in different communities with 1-u. Nevertheless, the authors consider that each node will exactly be connected to 16 different ones. In spite of this, the algorithms I implemented allow not only the mixing parameter to vary, but also the average degree. This was useful because allowed an increased number of benchmark tests using these networks.

GN Algorithm:

On the other hand, Lancichinetti-Fortunato-Radicchi benchmark networks, try to better resemble real networks. This means that not only they consider a power-law distribution of community sizes, but also a power-law distribution of node degrees. This way, the user should input the corresponding exponents for each distribution. Due to this characteristic, the previous implementation needed to be adapted to conform with these requirements.

LFR Algorithm:

3.2.2. Real Networks

3.3. Interface

3.4. Benchmarking

Text

4. Conclusion

Text

5. References

1. Albert-László Barabási Books. Network Science. http://networksciencebook.com. Acessed in October 2018;
2. WHO. The top 10 causes of death. http://www.who.int/news-room/fact-sheets/detail/the-top-10-causes-of-death. Acessed in October 2018;
3. Bruno Ribeiro-Gonçalves, Alexandre P. Francisco, Cátia Vaz, Mário Ramirez, João André Carriço; PHYLOViZ Online: web-based tool for visualization, phylogenetic inference, analysis and sharing of minimum spanning trees, *Nucleic Acids Research*, Volume 44, Issue W1, 8 July 2016, Pages W246–W251, https://doi.org/10.1093/nar/gkw359;
4. Wikipedia. *Streptococcus pneumoniae*. https://en.wikipedia.org/wiki/Streptococcus_pneumoniae. Acessed in October 2018;
5. CDC. Pneumococcal Disease. https://www.cdc.gov/pneumococcal. Acessed in October 2018.