José Antônio Pellizzaro

An algorithm for network community structure detection by Surprise

Brasil

José Antônio Pellizzaro

An algorithm for network community structure detection by Surprise

Dissertação submetida ao Programa de Pós-Graduação em Física do instituto de Física da UFRGS, como quesito parcial para obtenção do título de Mestre em Física Teórica.

Universidade Federal do Rio Grande do Sul – UFRGS

Instituto de Física

Programa de Pós-Graduação

Orientador: Daniel Gamermann

Brasil

Agradecimentos

Esse trabalho também é fruto de uma rede. Embora muito se fale do poder das conexões fracas, um trabalho tão grande como esse não é um produto delas, mas sim, é o resultado de conexões fortes e duradouras.

Com isso em mente, agradeço aos meus pais, Neuri e Marli, pelo incansável suporte durante os meus 26 anos. Pelo seu incentivo, principalmente nas horas mais difíceis, sou eternamente grato.

Agradeço ao meu orientador, Daniel Gamermann, por sua entrega e companheirismo durante esse projeto. Pelas dicussões e pela compreensão quando tive problemas pessoais.

Aos meus amigos agradeço por decidirem compartilhar suas vidas comigo.

A CAPES, agradeço pelo financiamento. Minha formação não teria sido possível sem a existência de instituições públicas de qualidade. Devemos defendê-las nesses tempos difíceis.

"I was born not knowing,
and have only had a little time
to change that here and there."
Richard P. Feynman

Resumo

O sucesso da teoria dos grafos para descrever sistemas complexos, bem como a onipresença destes, deu muito destaque a elaboração de métodos eficientes para sua analise. No entanto, varias questões continuam em aberto. Uma delas, a qual nos dedicamos neste trabalho, é a obtenção das comunidades presentes nessas redes. Muito embora não exista um consenso formal sobre sua definição, a presença de comunidades vem da ideia intuitiva de que nós formam subgrupos dentro da rede. Neste sentido, muitos algoritmos diferentes foram propostos para identificar tais grupos. Aqui nós atacamos este problema em duas frentes: primeiro, desenvolvemos um novo algoritmo baseado na função Surprise e segundo, criamos um novo benchmark, um conjunto de redes artificiais com comunidades pré-estabelecidas, para comparar a performance de diferentes algoritmos. O nosso algoritmo, chamado Surpriser, foi testado contra sete outros métodos da literatura em três benchmarks diferentes. Nós mostramos que métodos baseados na Surprise são os mais consistentes nos diferentes benchmarks e que o nosso Surpriser leva uma vantagem sobre os últimos. Finalmente, mostramos que o nosso benchmark é o mais difícil dos três, pois poucos algoritmos conseguem resolve-lo.

Palavras-chaves: Grafos, Redes, Detecção de Comunidades, Benchmarks, Surprise.

Abstract

The success of network science to describe many complex systems and their ubiquitous presence has brought the development of new, more efficient, methods of analysis to the spotlight. However, some problems still remain open. One of which, the focus of our work, is the determination of a network's community structure. Even though there's no consensual formal definition, communities come from the intuitive idea that nodes form subgroups in the larger networks. In this regard, many different algorithms have been proposed in order to identify such groups. Here we tackle this problem in two different fronts: first, we developed a new algorithm based on the Surprise function and secondly, we created a novel benchmark, a set of artificial networks with a seeded community structure, to compare the performance of competing algorithms. Our own Surpriser algorithm was tested against seven other methods from the literature in three different benchmarks. We show that the Surprise based methods are the most consistent among different benchmarks, with Surpriser having an edge over the competition. Finally, we show that our benchmark is the hardest of the three as very few algorithms are able to solve it.

Key-words: Graphs, Networks, Community Detection, Benchmarks, Surprise

List of Figures

List of Tables

List of abbreviations and acronyms

CPM	Constant Potts Model Algorithm
LFR	Lancichinetti-Fortunato-Radicchi Benchmark
NMI	Normalized Mutual Information
PI	Pielou's Index
RB	Reichardt and Bornholdt Algorithm
RC	Relaxed Caveman Benchmark
RN	Ronhovde and Nussinov Algorithm
SA	Simulated Annealing
VI	Variation of Information

Contents

Introduction

The modern network science is rooted in a field of mathematics called graph theory. This discipline was started by no other than Leonard Euler in 1736 as a way to solve a rather unusual problem.

Back on his time, the port city of Konigsberg enclosed four land masses connected by seven bridges (figure 1). A popular question was if it was possible to find a route across the city that passed through all the seven bridges crossing each exactly once.

Euler, without ever visiting the city, solved the problem using only his creativity, wit and, obviously, mathematics. He realized that the geometry of the city was irrelevant to the solution, the only thing that mattered was how the land masses were connected. The significance of this idea is that, instead of analysing the map of the city with all its roads, squares and buildings, a much simpler structure could be used. Figure 2 represents this structure, which, nowadays, we call a graph.

Before we go into the specifics of Euler's solution, we will take a break to present the most basic proprieties of graphs. Graphs are made of two components: the vertices and the edges between them. In our example, each dot from A to D represents a vertex while the numbered lines, from 1 to 7, represent edges. In the real world system, the land masses form the vertices and the bridges form the edges of the graph.

Vertices A,C and D have 3 edges while B has 5 edges, this number, the quantity of

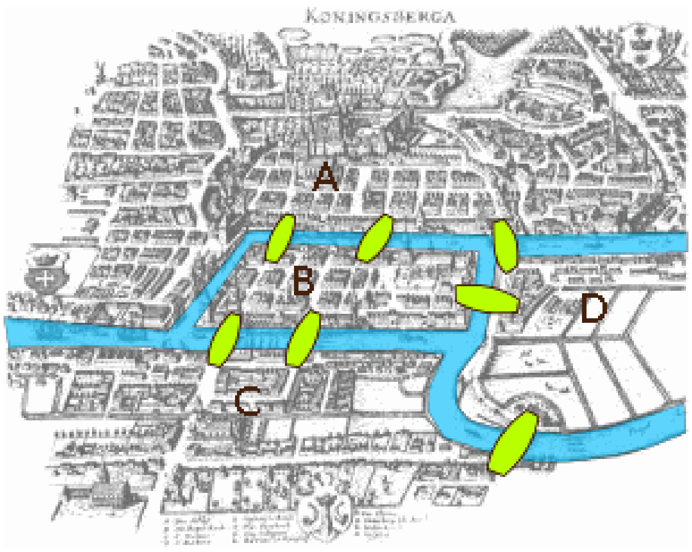


Figure 1 – The seven bridges of Konigsberg. Back on Leonard Euler's time, a common riddle was if it was possible to find a path through the city that contained all the bridges and crossed each bridge exactly once. This problem inspired Euler to start what we call graph theory today.

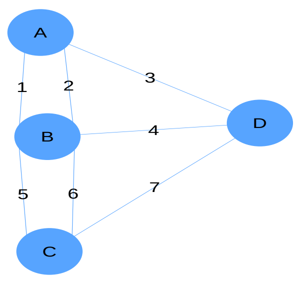


Figure 2 – Graph symbolizing the bridges of Konigsberg. Each vertex from A to D represents a land mass in the city (as per figure 1) and each edge represents a bridge (numbers one to seven).

edges a vertex has in the graph, is called the degree k of the vertex. This propriety will be the key to solve the problem of the bridges of Konisgberg.

In a formal way, the problem is equivalent to finding a path that contains every edge in the graph and that crosses each edge exactly once. As a vertex degree must always be an integer, there's no such thing as a half edge (as there's no half bridges, either), the idea behind solving this problem is analysing what happens to the path when we add a vertex whose degree is even and what happens when the degree is odd.

Euler realized that such path could have any number of vertices with an even degree. Let's think about it for a moment, when the number of bridges is even, we can always enter the land mass by a bridge and leave from another one. This means that we would never cross the same bridge twice, as intended.

When the number is odd however, things get a little more complicated. For example, in vertex A, we can enter by bridge 1 and leave by bridge 3 without a problem, but the next time we enter A we can never leave - because we would be crossing the same bridge twice. Based on this, we conclude that a graph can have at most two vertices with an odd degree, the starting point of the path and the ending point.

Recall that all the vertices in figure 2 have an odd degree, hence Euler proved that such a path does not exist. This example is interesting because the main idea behind network science is to study real-world structures. The concepts of network science are so simple and powerful that we can use them to model systems not only from math, but from physics, biology, social and computer sciences as well.

The usual nomenclature in network science is slightly different, instead of using the word vertices, we will use nodes and instead of edges it's common to use links or connections [1]. To give a bit more color to the wide range of systems described as networks, we have figures 3 a) to c). In figure 3 a) we have the network of airports, as taken from the Open Flights website, this network has 3425 nodes and 18797 links (a more complete visualization is available at (GUIMERA et al., 2005). Every node is an airport. We connect two airports, A and B, if there's a flight that starts in airport A and ends in airport B. The edges here have a direction, connecting A to B means that a flight leaves from A and arrives at B, while connecting B to A implies that a flight starts in B and ends in A. If A is connected to B and B is connected to A we say that the edge is bidirectional. Also, the node sizes in the image are scaled to their degree, meaning higher degree the larger the node in the figure. Observe that very few nodes have large degree, which is a common propriety of real-world networks that we will talk about later.

In figure 3-b) we have the passing network of a football team during a match. More precisely the network from Barcelona on their marvelous champions league win against Manchester United in 2011. In this network not only the edges are directed, showing the direction of the pass, but they have different 'strength', the size of the edge and its strength is proportional to the number of passes between two players in the match. Formally we say that this network is both directed and weighted (the strength of the edges). Obviously, the network can have from 11 to 14 nodes (if we consider substitutions), representing all the players in the match. It's also interesting to note that in this representation the position of the nodes in the diagram is of significance, it represents the average passing position for each player (there are several works regarding this kind of networks, so for those interested one can check (GONÇALVES et al., 2017; PENA; TOUCHETTE, 2012).

Finally we have a network where the edges are undirected and unweighted, so node A being connected with node B is the same as node B being connected to node A and so all the edges have equal weight. This is the metabolic network of *Synechocystis* created using data from KEGG (KANEHISA; GOTO, 2000) and represented in Figure 3-c). This unicellular, cyanobacteria has 967 different chemical compounds (metabolites) in its metabolism – the nodes in the network. Two metabolites are connected in the network if they form a product reagent pair in a chemical reaction in the organism, *i.e.*, we connect substances that are on different 'sides' of a chemical reaction.

Those are just some examples to show the structural diversity of real-world systems that find representation as networks, their sizes can range between a dozen of nodes -as the passing networks, to thousands - as the metabolic and airport networks, to billions -as the web . Apart from those, networks are used to study works of fiction (RIBEIRO et al., 2016; CARRON; KENNA, 2012), the interaction between proteins (ALDECOA;

[1] Throughout the text vertices, nodes and edges, connections, links will be used as synonyms to keep the reading more bearable.

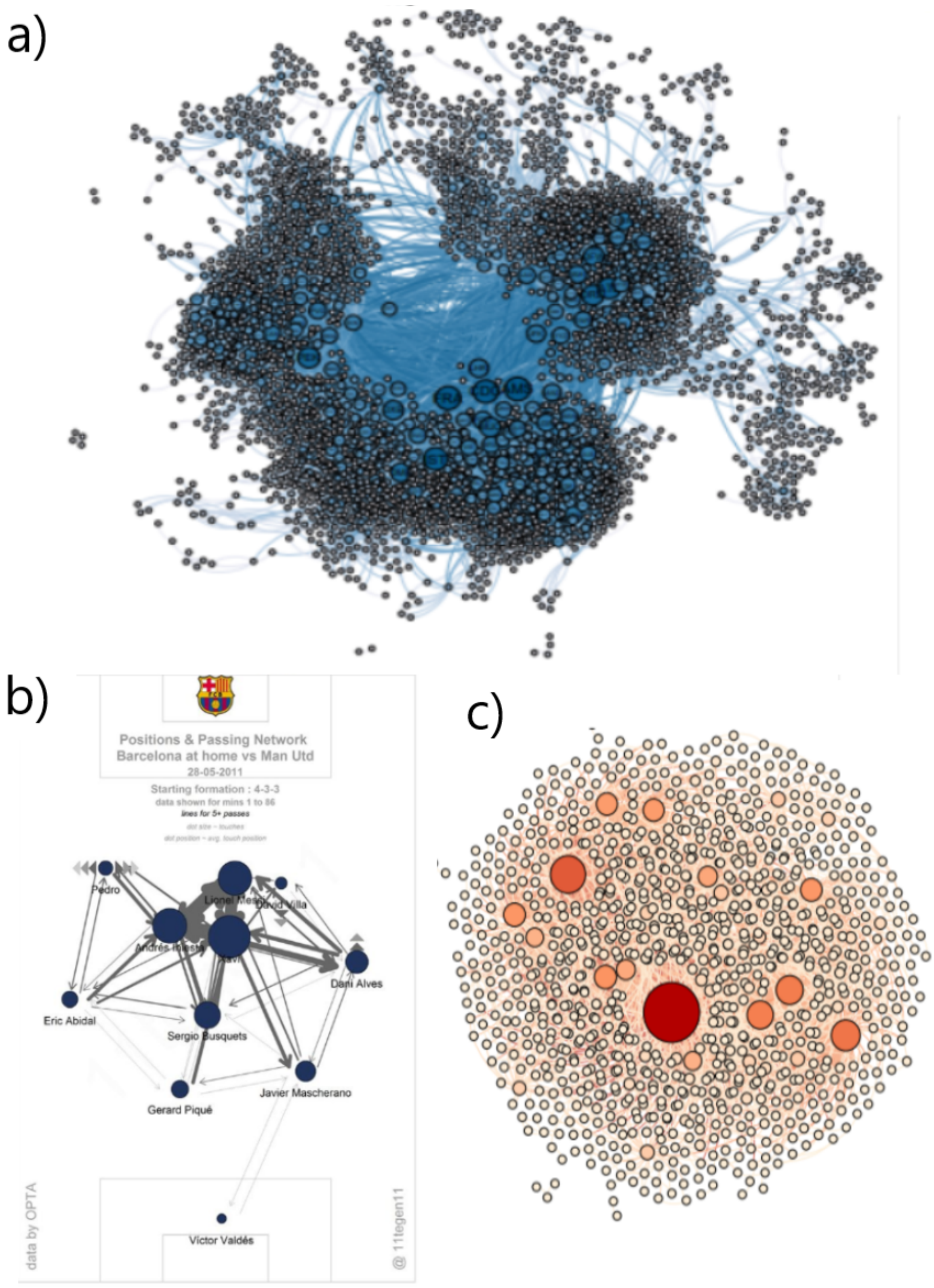


Figure 3 – Example of real-world networks in a) we have a directed network, in b) a directed and weighted network and in c) an undirected and unweighted network. a) Airport Network that has 3425 nodes and 18797 links. Node sizes as well as their colors scale with the nodes degree. Two airports, A and B, are connected if a flight starts in A and ends in B. b) Passing network from Barcelona's match against Manchester United in the champions league finals in 2011. Link represent passes between the players and its weight is given by the number of passes between such players. c) Metabolic Network from Synechocystis, each node represents a molecule. Two molecules are connected if they form a product-reagent pair in a chemical reaction.

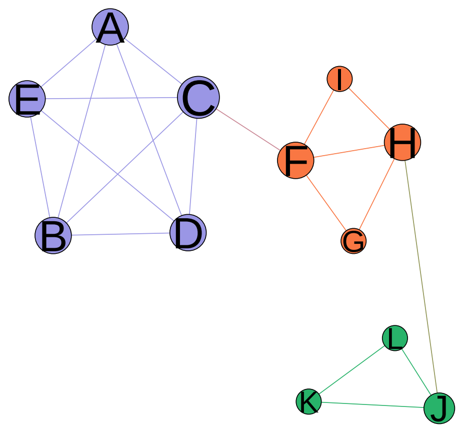


Figure 4 – Intuitive example of what we call the community structure of a network. The different communities, nodes A through E, F through I and J to L are identified by their colors.

MARÍN, 2010; RIVES; GALITSKI, 2003; ARNAU; MARS; MARÍN, 2004), the interactions between humans (RAMOS et al., 2015; JONES; HANDCOCK, 2003), how our brains are organized (NICOLINI; BIFONE, 2016; NICOLINI; BORDIER; BIFONE, 2017) and even how science works (ROSVALL; BERGSTROM, 2008; BARABÁSI; GULBAHCE; LOSCALZO, 2011; BARRAT et al., 2004). In resume we have covered that network science studies very important structures like our brains and our society to the most important from the unimportant things like football, so we have every base covered. Even with all this multiplicity, networks still share similar proprieties with one another, in special, the one that motivated this work is the presence of large-scale structures called communities (BARABÁSI; GULBAHCE; LOSCALZO, 2011; BARABASI; OLTVAI, 2004; FORTUNATO, 2010).

Communities come from the intuitive idea that nodes form subgroups in the larger network. A very basic example is in figure 4 where we have three communities in the colors blue (nodes A through E) orange (nodes F to I) and green (nodes J to L). Even though there's no consensus formal definition of what a community is, it is accepted that a community represents a group of nodes more connected between each other than with the rest of the network. A more grounded example is a group of colleagues or friends in a social network, or the different plot lines in a history.

Special interest in community detection techniques comes from the field of molecular biology, where modules relate to specific biological functions (HARTWELL et al., 1999).

Works have been done in various different fronts, in metabolic networks like example 3-c) (BARABASI; OLTVAI, 2004; PAPIN; REED; PALSSON, 2004; GUIMERA; AMARAL, 2005) , in Protein-Protein interaction networks (RIVES; GALITSKI, 2003; SPIRIN; MIRNY, 2003; ALDECOA; MARÍN, 2010), the name spoils it but, in those networks the nodes represent proteins and they are connected if they interact through some process, and in transcriptional regulation networks or gene regulatory networks (ZHU et al., 2008; SEGAL et al., 2003), in these directed networks genes symbolize the nodes and edges are placed between genes whenever the product of gene A controls the expression of gene B (JUNKER; SCHREIBER, 2011). Knowledge over these structures not only sheds light on the evolution of organisms (BARABASI; OLTVAI, 2004), but could also be used to diagnose and treat diseases (AZUAJE et al., 2013; BARABÁSI; GULBAHCE; LOSCALZO, 2011; ZANZONI; SOLER-LÓPEZ; ALOY, 2009).

The hardship to solve this problem is based on two fronts, first there's an insurmountable number of ways to divide a network of N nodes in m modules, the number of possible partitions (divisions) is proportional to the nth Bell number[2] for large N, this makes it impossible to analyze all the partitions as the Bell number grows faster than exponentially; second, there's no consensus formal definition of community, this means that different methods define communities in different ways. Those two conditions have led to the creation of a diverse number of heuristic algorithms. Here we propose a novel algorithm based on the Surprise function (ALDECOA; MARÍN, 2011) and compare it with seven other methods from the literature.

An emphasis will be given to methods that maximize the Modularity of the system. Modularity, like the Surprise, is a quality function. These functions seek to estimate how good or bad a division of the network in communities is. Every partition of the network has a given score, the value of the quality function. The higher this score is, the better is the partition theoretically. Maximization of Modularity is the most popular community detection method, as such, various works were dedicated to analyse the Modularity function. In particular, the works by (FORTUNATO; BARTHÉLEMY, 2007) and (GOOD; MONTJOYE; CLAUSET, 2010) showed that Modularity suffers from two main problems: first, it has a resolution limit, being unable to identify smaller communities depending on the size of the network, and second, the solutions are heavily degenerated, meaning that there's a large number of solutions with a high and similar Modularity score and that these solutions can be structurally very different. Some qualitative analysis of the resolution limit can be found in the papers (ALDECOA; MARÍN, 2011; NICOLINI; BORDIER; BIFONE, 2017; NICOLINI; BIFONE, 2016).

As for Surprise, the function was first introduced in (ARNAU; MARS; MARÍN, 2004) as a way to measure the quality from the partitions found by the UVCluster

[2] We give a formal expression for the Bell number in section 1.1.

algorithm. The authors further expanded this idea in (ALDECOA; MARÍN, 2011), where they propose a meta-algorithm. After applying different methods from the literature, the best partition would be the one with the highest Surprise. They show that this meta-algorithm outperforms any individual method. In the same work they make a quantitative analysis of the resolution limit where they show that Surprise is able to identify the cliques in the ring of cliques networks and in other examples where Modularity fails. Later, they introduce an approximation for the Surprise in (TRAAG; ALDECOA; DELVENNE, 2015). We disregard such approximations and aim to maximize the Surprise as defined in (ALDECOA; MARÍN, 2011) with our own greedy algorithm.

Another challenge for community detection methods is that, in general, the community structure of a real system is not known *a priori*, which limits the ability to test the algorithms in real networks. Instead, a few benchmark networks were proposed so that it's usual to compare the performance of algorithms in those 'synthetic' networks. A benchmark network is a network where the community structure is seeded upon its creation. This makes it straightforward to compare the results from different algorithms with the correct (planted) communities. In this work, we limit our analyses to benchmark networks[3]. We selected two benchmarks from the literature, Lancichinetti–Fortunato–Radicchi (LFR) (LANCICHINETTI; FORTUNATO; RADICCHI, 2008) and Relaxed Caveman Graphs (RC) (ALDECOA; MARÍN, 2011), a modification on the Caveman Graphs from (WATTS, 2004), and we also propose a new benchmark to test the community detection algorithms.

This work is divided in four chapters. In the first one, called network basics, we do a brief review of the mathematical description of networks and some of their proprieties with a focus on communities and the presentation of useful metrics in the study the community structure. Next, chapter two contains an explanation of the workings of each algorithm. In chapter three we present the benchmarks used to test those algorithms and in chapter four we present the results obtained by applying those methods on the benchmark networks.

[3] Some work has already been done with the use of Surprise in real networks, for that check (NICOLINI; BORDIER; BIFONE, 2017; NICOLINI; BIFONE, 2016).

1 Network Basics

Networks come in different shapes and sizes: they can be directed or undirected, be weighted or unweighted or combinations of both [1]. All of those, however, can be drawn as nothing but pair of dots coupled by lines. This visual representation is translated by what we call the adjacency matrix A. Each network has an adjacency matrix associated with it. The properties of those matrices however, will depend on the characteristics of the network.

In an undirected and unweighted network, an element A_{ij} from the adjacency matrix is equal to one when nodes i and j are connected and zero otherwise. Saying that a network is undirected is the same as saying that all its connections are bidirectional, this means that if i is connected with j, j is also connected with i. Next, an unweighted network is one where every single link has the same strength or weight w_{ij} (the convention is to use $w_{ij} = 1$).

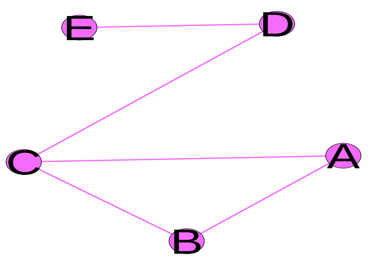


Figure 5 – Simple network with 5 nodes and 4 links. Equation 1.1 shows the associated adjacency matrix.

An example of undirected and unweighted network is figure 5 with its respective adjacency matrix represented in equation 1.1. Note that, in this case, the matrix will be symmetric and $N \times N$, where N is the number of nodes in the network.

$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}. \tag{1.1}$$

[1] There are other classifications, but to keep it short we will stick to these ones

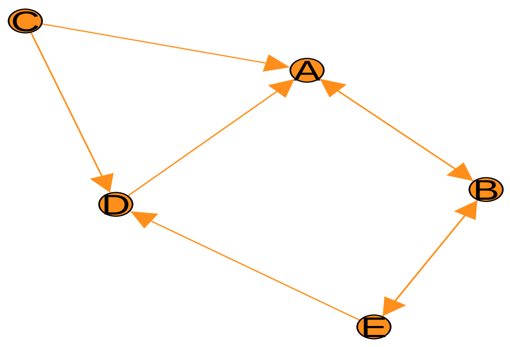


Figure 6 – Directed network with 5 nodes and 8 links. The respective adjacency matrix is in equation (1.4).

The number of links each node has, also known as the node's degree k_i, is easily obtained in terms of the adjacency matrix (the index i represents the row and the index j represents the column):

$$k_i = \sum_{j=1}^{N} A_{ij}. \tag{1.2}$$

Furthermore a sum over all the k_i is related to the total number of links L in the network by the handshake lemma (EULER, 1741):

$$L = \frac{1}{2}\sum_{i=1}^{N} k_i. \tag{1.3}$$

If a network has unidirectional connections, we say that the network is directed. The best way to explain this concept is through an example. Consider the web. Here we have two websites, i and j. If website i has a link to website j, we say that an edge comes out from node i to node j. Further, if website j does not have a link to website i, we say that the connection (the edge) between i and j is unidirectional. If website i has a link to j and website j has a link to i, we say that the connection is bidirectional (in the same vein as undirected networks).

Figure 6 shows an example of a directed network. The main difference between the undirected and directed networks adjacency matrix is that the later is not necessarily

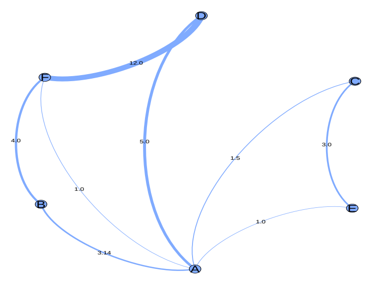


Figure 7 – Weighted and undirected network with 6 nodes and 8 links, each edge is labeled by its weight. Equation 1.7 represents its adjacency matrix.

symmetric (in most cases it isn't) as seen in equation (1.4) bellow.

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix}. \tag{1.4}$$

In these networks the nodes degrees are divided in two: k_{in}, that measures the number of links that point towards the node and k_{out} that measures the number of links that come out from the node. Explicitly,

$$k_{in} = \sum_{j=1}^{N} A_{ij}, \tag{1.5}$$

and

$$k_{out} = \sum_{i=i}^{N} A_{ij}. \tag{1.6}$$

In weighted networks each edge is given a weight w_{ij} to express its strength in the network (w_{ij} can be any real number). The adjacency matrix becomes something like:

$$A = \begin{pmatrix} 0 & 3.14 & 1.5 & 5 & 1 & 1 \\ 3.14 & 0 & 0 & 0 & 0 & 4 \\ 1.5 & 0 & 0 & 0 & 3.0 & 0 \\ 5 & 0 & 0 & 0 & 0 & 12 \\ 1 & 0 & 3.0 & 0 & 0 & 0 \\ 1 & 4 & 0 & 12 & 0 & 0 \end{pmatrix}. \tag{1.7}$$

In resume the adjacency matrix A_{ij} is as follows:

$$A_{ij} = \begin{cases} w_{ij}, \text{if } i \text{ and } j \text{ are connected,} \\ 0, \text{otherwise.} \end{cases}$$

keeping in mind that if the network is unweighted $w_{ij} = 1$.

Another common propriety in the literature is the so called scale-freeness of the networks. Many works have claimed that the web, airports, scientific collaboration network, human sexual relationships (BARABÁSI; GULBAHCE; LOSCALZO, 2011; CALDARELLI, 2007; BARABÁSI; BONABEAU, 2003; BARRAT et al., 2004) share this quality, even though there's some recent controversy to the validity of these claims (BROIDO; CLAUSET, 2019; GAMERMANN; TRIANA; JAIME, 2019). A scale-free network is one where the probability distribution of their nodes degree follows a power law:

$$\mathcal{P}(k) = Dk^{-\gamma}. \tag{1.8}$$

Here k is the degree of the node and D and γ are constants. In general, on real networks $2 < \gamma < 3$ (BARABÁSI; GULBAHCE; LOSCALZO, 2011). This means that the probability of finding a node with a low degree is much higher than the probability of finding a node with a high degree. In other words, a large portion of the network has very few neighbours while a small fraction, the so called hubs, are highly connected. This is what we saw in figure 3-a) where the hubs, the largest and dark colored nodes, are few and far between in the network.

We call this distribution scale-free because scaling the degree by a constant factor c produces a proportionate scaling on the distribution itself:

$$P(ck) = D(ck)^{-\gamma} = c^{-\gamma}P(k). \tag{1.9}$$

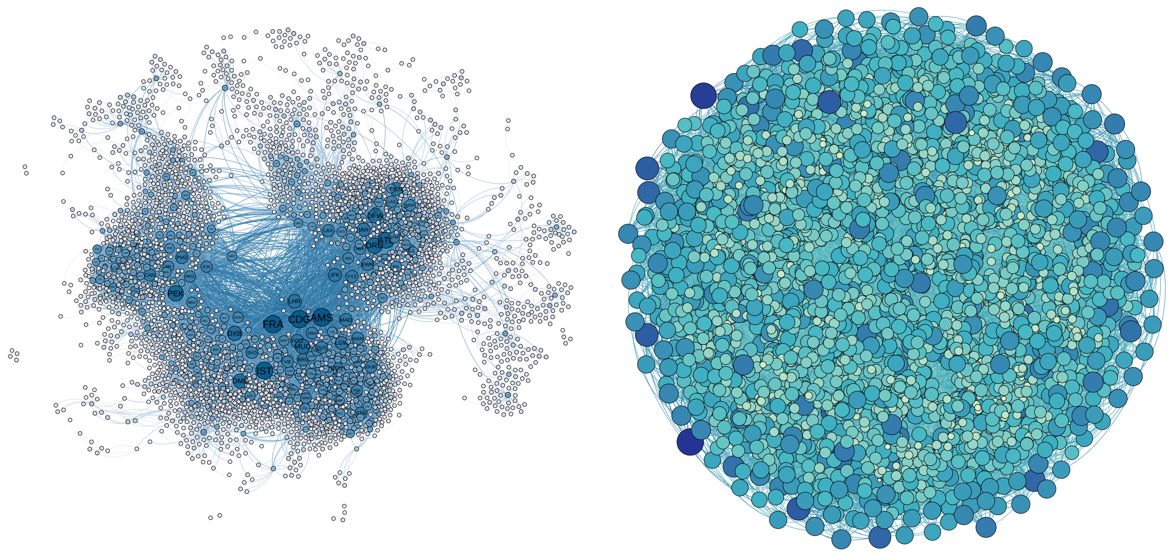
The pattern above resembles what is seen in fractals (CALDARELLI, 2007), where a small fraction of the system, when enlarged, forms a copy of the whole. Observe that when plotted in a logarithmic scale, expression (1.8) turns into the equation of a line

$$\log \mathcal{P}(k) = \log D - \gamma \log k, \tag{1.10}$$

and that plugging the scaling factor c just changes the intersection with the x and y axes

$$\log \mathcal{P}(ck) = \log D - \gamma(\log k + \log c). \tag{1.11}$$

This behavior is very different from what is expected from a random graph as studied by Erdos-Rényi. In that model edges emerge at random with a constant probability p. To directly compare the models we built a random graph with the same number of nodes as the airports network and with the appropriate value of p to assure a comparable number of links. Here p is given by $p = 2L/(N(N-1))$, where N is the number of nodes in the network (3425) and L is the number of links (18797), plugging those values gives $p = 0.00320570$. Figure 8 represents the resulting network which has 3425 nodes and 19024 links (the coloring and scaling scheme is the same as in figure 3). Clearly the distribution of nodes degrees is way more even than in the real network, where we had few hubs and lots



(a) Aiports Network.

(b) Random graph with an equivalent number of nodes and links.

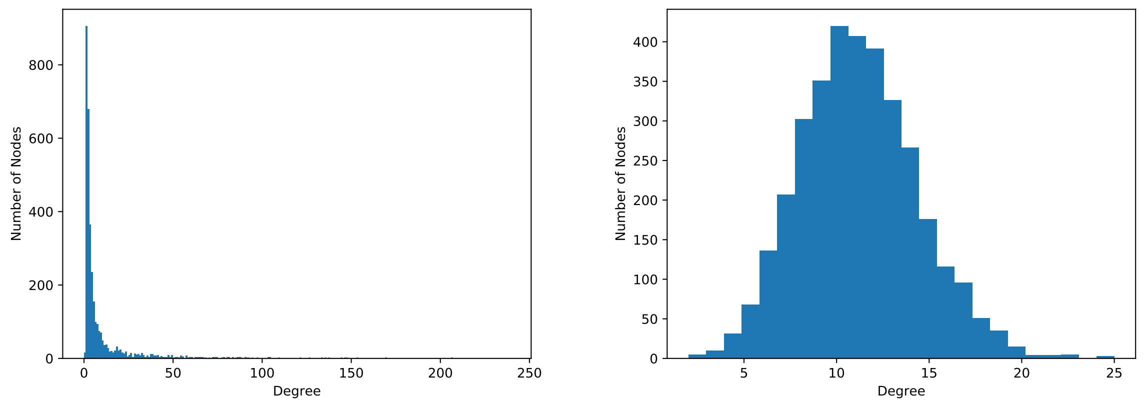
Figure 8 – Comparison between the real airports network with a random network created with the appropriate number of nodes and links as predicted by the Erdos-Rényi model.

of nodes with few links. We compute that by making a histogram of the degree distribution (figures 9-a) and 9-b)).

Those results are not surprising as it would be a really poor example otherwise. The real-world network appears to follow a power-law while the random graph follows a binomial distribution. Due to its ubiquitousness one of the benchmarks that we will use creates networks whose degree distribution follows power laws.

1.1 Communities

After this brief review it's finally time to talk about the property that motivated this study in the first place: the presence of communities in the networks. Intuitively, when we talk about communities we think in tightly-knit groups of people, be it by a common interest or goal, affinity or simply by physical location. In spirit, the concept remains the same when viewed in the context of networks. Instead of looking for tightly-knit groups of people, we are searching for tightly-knit groups of nodes, but what it means exactly to be



(a) Airports Network. (b) Random Network

Figure 9 – a) Histogram of the degree distribution from the airports network, we see that this distribution appears to follow a power law. b) Histogram of the degree distribution from a random graph with the same number of nodes as the airport network from figure 3 and an equivalent number of links, we see that this distribution is binomial.

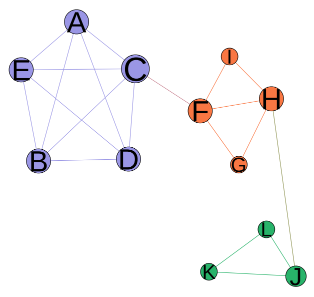


Figure 10 – Intuitive example of what we call the community structure of a network. The different communities, nodes A through E, F through I and J to K are identified by their colors. Note that there's no overlap, every single node needs to be in one and only one community.

tightly connected in terms of a network? While there's no consensual formal answer to that question, all definitions agree that members of a community must be more connected between themselves than with the rest of the network.

The community structure of a network is defined by a partition P: to every node in the network a number between 1 and m is assigned indicating to which one of the c_m communities it belongs. Each node can only be assigned to one community so that $c_i \cap c_j = 0$ for $i \neq j$ and that $\bigcup_{i=1}^{m} c_i = N$. Also, if every community has n_c nodes, the expression $\sum_c n_c = N$ is true. With this definitions the number of possible ways to divide

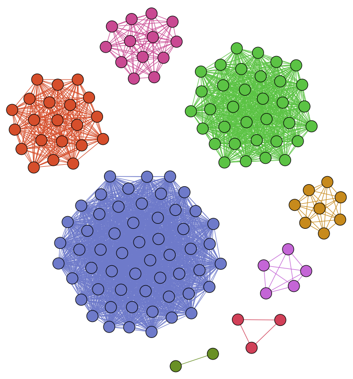


Figure 11 – Cliques of different sizes

a network in m clusters in the limit of large N is the n-th Bell number (LOVÁSZ, 2007):

$$B_N = \frac{1}{\sqrt{N}} \lambda(N)^{N+1/2} e^{\lambda(N)-N-1}, \tag{1.12}$$

where $\lambda(N)$ is defined by $\lambda(N) \log \lambda(N) = N$. This means that the number of partitions grows faster than exponentially, which in turn makes it impossible to evaluate all the partitions computationally. Because of that, communities are often defined by an algorithm. For example, we will have a quality function that measures how good or bad a partition of the network is, and an algorithm will look for partitions that best fit this criteria, thus returning the community structure. Different algorithms may have different quality functions. Going back to our analogy with groups of people, an algorithm may group them by location, another by common goals and interests, the result of the algorithms will still be communities even though the methods didn't start with the same definition of what a community is.

The concept of community closely relates to the idea of density of connections, a clique, a structure where every pair of distinct vertices is connected, represents the maximal possible density of connections in a graph and is, therefore, of great significance in the context of community detection. The RC and our benchmark use cliques to seed the initial community structure in the networks. Some examples for different sized cliques are shown in figure 11 bellow, each clique has $L = \frac{N^2-N}{2}$ connections, where N is the number of nodes inside the clique.

To shed light in the community structure of the different benchmarks, we will use the following metrics: the number of communities, the distribution of community sizes and the Pielou's Index, PI (PIELOU, 1966). Additionally, in order to compare the community structure among different partitions we will use the Variation of Information, VI (MEILĂ, 2003). From those, the latter two deserve a proper explanation.

The Pielou's Index was introduced as a way to measure the diversity of species in a population. As the interpretation is pretty straightforward in an ecological frame, we'll start by borrowing those terms and then translate them to a network science point of view.

Diversity in a biological population is related to the uncertainty in predicting the species of a single individual chosen at random from that population. The higher this uncertainty the more diverse the population is. A simple example is a population consisting of only one species, say 1000 dogs, if we pick a member at random from this set, we are 100% sure that it'll be a dog, hence it's the least diverse population possible. If instead we had, 30 dogs and 20 cats, we can certainly say that the chance of picking a dog at random is higher that picking a cat, however there's an uncertainty associated with this claim, hence this population is more diverse than the first one, in fact, we will later see that a population with an equal number of cats and dogs is the most diverse possible in this case. In short, Pielou's index is a way to quantify this uncertainty and by extension to measure how diverse a population is.

In mathematical terms, the Pielou's index is related to the Shannon entropy H by

$$\mathrm{PI} = \frac{H}{H_{\max}}, \tag{1.13}$$

this value ranges from 0 to 1, when PI= 1, we have the most diverse population possible. To define Shannon entropy we need some extra assumptions, let's assume that the population contains S different species, and that the individuals of each of those species compose a fraction p_i of the overall population, Shannon entropy is then

$$H = -\sum_{i=1}^{S} p_i \log p_i, \tag{1.14}$$

this expression reaches its maximum when an individual has an equal chance of being from any of the species in S, in other words, when all p_i are equal, we have then

$$H_{\max} = -\sum_{i=1}^{S} \frac{1}{S} \log \frac{1}{S} = \log S. \tag{1.15}$$

Pielou's index, also called Pielou's evenness index, reaches its maximum when $H = H_{\max}$, this means that, when we have a large number of species evenly distributed, diversity is at its highest, conversely, when individuals of only a single species compose a major fraction of the population diversity is compromised.

Going back to the network notation, suppose we have a partition P with m communities, the chance of a node being in a certain community j is given by:

$$\mathcal{P}(j) = \frac{n_j}{N}, \tag{1.16}$$

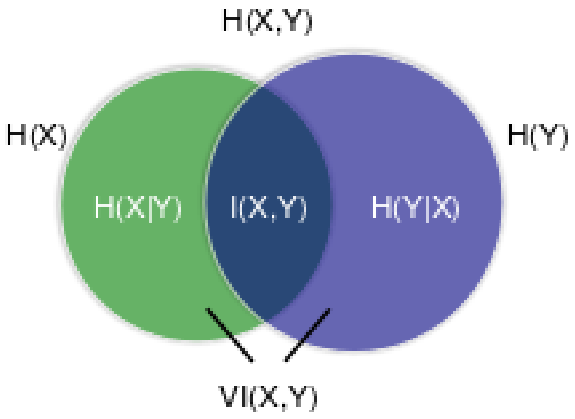


Figure 12 – Veen Diagram. The blue and green circles represent the entropies of partitions Y ($H(Y)$) and X ($H(X)$), respectively. The joint entropy $H(X,Y)$ is the area enclosed by both circles, whereas $VI(X,Y)$ is the area outside the intersection between $H(X)$ and $H(Y)$, the bigger the intersection the smaller the value of the VI. When the intersection is maximum, as in, when $H(X)$ and $H(Y)$ are equal, the VI is zero.

this value represents precisely the p_i from equation 1.14, so that H of partition P becomes:

$$H(P) = -\sum_{j=1}^{m} \mathcal{P}(j) \log \mathcal{P}(j). \tag{1.17}$$

As an example, a partition where every node is in it's own individual community has a PI of one, whereas the network of cliques in figure 11 has a PI = 0.74. Shannon's entropy will also be the basis for the definition of the variation of information. Imagine we have two partitions X and Y, with $\{c_i, \ldots, c_m\}$ and $\{c'_{i'}, \ldots, c'_{m'}\}$ communities respectively, the probability that a given node belongs to community c_m in partition X and community $c'_{m'}$ in partition Y is given by:

$$\mathcal{P}(X,Y) = \frac{|c_m \cap c'_{m'}|}{N}, \tag{1.18}$$

the associated entropy $H(X,Y)$ is called the joint entropy and is defined as

$$H(X,Y) = -\sum_{m}\sum_{m'} \mathcal{P}(X,Y) \log \mathcal{P}(X,Y), \tag{1.19}$$

the variation of information can be written as a function of the joint entropy and the entropies of partitions X, $H(X)$, and Y, $H(Y)$, respectively. In this form, the variation of information is:

$$VI(X,Y) = 2H(X,Y) - H(X) - H(Y). \tag{1.20}$$

The variation of information is always non-negative, assuming values from 0 to $\log N$, we can construct a normalized version of the VI easily by dividing equation 1.20 by $\log N$. After the division the VI will always be between 0 and 1.

To get an intuitive feel for the VI we can use the Veen diagram depicted in figure 12. The entropy $H(X)$ is represented as the green circle, while $H(Y)$ is the blue one. The joint entropy $H(X,Y)$ is the area enclosed by both circles, and $VI(X,Y)$ is the area outside the intersection of $H(X)$ and $H(Y)$. If $H(X)$ and $H(Y)$ are equal, which means they intersect completely, VI is zero. This is easy to see from expression 1.20, in this case $H(X,Y) = H(X) = H(Y)$, so the equation resumes to $VI(X,Y) = 2H(X) - H(X) - H(X) = 0$.

If the intersection between $H(X)$ and $H(Y)$ is zero, VI is maximum. Again we use equation 1.20, but now we have $H(X,Y) = H(X) + H(Y)$ so that $VI(X,Y) = 2(H(X) + H(Y)) - H(X) - H(Y) = H(X) + H(Y)$.

2 Algorithms

We tested our Surprise maximization algorithm against 7 others from the literature. Those seven algorithms are divided in four categories, the ones based on Potts models, consisting of CPM, from Constant Potts Model (TRAAG; DOOREN; NESTEROV, 2011), RB, from Reichardt and Bornholdt (REICHARDT; BORNHOLDT, 2006), and RN, from Ronhovde and Nussinov (RONHOVDE; NUSSINOV, 2010), the ones based on Hierarchical Clustering, SCluster (ALDECOA; MARÍN, 2010) and UVCluster (KING; PRŽULJ; JURISICA, 2004), the ones based on optimization of a quality function, Louvain (BLONDEL et al., 2008) and our own Surpriser, and the one based on information theoretic principles, Infomap (ROSVALL; BERGSTROM, 2008).

Methods based on Potts models (CPM,RB, and RN) aim to find the ground state of the Hamiltonian of a spin glass associated with the network. Infomap on the other hand, seeks to minimize the description length of a random walk through the networks links [1]. UVCluster and SCluster, perform an iterative hierarchical clustering in the network, then group nodes that stay close together in those different clusterings. Lastly, Louvain and Surpriser maximize a quality function called the Modularity and the Surprise, respectively.

The implementation of Louvain algorithm is available in the Python NetworkX package (HAGBERG; SWART; CHULT, 2008), whereas the other algorithms, with the exception of Surpriser, can be found in the SurpriseME Python distribution (ALDECOA; MARÍN, 2013b). The distribution contains an implementation of the meta-algorithm described in (ALDECOA; MARÍN, 2011). Here different algorithms are run and the partition that yields the highest value of Surprise is chosen. Because of that, some algorithms (CPM, RB, UVCluster and SCluster), already use the Surprise function as an auxiliary criteria. We will describe that in greater detail in the next sections.

This chapter is divided as follows, first, we have a section on the Louvain algorithm, as it is the most popular one, then UVCluster and SCluster are clustered together in the Hierarchical Clustering section, next, Infomap, as the only informational theoretical method, gets a section of its own, CPM, RB and RN are grouped in the Potts Model subsection and lastly we have a section dedicated to our own Surpriser.

[1] The Potts Models and Infomap also aim to optimize a quality function, in this sense, the division in different sections is purely didactic and based on the way the quality functions are defined.

2.1 Louvain

Before delving in the algorithm it is interesting to spend a few words describing the Modularity Q function(NEWMAN, 2006). Considering unweighted networks, Modularity is defined as

$$Q = \frac{1}{2L} \sum_{i,j} \left(A_{ij} - \frac{k_i k_j}{2L} \right) \delta(c_i, c_j), \tag{2.1}$$

where k_i and k_j are the degree of nodes i and j respectively, L is the number of links in the network, $\delta(c_i.c_j)$ is a Kronecker delta and A_{ij} is an element from the adjacency matrix A.

This definition uses a null model where the network retains its degree distribution but each link emerges at random. In this scenario the probability, $\mathcal{P}_{ij}$, of a link between nodes i and j existing is given by $\mathcal{P}_{ij} = \frac{k_i k_j}{2L}$. This is the second term from equation 2.1 and means that good modules, in a Modularity landscape, are modules where the actual number of links (first term in equation 2.1) exceed the expected number of links in the null model (the second term in equation 2.1).

Now we can talk about the algorithm itself. In the beginning each node is in its own community, so that there is N communities in the partition. Nodes are moved to the community that produces the highest gain in Modularity. To do that, for every node i and for each of its j neighbours, the change in Modularity caused by moving i to the community of j is calculated. The node i is then placed in the community that increases the Modularity the most. Note that this is a greedy algorithm so only positive changes in the quality function are accepted. Once no single movement can increase the Modularity, Louvain enters its second phase. The network is collapsed, communities found in the first phase become nodes in the new network, the edges of this network are weighted according to the number of existing links between the original communities, connections inside communities become self-loops in this network[2]. Nodes in the collapsed network are moved in the same way as before, as in, to produce the largest gain in Modularity. The two phases are repeated until no increase in Modularity is possible. One can consider the networks formed by collapsing the communities into single nodes as intermediate solutions, which one can represent as a dendrogram and use to study the networks hierarchy.

2.1.1 Issues with Modularity

Modularity maximization methods are the most widespread form of community detection and as such Modularity deserves a more detailed analysis. It has already been shown that, despite its popularity, Modularity suffers from a resolution (FORTUNATO; BARTHÉLEMY, 2007; GOOD; MONTJOYE; CLAUSET, 2010) and a degenerancy

[2] Adapting equation 2.14 for weighted networks is really simple. In place of the degree k_i we use w_i (the sum of the weights of node i) and in place of L we use the sum of the weights over the entire network.

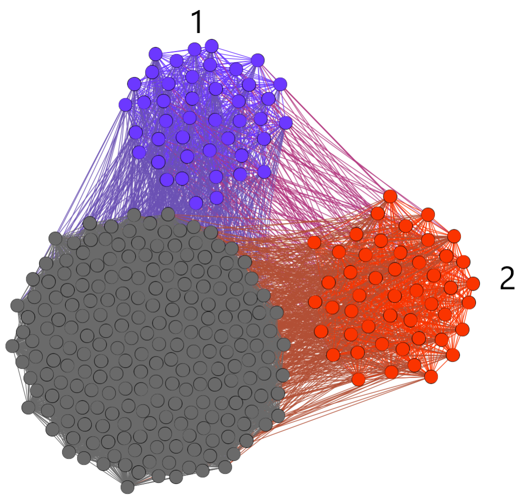


Figure 13 – Example graph to study the resolution limit from the Modularity function. We want to check the conditions by which the modules 1 (in blue) and 2 (in red) are connected. The largest group in gray has a sub-community structure but it's omitted for clarity.

problem (GOOD; MONTJOYE; CLAUSET, 2010; NICOLINI; BIFONE, 2016). We will start by discussing the former.

The resolution limit comes from the null model used in the definition of Modularity (GOOD; MONTJOYE; CLAUSET, 2010). In order to show it mathematically, imagine we have a structure as figure 13. We seek to find the necessary conditions to join communities 1, in blue, and 2, in red. As the best partition is the one that maximizes Modularity, combining 1 and 2 will be optimal if $\Delta Q > 0$, we need therefore to find the scenarios where this happens.

Before that, let's rewrite the expression from equation 2.1 in a more convenient form. The first term is equivalent to the number of edges in a given community e_m and is written as $e_m = \sum_{ij} A_{ij}\delta(c_i, m)\delta(c_j, m)/2$ while the second term is equivalent to the square of the sum of the nodes degrees d_m^2, $d_m^2 = \sum_{ij} k_i k_j \delta(c_i, m)\delta(c_j, m)$. Equation 2.1 then becomes:

$$Q = \sum_m \left(\frac{e_m}{L} - \frac{d_m^2}{4L^2} \right). \tag{2.2}$$

The initial Modularity value (before joining) is given by:

$$Q_i = \frac{e_1}{L} - \frac{d_1^2}{4L^2} + \frac{e_2}{L} - \frac{d_2^2}{4L^2} + \sum_{i=3}^{m} \left(\frac{e_i}{L} - \frac{d_i^2}{4L^2} \right), \tag{2.3}$$

after joining them we have:

$$Q_f = \frac{e_1 + e_2 + e_{12}}{L} - \frac{(d_1 + d_2)^2}{4L^2} + \sum_{i=3}^{m} \left(\frac{e_i}{L} - \frac{d_i^2}{4L^2} \right), \tag{2.4}$$

note that the number of intra-community links is now equal to the number of links in modules 1 and 2 plus the number of links between them (e_{12}), to compute ΔQ we subtract equation 2.3 from equation 2.4:

$$\Delta Q = \frac{e_{12}}{L} - \frac{d_1 d_2}{2L^2}. \tag{2.5}$$

As we seek to maximize the Modularity a change is accepted whenever $\Delta Q > 0$, using this and putting the expression 2.6 in a general form, yields:

$$e_{ij} > \frac{d_i d_j}{2L}. \tag{2.6}$$

Modularity will join two groups whenever the number of connections between them, e_{ij}, exceeds the expected number of edges in a random graph with the same degree distribution (the term on the right). This is a problem because as the network gets bigger, and hence L increases, the expected value of $\frac{d_1 d_2}{2L}$ gets smaller to a point where $\langle e_{ij} \rangle < 1$, in this case only one edge between two modules $e_{ij} = 1$ is enough for Modularity to join them.

We can show this problem explicitly by analysing the ring of cliques network. In this case, we have a set m of cliques with the same size n, each connected to their neighbours in such a way as to form a ring (figure 14). The ideal partition should keep the cliques separated, let's check the conditions for joining them as given by equation 2.6. First we know that, by definition, the number of edges between each clique is always one, hence $e_{ij} = 1$, further, the quantity d_i is equal for each of the m cliques, so we have:

$$1 > \frac{{d_i}^2}{2L}. \tag{2.7}$$

The total number of edges L is given by m times the number of connections inside each clique plus the m connections joining them. As each clique has $n(n-1)/2$ internal edges, L is given by $L = m(n(n-1)/2 + 1)$. By using equation 1.3, we obtain the sum of the nodes degrees $d_i = n(n-1) + 2$ (the 'extra' two comes from the two external edges in each clique). Plugging these results in 2.7 gives

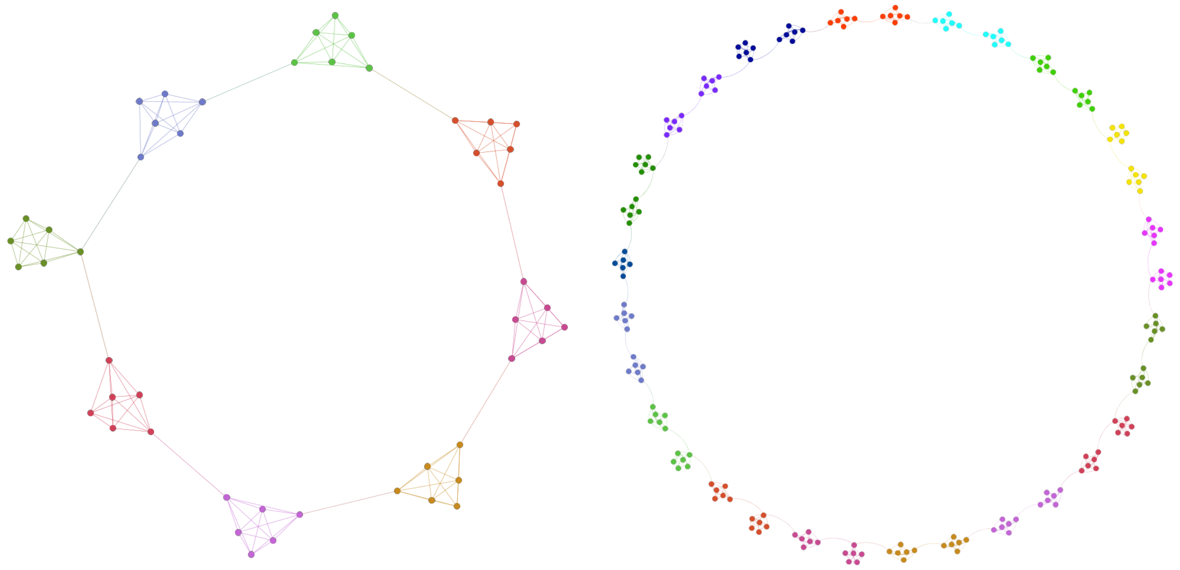
$$2m \left(\frac{n(n-1)}{2} + 1) \right) > (n(n-1) + 2)^2, \tag{2.8}$$

solving for m, yields:

$$m > n(n-1) + 2. \tag{2.9}$$

This means that for every size of clique n, Modularity will start joining them after a certain value of m. Cliques of size 6 will be united for m bigger than 32, as we see in figure 14 b). In resume, the bigger the network the bigger the communities. A qualitative analysis of this problem can be seen in (ALDECOA; MARÍN, 2013a; NICOLINI; BIFONE, 2016) and a similar analytical proof is given in (GOOD; MONTJOYE; CLAUSET, 2010).

Regarding the degeneracy problem (GOOD; MONTJOYE; CLAUSET, 2010) whenever two or more partitions have the same (or a similar) value of Modularity and those values are very close (or equal) to the global maximum we say that those partitions



(a) Ring of cliques network with 8 cliques of size 6. (b) Ring of cliques network with 33 cliques of size 8.

Figure 14 – Partitions found by using the Louvain algorithm for the ring of cliques network with different numbers of cliques. After a certain point, Modularity starts combining two neighbouring cliques in the same community (b), we show that this behavior happens for every clique size.

are degenerated. The work by Good, et al(2010) showed that in Modularity's case this problem is very pervasive, occurring to both benchmark networks, consisting of hierarchical random graphs (CLAUSET; MOORE; NEWMAN, 2008) and ring of cliques networks, and to real networks, consisting of the metabolic networks from *T. palli- dum*,*Mycoplasma pneumoniae* and *Ureaplasma parvum*.

What this means is that, for every network there is a plethora of different partitions that have a Modularity value very close to the global maximum of the function,but, not only that, upon comparing the different near-optimal solutions they showed that the discrepancies arise from the largest modules identified, or in other words, the different near-optimal partition do not agree on the large scale community structure of the networks. Lastly, they tested if some simple statistics remained consistent between these degenerated solutions, namely the average module density and the distribution of modules sizes. The result was that, while the average value of those parameters was similar, their variance was large.

In resume, although wildly used, Modularity suffers from two main problems, it has a resolution limit, where smaller modules may be merged into larger ones even if they are poorly connected and its solutions are highly degenerated, meaning that many different partitions have values of Modularity close to optimum and that the discrepancy between those nearly-optimal partitions is large. Surprise on the other hand, does not seem to suffer from these problems in quite the same way, as we will discuss later.

2.2 Hierarchical Clustering Algorithms

The concept of hierarchy in community detection algorithms relates to the existence of communities within communities, that small groups of nodes can be joined to form bigger ones in a structured manner. One way to represent the hierarchy is to construct a dendrogram or a tree of the system, both UVCluster and SCluster use this idea. First the actual clustering algorithms are applied on the network then based on their results a second algorithm creates the tree (or dendrogram) of the partition found.

The networks adjacency matrix A_{ij} is the input for both algorithms, from this, each algorithm performs the following:

- UVCluster: The nodes are ordered randomly in a list then the first node from this list is selected. The largest possible clique is made from that node neighbours. All the nodes in this clique form a community and are removed from the list. This process is repeated until no nodes are left without a community.
- SCluster: Also starts with a random ordered list of the nodes. The first node is selected then all of the neighbouring nodes are joined in a community and removed from this list. Once again this process is repeated until no nodes are left without a community.

Those methods are applied until N solutions are obtained, in our runs we set $N = 10000$. The goal here is to combine those results to find the ideal partition. To do that, a distance matrix is defined in the following way: the distance between two nodes is equal to the number of times they were placed in different communities divided by the total number of runs N. With this matrix the dendrograms are created using the UPGMA (SOKAL, 1958) algorithm. After all is said and done, the division of the tree that maximizes the Surprise is chosen.

2.3 Infomap

The gist of the Infomap algorithm is to recover the network's communities from the analysis of the information flow in the network. A community in this regard is a group of nodes where the information flows easily and quickly between its members, furthermore the links between those modules represent the pathways from which the information travels between communities (ROSVALL; BERGSTROM, 2008).

The question is, then, how to represent the information flow in a network. Rosvall and Bergstrom propose that a random walk among the network's nodes reflects the information flow, because a random walk is defined by the networks topology and nothing else and it offers a quick way to generate the dynamics with only the network description.

With that in mind the problem of identifying the community structure boils down to finding an efficient way to describe the paths that arise from the random walks performed in the network. In computational terms, the aim is to find the code that allow us to describe the entirety of random walk in the least amount of bits, *i.e.*, to minimize the per-step description length ($\mathcal{L}$) of the random walk.

The most straightforward way to describe a random walk is to assign a different name (or code) to each node and then write a sequence where each term is the name of the node visited in each instant of time. In their work, this idea is improved in two ways.

First, the name of the nodes are set based on the Huffman Code (HUFFMAN, 1952), which gives longer names to rarer events and smaller names to more frequent (common) events. As we are talking about a random walk, the node's names are given based on their visit frequencies in an infinitely long walk (their ergodic visit frequencies).

Second, they opt for a two-level description: communities and nodes are named. Each community is assigned an unique name and exit code -used whenever the random walker moves from a community to the next. Nodes receive unique names inside the communities, but names are repeated among different communities. Borrowing the analogy made in their original paper, think of communities as cities and nodes as street names, it's very common for cities to have streets with the same name but its very rare for neighbouring cities to have the same name.

Working this all out we obtain the expression for the average per-step description length of the random walk

$$\mathcal{L} = \mathrm{q}H(\mathcal{Q}) + \sum_{i=1}^{m} p^i H(\mathcal{P}^i), \tag{2.10}$$

the first term is the entropy of movement between communities and the second is the entropy of movement inside communities. They represent the average length in bits for movements between and inside modules, respectively. In this expression, $H(\mathcal{Q})$ is the entropy of community names, $H(\mathcal{P}^i)$ is the entropy of movements within communities, p^i is the fraction of within-module movements that occur inside community i and q is the probability that the random walker switches communities on any given step.

Before continuing it's important to note that the algorithm does not perform a random walk on the network per se, but rather models the random walk mathematically and uses this model to calculate $\mathcal{L}$. The random walker movement is described as follows: with a probability τ the random walker will teleport to a random node in the network (if the node does not have any link $\tau = 1$), then with the remaining $1 - \tau$ probability, the random walker will move to one of the node's neighbours with a $1/k_i$ chance of going to each one (remember we're considering undirected and unweighted networks here). Given this considerations the process can be modelled as an irreducible and aperiodic Markov-Chain. This means that, by the Perron-Frobenious theorem, the system has a unique steady state that describes the ergodic node visit frequencies p_j. Now here's the catch, the expression (2.10) can be written as a combination[3]of p_j and q, which can be easily obtained. The algorithm uses the power method to find the values of p_j (note that p_j does not depend on the actual partition of the network) and with them calculate $\mathcal{L}$ for a given partition.

After that the task of minimizing $\mathcal{L}$ becomes simple. The algorithm starts by calculating the ergodic visit frequencies by the power method, then the nodes are placed in their own community (there's an equal number of communities and nodes). The values of p_j are used to calculate the exit probabilities q of this initial partition. Then a greedy algorithm is used: partitions are merged in a way that decreases the value of $\mathcal{L}$ the most. Once no merge is able to decrease the per-step description length, the partition obtained by the greedy algorithm is refined by a simulated annealing (SA) method. The SA algorithm is run from 10 different starting temperatures. From those results, the partition with the lowest value of $\mathcal{L}$ is returned by Infomap.

2.4 Potts Model Algorithms

All of the following algorithms aim to find the ground state of a Hamiltonian associated with the network. As it is usual with community detection methods we want to reward connections inside communities while penalizing the missing links in those groups (TRAAG; DOOREN; NESTEROV, 2011; REICHARDT; BORNHOLDT, 2006). Of course, each algorithm will define its Hamiltonian in a different way. We start by presenting the Constant Potts Model (CPM), which is formalized as

$$\mathcal{H}_{\text{CPM}} = -\sum_{ij}(A_{ij} - \gamma_{\text{CPM}})\delta(c_i, c_j), \tag{2.11}$$

where A_{ij} is the adjacency matrix, $\delta(c_i, c_j)$ is a Kroenecker delta and γ_{CPM} is a constant. Hence the name Constant Potts model. Continuing with our analogy the first term in equation 2.11 is reward for intra-community links and γ is the cost for missing links. In this model γ acts as a threshold by which communities are formed or dismantled. Every group inner link density must be higher than γ_{CPM} whilst the outer link density remains smaller than γ_{CPM}.

The algorithm by Reichardt and Bornholdt(REICHARDT; BORNHOLDT, 2006) uses the following Hamiltonian

$$\mathcal{H}_{\text{RB}} = -\sum_{ij}(A_{ij} - \gamma_{\text{RB}} p_{ij})\delta(c_i, c_j), \tag{2.12}$$

to avoid confusion we call the constant γ_{RB} in this case, all the other terms are the same from equation 2.11 with exception of p_{ij} that represents the probability of having a link

[3] For the details of this calculation refer to the supporting material from (ROSVALL; BERGSTROM, 2008).

between nodes i and j. The choice of p_{ij} depends of the null model chosen, in the scenario where all links occur with the same probability we can simply write $p_{ij} = p = 1/2L$ where L is the number of links in the network. A more interesting formulation establishes a dependence of p_{ij} with the degrees of the nodes i and j. This is done in the form $p_{ij} = \frac{k_i k_j}{2L}$, note that we still assume that the nodes occur randomly, however, nodes with higher degrees should have a higher chance of being connected. If we plug this definition of p_{ij} in 2.12 it yields

$$\mathcal{H}_{\mathrm{RB}} = -\sum_{ij} \left(A_{ij} - \gamma_{\mathrm{RB}} \frac{k_i k_j}{2L} \right) \delta(c_i, c_j), \tag{2.13}$$

from here if we set $\gamma_{\mathrm{RB}} = 1$ and rearrange the remaining terms we obtain the formulation of Modularity (Q)

$$Q = -\frac{1}{L}\mathcal{H}. \tag{2.14}$$

The implementation of RB we ran uses the null model defined by equation (2.13). It's trivial to see the relationship between the CPM and RB models, just set $\gamma_{\mathrm{CPM}} = \gamma_{\mathrm{RB}} p_{ij}$. The last of the algorithms based on Potts model is the one by RONHOVDE; NUSSINOV, which has the form

$$\mathcal{H}_{\mathrm{RN}} = -\frac{1}{2} \sum_{ij} \left(A_{ij}(1 + \gamma_{\mathrm{RN}}) - \gamma_{\mathrm{RN}} \right) \delta(c_i, c_j). \tag{2.15}$$

We've seen the form of the Hamiltonian for each of the three algorithms, though two questions still remain. How each of these algorithms seeks the minimum value of $\mathcal{H}$ and how to choose the best value of γ in each case. CPM and RB use a greedy algorithm akin to Louvain in order to minimize the Hamiltonian, while RN performs a Simulated Annealing to search for the minima. Furthermore, in the implementation used all three algorithms test different values of γ and choose among them using the following criteria: CPM and RB, return the γ that yields the highest value of Surprise, while RN returns the partition with the best and most stable value of the Normalized Mutual Information (NMI). [4]

2.5 Surpriser

2.5.1 Surprise

Surprise is an accumulated hyper-geometric distribution that measures how surprising (unlikely) it is to find the same distribution of intra and extra-community links in

[4] The Normalized Mutual Information is another information theoretic index that compares two different clusters. It can be written in terms of the VI by using:

$$NMI = \frac{H(X) + H(Y) - VI(X,Y)}{2\sqrt{H(X)H(Y)}}. \tag{2.16}$$

Note that the NMI takes values from 0 to 1. Whenever the clusters are identical we have $NMI = 1$.

a random network with the same community structure. It is defined as follows:

$$S = -\log \sum_{j=\zeta}^{\min(\mathrm{M,L})} \frac{\binom{M}{j}\binom{F-M}{L-j}}{\binom{F}{L}}, \tag{2.17}$$

where F represents the total number of possible links in the network (it's equivalent to the number of links in a clique of size N, $F = (N(N-1)/2)$), and L represents the actual number of links in the network. As F and L are characteristics of the network size, equation (2.17) is in fact a function of only two variables, ζ and M, that depend on the network's actual community structure. Here M represents the maximum number of intra-community links and ζ the actual number of intra-community links in the partition. For a network with m communities where each community has n_c nodes, M is given by:

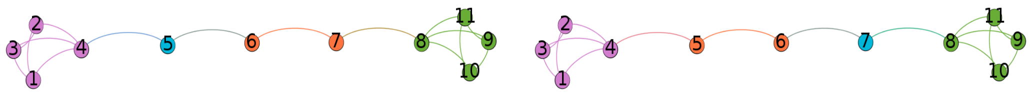
$$M = \sum_{c=1}^{m} \frac{n_c(n_c-1)}{2}, \tag{2.18}$$

M is maximum whenever all communities are fully connected (formed by cliques of size n_c). Going back to equation 2.17, the term on the denominator represents the number of ways to draw L links from F possible links, *i.e.*, the number of different ways to draw the actual network. The first term in the numerator depends only on M and ζ and as such refers to the intra-community links in the network. It represents the number of ways to draw j intra-community links from a pool of M possible intra-community links. In the last term, note that $F - M$ represents the maximum number of extra-community links whereas $L - j$ represents the actual number of extra-community links. The combination of both is equivalent to the number of different ways to draw the extra-community links.

The formulation of equation 2.17 is also equivalent to an urn problem. In this scenario every *possible link* F represents a ball in the urn, a number of them is red $(F - M)$ and a number of them is blue (M). Red balls represent failures (extra-community links) and blue balls represent successes (intra-community links). For every partition the proportion of blue and red balls is different (as M is a function of the community structure). Surprise measures the probability of obtaining at least ζ successes (the number of intra-community links in the partition) in L draws (as there's L links in the actual network) without replacement.

In resume, it is not worthwhile to add a poorly connected link into a community because, while there will be an increase in the value of ζ the induced change in M will be far greater. To give an example we have the network of figure 15. The structure that maximizes the Surprise is composed by 4 communities: two cliques formed by nodes 1 through 4 and 8 through 11, nodes 5 and 6 grouped together and node 7 by itself in another community. Trying to add nodes 7 or 5 to any of the cliques lowers the value of Surprise found, this happens because as much as there's an increase in the number of intra-community links, from 6 to 7 in each case, the number of potential intra-community links goes from 6 to 9 which yields a lower Surprise overall.

A more diligent reader might ask why nodes 5 and 6 are coupled together and not nodes 6 and 7? Well, truth is, both nodes 5 and 7 are indistinguishable. The partition that joins nodes 6 and 7 and leaves node 5 alone also maximizes the Surprise (figure 15 a). When different solutions represent the same maximum value of the quality function, or values very close of that maximum, we say that those solutions are degenerated. *Good et al.* (2010) did a wonderful work analysing the degeneracy problem in the context of Modularity maximization methods. Here we apply similar principles to Surprise maximization and will later show that this problem is not as prominent in a Surprise landscape.



(a) Partition a. (b) Partition b.

Figure 15 – (a) and (b) Communities of a network with 11 nodes and 16 links. Adding a poorly connected node such as 5 or 7 to the four nodes cliques (nodes 1 through 4 and 8 through 11) lowers the Surprise of the structure. (b) Partitions (a) and (b) share the same maximum value of the Surprise function. This is indicative of the degeneracy problem that will be treated later.

2.5.2 Algorithm

Before getting into the details of our algorithm, we will do a brief review of the earlier work in relation to the usage of the Surprise as quality function. In their initial works, Aldecoa and Marín (2011), created a meta-algorithm using the Surprise function. Surprise was the criteria by which they selected the best result among many different algorithms. They showed that, this meta-algorithm outperformed any individual algorithm (ALDECOA; MARÍN, 2011). Later, they attacked the problem of Surprise maximization by proposing an asymptotic approximation for the function (TRAAG; ALDECOA; DELVENNE, 2015).

Methods	Description
Merge	Combines two communities. Two communities, A and B, are joined to form a new community C.
Extract	Removes an element from a community and forms a new community with it. For example, a node or sub-community from community A is removed to form a new community B.
Exchange	An element is moved from a community to another one. A sub-community from A is removed and placed in community B.

Table 1 – Methods of the Surpriser package. The merge function works on a community level, while Extract and Exchange deal with nodes and sub-communities.

We don't use those puny tactics, our greedy algorithm aims to maximize the Surprise as defined by equation 2.17.

The core of our algorithm is made of three basic operations, which we call merge, extract and exchange (described in table 1). Merge works on a community level, while extract and exchange may be used in a sub-community level as well. The merge operation joins two communities together, *i.e.*, members of modules A and B are grouped to form a new community C. The extract operation removes an element from a given community and creates a new, separated community with it. Finally, the exchange operation moves an element from a community to another one. The extract and exchange methods don't deal with communities as a whole, so in this context an element may be either a single node or a sub-community (the sub-communities of a community are determined applying recursively the algorithm to the sub-graph formed by the community elements alone).

The algorithm starts with a loop over all communities. First it performs all the possible merges that increase the Surprise. This is done as follows: in each community it runs a loop over its members and try, until exhaustion, to merge the current community to one of its members neighbours. If it fails (there is no merge that increases the Surprise) it tries to exchange nodes between them. Once no improvement can be made by merging communities nor exchanging nodes between communities, it executes the extract operation over every community. It begins by extracting nodes and then sub-communities. Finally it tries to exchange sub-communities between different communities. The respective pseudo code can be found in algorithm 1, in this case, a success means that an operation was able to increase the value of the Surprise. As it's a greedy algorithm, a change in the community structure is only accepted if the value of Surprise increases. So when there's no possible merge left, we mean that no merge increases the Surprise value of the partition. Further, at each step we perform the change that increases the Surprise the most from all possible options.

In greater detail, we begin with the operations on a community level, so we make all

```
Data: input is a network with a partition;
while Any change done to the partiton do
    for every community in the partition do
        for every node in the community do
            for every neighbour of the node do
                if community of neighbour $\neq$ community then
                    try to merge communities and if fails, try to exchange element
                      between them;
                end if
            end for
        end for
    end for
    while successful do
        try to extract an element from community;
    end while
    while successful do
        try to extract a sub-community from community;
    end while
    while successful do
        for every other community in partition do
            try to exchange a sub-community with the other community;
        end for
    end while
end while
```

Algorithm 1: Surpriser Algorithm

the merges until no improvement is possible. With this result we move to operations in the sub-community level. First, we try to exchange and then extract nodes until exhaustion. Lastly, we try to extract and then exchange sub-communities. Note that, unlike the Louvain algorithm, that basically only merge communities, our Surpriser may also divide communities into smaller groups with the extract and sub-community extract operations.

For the initial conditions of the network we choose to put each node in it's own individual community, so that in the beginning of the algorithm there is the same number of communities as there are nodes. This choice is also very common in the literature, being the initial conditions of Louvain, RB, RN, CPM, UVCluster, SCluster.

3 Benchmarks

As we saw previously a definitive formal definition of community does not exist. Because of that most algorithms have their own definition of a proper community or partition. This becomes a problem when we want to compare results among different algorithms as each algorithm will find good modules given their own definition, but which solution is the best overall?

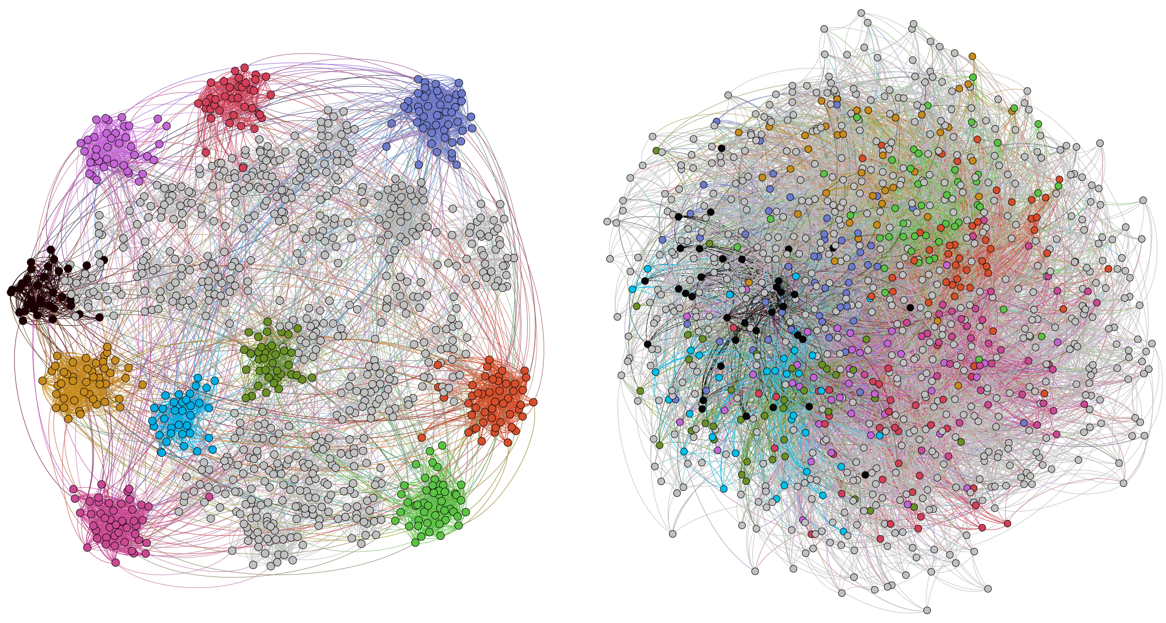
The easiest way would be to find real world networks where the communities are already known, run the algorithms and finally compare the results. However, in most networks the community structure is not known a priori, so this kind of data is not readily available. We are limited to networks where the modules are a given, this happens in social networks where its members (humans) actively know their associations. As there aren't many works in this respect, the most common way to evaluate the performance of the algorithms - and the one we will use in this work, is through the usage of benchmark networks. A benchmark network is an artificially created network with an embedded community structure, i.e, each of the benchmark networks is created with an initial partition that represents its community structure. This is advantageous because it allows us to perform a direct comparison, through the VI, between the results of the algorithms and the planted partition. The closer to zero the VI is, the better the result of the algorithm.

We will study three different benchmarks, LFR (from Lancichinetti-Fortunato-Radicchi), RC (Relaxed Caveman) and a novel benchmark proposed by us. The next sections are a review of both LFR and RC benchmarks, followed by an explanation of our new method.

3.1 LFR Benchmark

The LFR benchmark (LANCICHINETTI; FORTUNATO; RADICCHI, 2008) creates networks that are scale-free. Even though there are discussions about the pervasiveness of this characteristic in real world networks, this benchmark is a hallmark in the literature being used in a plethora of works (TRAAG; DOOREN; NESTEROV, 2011; ALDECOA; MARÍN, 2011; YANG; ALGESHEIMER; TESSONE, 2016).

Both the degree distribution and the community sizes distribution of these networks follow power-laws. Their exponents β and γ, respectively, are parameters of the benchmark and can be set by the user. *Yang et al.* (2016) have shown, however that the most important feature of this benchmark is the mixing parameter μ. This value, that ranges from 0 to 1, represents the fraction of links that each node has with elements outside its community.



(a) $\mu = 0.1$.

(b) $\mu = 0.5$

Figure 16 – LFR networks generated with $\beta = 1$, $\gamma = 2$, $\bar{k} = 15$, $k_{\max} = 50$ and $N = 1000$ nodes. In a) and b) the 10 largest communities were colorized, we see that the larger the value of μ the fuzzier the community structure.

Conversely, the number $1 - \mu$, represents the fraction of links between members of the same community. In a sense, for values above $\mu > 0.5$, we don't have a strong definition of community anymore, as every single node will have more connections with members outside its community than with members inside its community. That hasn't stopped us to create networks with μ bigger than 0.5, but this factor has to be kept in mind.

To create the networks, six parameters are used. The network's average degree $\bar{k}$, the network's maximum degree $k_{\max}$, the number of nodes N, the power-laws exponents, β for the community sizes and γ for the degree distribution, and the mixing parameter μ of the network. The process can be described as follows, first, to each node i a number of links k_i is assigned to fit the power-law distribution and the average and maximum values of k. Next, the communities are assigned, until no node is left without a community. Finally, the connections are rewired so that the value of μ is respected. In this last step, the degree k_i of each node remains the same, only the proportion of intra and extra-community links is altered (the connections are 'shuffled' in the network until μ is reached).

In our examples we set the node degree exponent γ as 2, the community sizes

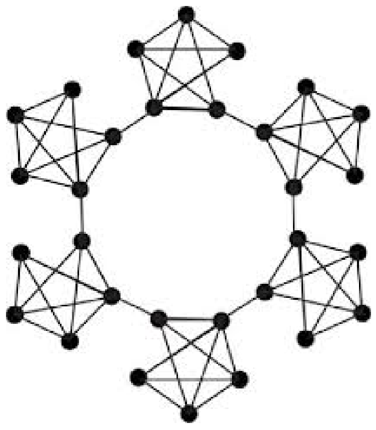


Figure 17 – Example of a connected caveman graph.

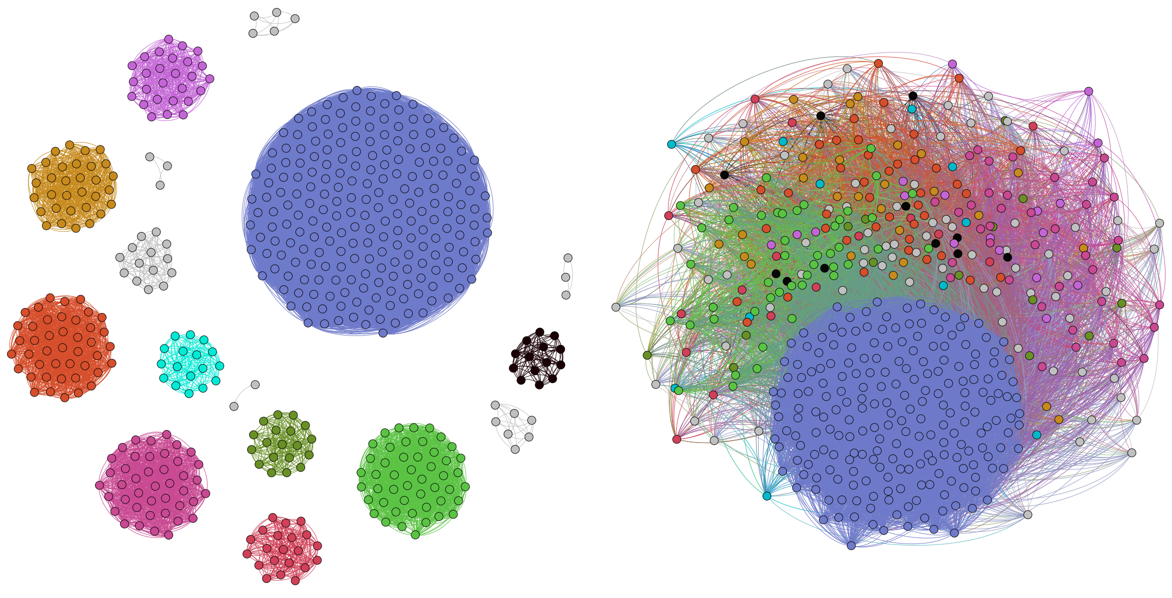
exponent β as 1, $N = 1000$, $k_{\max} = 50$, and $\bar{k} = 15$. Figure 16 shows the effect of the increase in μ in the community structure.

3.2 RC Benchmark

The Connected Caveman Graphs (figure 17) come from the social sciences and are defined by a set of connected cliques. To draw them we remove a link from each clique and use it to connect the neighboring cliques in such a way as to form a ring.

The RC benchmark as proposed in (ALDECOA; MARÍN, 2011; ALDECOA; MARÍN, 2010) also starts with a set of cliques, in this case however, they are isolated. Note that this is the strongest possible definition of community, a set of fully connected subgraphs isolated from each other (figure 18-a). As identifying these communities is trivial, the RC benchmark degrades these networks. The degradation process is made in the following way: a percentage R of its links are randomly removed and the same percentage R of the *remaining links* are shuffled between the nodes. This means that for each network created we have the original structure, that represents the isolated cliques, and a number n of additional structures, one for every value of R, that represent the network in different stages of the degradation process (figure 18-b). The task presented here is for the algorithms to identify this original structure in the degraded networks.

The *RC* benchmark uses three parameters to create the networks, the number of nodes N, the number of cliques N_c and the Pielou's index (PI) for the distribution of clique sizes. We set these parameters to 512, 16 and 0.75, respectively. Theses choices were made to create networks that differ greatly from the LFR benchmark ones (a more detailed discussion on the PI of both benchmarks is made on the results section) and to minimize the computational time, as the number of connections grows with the square of the clique size.



(a) $R = 0$. (b) $R = 50$

Figure 18 – Communities of a RC network generated with $N = 512$ nodes, 16 cliques and $PI = 0.75$. In a) and b) the 10 largest communities were colorized. a) The original community structure, represented as a set of 16 disjointed cliques. b) The same network after the degradation process with $R = 50\%$, here 50% of its links were removed and the same percentage of the remaining links were shuffled in the network.

3.3 New Benchmark

The new benchmark we propose share some similarities with both the LFR benchmark and the RC benchmark. As the RC benchmark we start with a set of cliques, however their sizes are given by a power law distribution [1] in the same way as the communities sizes from the LFR benchmark. To that set of cliques a fraction r of nodes is randomly connected to the network. Intuitively, we are trying to combine a perfect definition of community, a fully connected clique, with a much weaker one, a single link node.

Unlike the other benchmarks we implemented two parameters to control the degradation process of the initial communities. First, a link may be created between any pair of nodes from two different cliques with probability q, and then each link inside a community may be removed with a second probability p. Without further ado, the creation

[1] Note that the minimum size of a clique and, by extent, the minimum value of the power law distribution are equal to 2.

of a benchmark network follows the steps:

- Creation of N_c cliques with sizes given by an scale-free distribution with parameters $x_0 = 2$ (minimum click size) and γ_{SF}. Remember that a clique with size 1 is a single disconnected node.
- A fraction r of nodes is randomly connected to the network. Each of those nodes has exactly one link
- Each link inside each clique i with size c_i may be lost with probability p.
- Given any two cliques of sizes c_i and c_j, each one of the $c_i c_j$ possible connections between the two is created with probability q. In other words, any pair of nodes i and j from any two cliques c_i and c_j has a probability q to be connected.

So the benchmark network starts from a sequence of N_c clicks sizes, $c_1, c_2, ..., c_{N_c}$. The number of nodes in a network is

$$N = \frac{1}{(1-r)} \sum_{i=1}^{N_c} c_i, \tag{3.1}$$

the expected number of links inside communities is

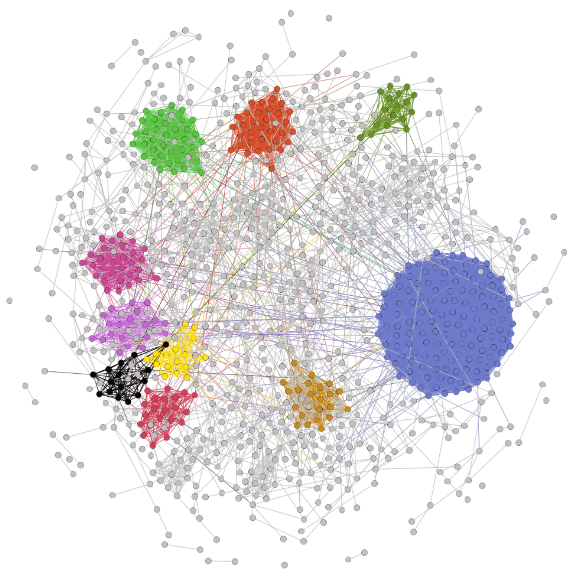
$$\langle L_{in} \rangle = \sum_{i=1}^{N_c} (1-p) \frac{c_i(c_i-1)}{2}, \tag{3.2}$$

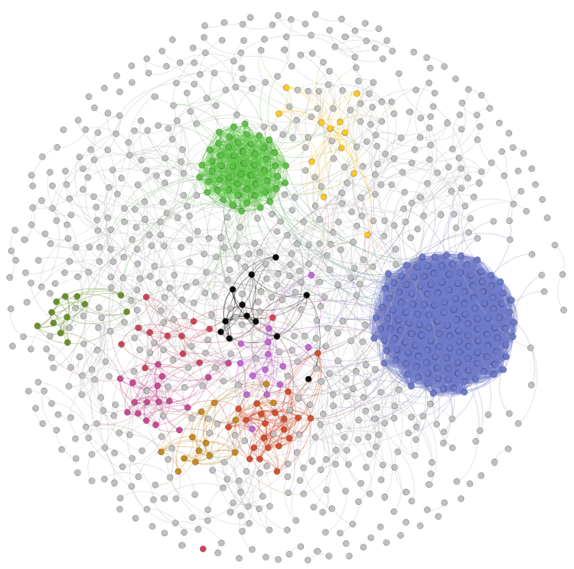
and the expected number of links between communities is

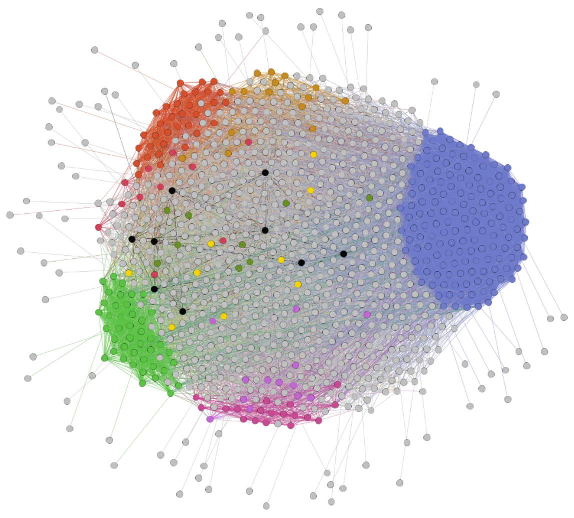
$$\langle L_{out} \rangle = q \sum_{i=1}^{N_c} \sum_{j<i}^{N_c} c_i c_j + rN. \tag{3.3}$$

Combining both we have, for the total expected number of links:

$$\langle L \rangle = \sum_{i=1}^{N_c} (1-p) \frac{c_i(c_i-1)}{2} + q \sum_{i=1}^{N_c} \sum_{j<i}^{N_c} c_i c_j + rN \tag{3.4}$$


(a) $q = 0.001$ $p = 0.10$


(b) $q = 0.001$ $p = 0.50$


(c) $q = 0.015$ $p = 0.10$

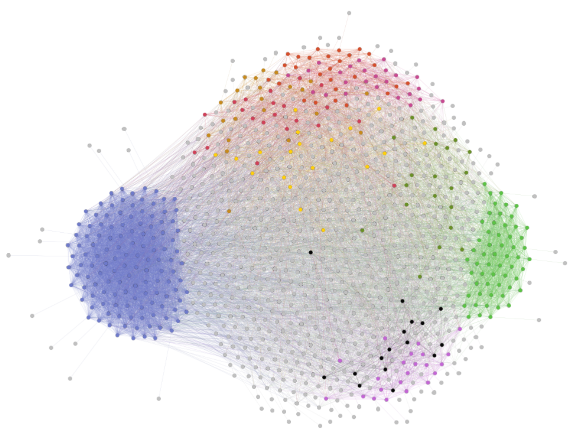

(d) $q = 0.015$ $p = 0.50$

Figure 19 – Different networks created using our novel benchmark. We see the two ways the community structure is weakened. As q increases the number of extra-community links grows, shown by the majority of the grey links in figures 19 c) and d), and as p increases the number of intra-community links diminishes, as given by the fuzziness in the community structure in b and d. As always the 10 biggest communities were colorized. The scale-free proprieties are evident, as we have a small number of large cliques joined by a huge proportion of smaller groups.

4 Results and Discussions

We start this chapter with a statistical description of the benchmark networks created (section 4.1). Next, we present the results (sections 4.2 and 4.3). We compare the performances of the different algorithms and then study the degeneracy problem in the landscape of the Surprise function.

4.1 Benchmark Networks

In total, we created 13200 networks: 900 from the LFR benchmark, 3300 from the RC benchmark and 9000 networks from our benchmark. Before we dig into the statistics let's describe our choice of parameters for each of the benchmarks.

The 900 networks from the LFR benchmark share the following characteristics: the node degree exponent γ is equal to 2, the community sizes exponent β is equal to 1, the number of nodes N is equal to 1000, the maximum degree $k_{\max}$ is equal to 50 and the average degree $\bar{k}$ is equal to 15 [1]. The mixing parameter μ takes values between 0.1 and 0.9, so that $\mu = 0.1,0.2,0.3,...,0.9$. For each of those values of μ, 100 networks were created.

Regarding the RC benchmark, we start with 500 different networks. They are formed by 512 nodes divided into 16 cliques. Further, the distribution of clique sizes is given in such a way that the networks Pielou's index is approximately 0.75. To get the 3300 RC networks, we degrade the 500 initial networks to various points. A percentage R of links were removed and the same percentage R of the *remaining links* were shuffled among the network's nodes. For values of R between 10% and 50% all of the 500 networks were degraded (accounting for 2500 of the 3300), while for $R \geq 60\%$ we only degraded 200 of those networks (the remaining 800). We limited our analysis at large values of R because most algorithms perform very badly in that region, as the definition of community no longer holds.

For our benchmark we have two parameters: p (the probability that a link is removed from a clique) and q (the probability that a link is created among nodes from different cliques). We varied p from 0 to 0.95 in intervals of 0.05, and q from 0.000 to 0.016 in intervals of 0.001. Alongside that interval we also studied $q = 0.020$ and $q = 0.030$. For q bigger than 0.030 the number of links grows rapidly which makes the analysis costly in terms of computational time. At $q = 0.030$ the number of extra-community links is around six times the number of intra-community links in our benchmark, *i.e.*, we are already

[1] Upon further analysis, even tough similar choices of parameters are quite common in the literature (TRAAG; DOOREN; NESTEROV, 2011; ORMAN; LABATUT, ; GREGORY, 2010), the values of $\bar{k}$ and $k_{\max}$ may be a poor choice for this value of γ.

delving in a weak definition of community. In resume, we have 20 possible values of p and 18 possible values of q, for each combination of q and p 25 networks were created, this yields a total of 9000 networks from our benchmark.

We can compare the parameters of the three different benchmarks by calculating the values of p and q for the RC and LFR networks. In table 1 we present those results. To find p we determine the ratio between the actual number of links inside a community (l_c) and the maximum possible number of links inside that community ($n_c(n_c-1)/2$). This ratio is the fraction of existing links inside a community, as p represents the fraction of *missing links* inside a community, we subtract the value obtained from one. Writing this out gives:

$$p = 1 - \frac{2l_c}{n_c(n_c-1)}, \tag{4.1}$$

We then take the average over all the communities to obtain the value of p for a given network. Table 2 shows the average value of p among the networks with a given value of R or μ. For q we have:

$$q = \frac{L-\zeta}{F-M}, \tag{4.2}$$

putting into words, q is the ratio between the actual number of extra-community links and the maximum possible number of extra-community links. Here we borrowed the terms from the definition of Surprise, F is the maximum possible number of links, M is the maximum possible number of intra-community links, whereas L and ζ are the actual number of links and the actual number of intra-community links, respectively. Again, table 2 shows the average of those values.

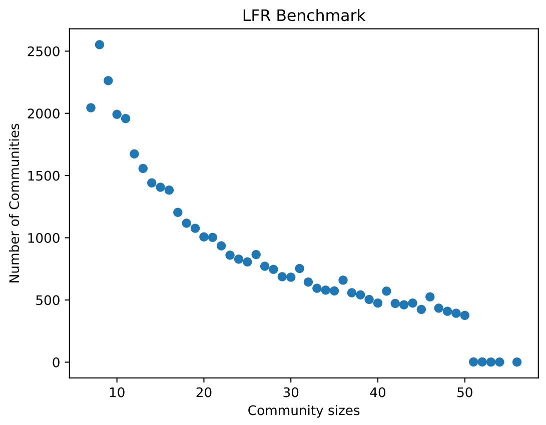
In the RC networks, we see that q increases for $R \leq 50\%$, and starts decreasing after that threshold. This happens because R influences the value of q in two antagonistic ways: on one hand as the links are randomly shuffled among nodes, q has a tendency to increase, alternatively, as R increases, more and more links are removed from the network, so that L gets smaller in relation with F (that remains constant) and so q decreases. For $R \leq 50\%$ the number of links shuffled balances the number of links being taken out of the network, but for higher values of R, the decrease in L is to steep to be overcome. Those two behaviors contribute to the increase in p seen in table 2, as links are being shuffled among communities, intra-community links are becoming extra-community links, further, when links are removed from the network the actual number of links inside a community decreases while the maximum number of links stay the same.

Regarding the LFR benchmark we see steady increases in both p and q as μ gets bigger. This comes directly from the definition of μ. Higher values of μ mean that the fraction of extra-community links increases in comparison with the intra-community links, this yields higher values of p, as more intra-community links are missing, and q, as we have more extra-community links.

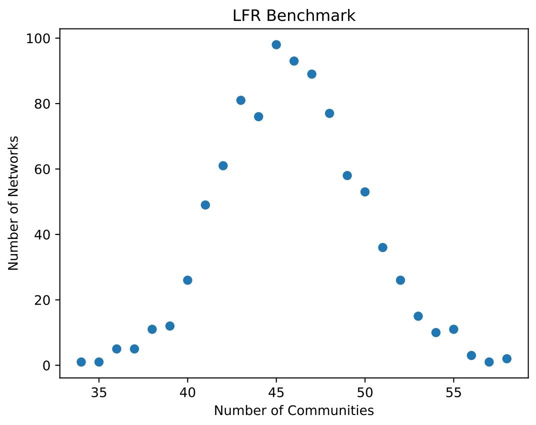
	RC Benchmark								
	10%	20%	30%	40%	50%	60%	70%	80%	90%
p	$0.188365^{0.161830}_{0.092242}$	$0.353417^{0.183596}_{0.134805}$	$0.492930^{0.153563}_{0.180735}$	$0.623775^{0.154420}_{0.163208}$	$0.725296^{0.121798}_{0.167390}$	$0.809591^{0.091014}_{0.159088}$	$0.878878^{0.065495}_{0.165902}$	$0.935670^{0.041866}_{0.117412}$	$0.974553^{0.019027}_{0.080002}$
q	$0.020294^{0.002379}_{0.001781}$	$0.034999^{0.003950}_{0.002995}$	$0.044720^{0.004895}_{0.003730}$	$0.049853^{0.005319}_{0.004064}$	$0.050836^{0.005311}_{0.004050}$	$0.047586^{0.005336}_{0.003793}$	$0.040977^{0.004522}_{0.003234}$	$0.030761^{0.003301}_{0.002378}$	$0.017087^{0.001804}_{0.001335}$

	LFR Benchmark								
	0.1	0.20	0.3	0.4	0.5	0.6	0.7	0.8	0.9
p	$0.376852^{0.157362}_{0.215281}$	$0.415103^{0.160781}_{0.208957}$	$0.462157^{0.157159}_{0.202404}$	$0.515839^{0.159186}_{0.198517}$	$0.572580^{0.155071}_{0.188106}$	$0.638958^{0.144407}_{0.176287}$	$0.707239^{0.127260}_{0.159852}$	$0.792614^{0.097731}_{0.149043}$	$0.891979^{0.051451}_{0.092870}$
q	$0.001580^{0.000037}_{0.000034}$	$0.003150^{0.000069}_{0.000060}$	$0.004751^{0.000077}_{0.000100}$	$0.006308^{0.000151}_{0.000124}$	$0.007901^{0.000180}_{0.000166}$	$0.009442^{0.000158}_{0.000218}$	$0.011050^{0.000185}_{0.000214}$	$0.012582^{0.000246}_{0.000238}$	$0.014178^{0.000295}_{0.000288}$

Table 2 – Values of p and q in the RC and LFR benchmarks. The uncertainties shown around the average are standard deviations calculated for values bigger and smaller than the average, respectively.



(a) Community sizes.



(b) Number of communities.

Figure 20 – Distribution of the community sizes and the number of communities for all the LFR networks.

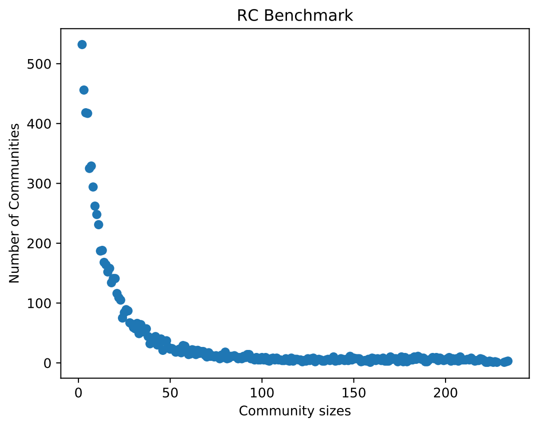
Now we can talk about the statistics. Remember that each network has a seeded community structure given by the benchmark (which we will call initial partition from now on) and that our goal is to compare the partitions obtained from each algorithm with the initial ones. Because of this, we are particularly interested in the statistics that involve the community structure, hence we will analyse the community sizes, the number of communities and the Pielou's index of the networks.

As always, we will start by commenting on the LFR and RC benchmarks and then we'll talk about the results from our benchmark. In figure 20 b) we see that the LFR benchmark produces partitions with something around 45 communities and that those communities share a similar size, ranging from 7 to 56 nodes in each. For the RC benchmark we have a smaller number of communities, by definition each network has 16 communities (the initial cliques), but with a much wider range of sizes, from 2 to 234 nodes (figure 21).

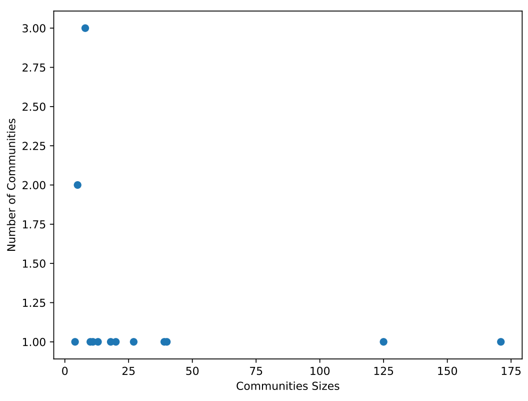
Pielou's index (PI) measures how even is the distribution of community sizes. An index equal to one means that all the communities have the same size. In agreement with figures 20 and 21, we see that the LFR networks (figure 22-a) have a PI close to one, $PI_{\text{LFR}} = 0,961 \pm 0,005$, whilst RC networks (figure 22-b) have a PI of $PI_{\text{RC}} = 0,751 \pm 0,003$ (this result is expected as we set PI$= 0.75$ when we created the RC networks).

In our networks, the clique sizes follow power laws (for sizes equal to, or higher than 2), which we can clearly see from the figure 23-a). Alongside that, a fraction of the initial partition is made from isolated nodes in their own communities, this makes it so that the average number of communities is the highest among the benchmarks as shown in figure 23 b). This two characteristics yield an average PI of 0.87 ± 0.05.

In resume, RC benchmarks have the smallest number of communities and the

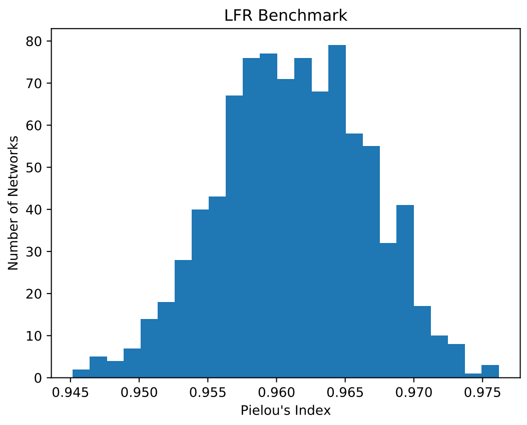


(a) Community sizes.

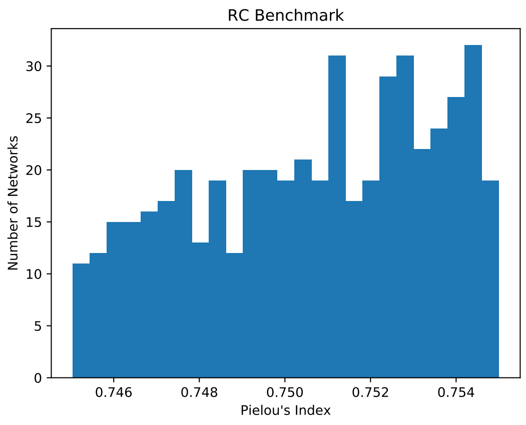


(b) Community sizes for a single network.

Figure 21 – Distribution of the community sizes for all the RC networks, remember that, by definition the RC networks have 16 communities each so that the plot of the distribution of number of communities is rather dull (as it would always be equal to 16). Also, to emphasize that the distribution in a) is from all the networks, we have the distribution of community sizes from a single network in b)

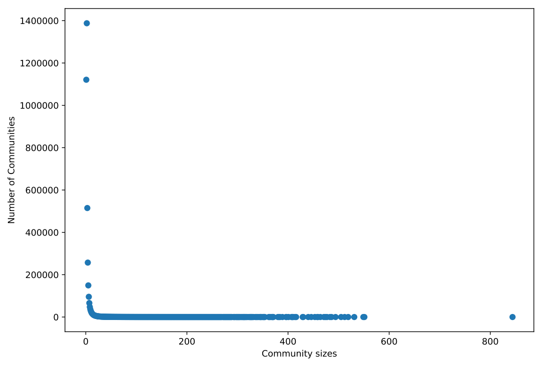


(a) LFR benchmark.

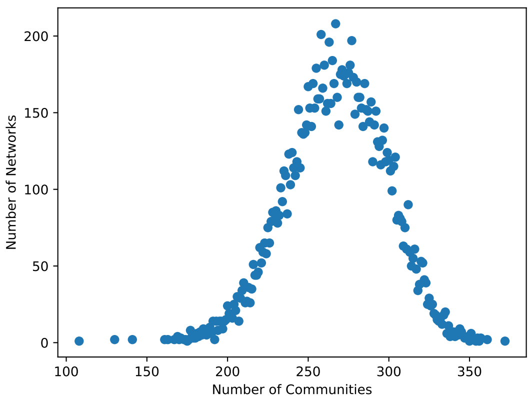


(b) RC benchmark.

Figure 22 – Pielou's index for the LFR (PI= $0,961 \pm 0,005$) and RC (PI= $0,751 \pm 0,003$) benchmark networks. The numbers show that the LFR networks have a much more even distribution of community sizes than the RC networks. Also remember that the PI_{RC} was set at 0.75 in the networks creation.



(a) Community sizes.



(b) Number of communities.

Figure 23 – Distribution of the community sizes and the number of communities for all the networks in our benchmark.

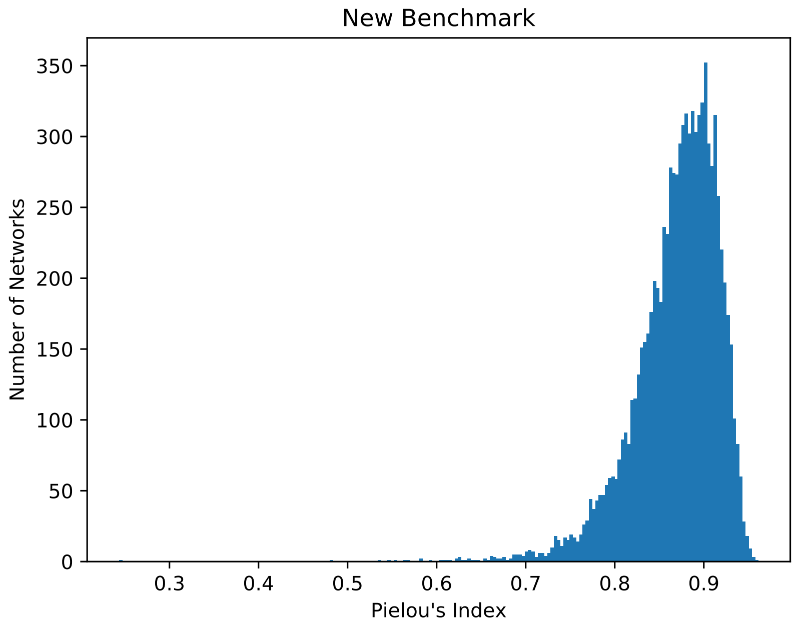


Figure 24 – Pielou's index for the networks in our benchmark. The average PI of 0.87±0.05 is nicely placed in the middle of the interval between the PI of the RC benchmark (0.75) and the PI of the LFR benchmark (0.96).

	Pielou's Index	Number of Communities	Community Sizes
LFR	$0.961 \pm 0,005$	$[34, 58]$	$[2, 234]$
RC	$0.751 \pm 0,003$	16	$[7, 56]$
New	$0.87 \pm 0,005$	$[108, 372]$	$[1, 844]$

Table 3 – Ranges of community sizes, number of communities and the respective Pielou's index for each of the benchmarks.

smallest Pielou's index, this means that the distribution of community sizes is the most uneven of the three benchmarks. The LFR benchmarks have an intermediary number of communities but they all have a similar size as they have the highest PI. Figure 20-a) also shows that the communities are relatively small if compared to the other benchmarks. Our benchmark, on the other hand, has the highest number of communities with an intermediary value of PI, this means that we have lots of very small communities but some very large ones as well. Table 3 contains a summary of the statistics.

4.2 Algorithms Performance

We quantify the quality of the results using the VI between the two partitions, the one found by the algorithm and the one seeded by the benchmark. A value of VI equal to zero means that the two partitions coincide perfectly, whereas a value of 1 means that no information can be learned from one partition using the other one. The ultimate goal is to create an algorithm that is able to find a VI very close to zero for each network in every benchmark. Here what we truly seek is consistency, as the community structure of real networks is most often unknown *a priori*, an algorithm that has a good performance in every benchmark is preferable over an algorithm that is great in some benchmarks but that performs poorly in another ones.

Keep in mind while reading through those results that some methods, CPM, RB, UVCluster and SCluster, already use the Surprise as a criteria to choose among different partitions. None of them seek to maximize the Surprise heads-on, as our Surpriser does, but γ_{RB},γ_{CPM} are choosen because they give the partition with the highest value of Surprise. The same is true for UVCluster and SCluster, where they pick the partition that maximizes the Surprise after applying the tree making algorithm. Infomap, Louvain and RN do not use the Surprise at all.

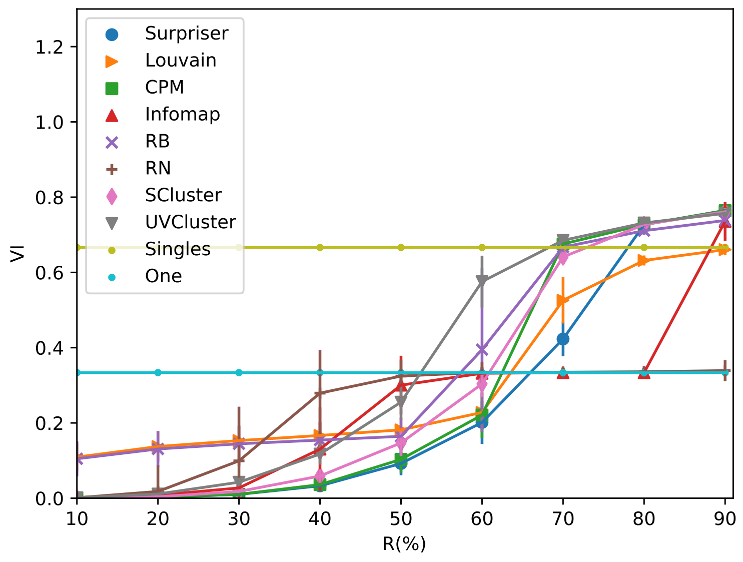
First let's analyze the results for the RC benchmark as replicated in figure 25. In the figure each dot represents the average VI found at each stage of the degradation parameter R. While it's clear that the smaller the value of the VI the better, the point where the results start to become unacceptable is not a clear-cut. To get some intuition about it we will use an idea from the SurpriseME paper (ALDECOA; MARÍN, 2013b). They created two additional partitions for each network, the first one called Singles and the later called One. In the Singles partition every node is in its own individual community, while in the One partition, every single node is in the same community (so that there's only one group that encompass every node). In figure 25 they represent the two horizontal lines. Whenever the VI found by every algorithm crosses any of those two lines we consider that they failed miserably to find the community structure of those networks. For the RC benchmark, every algorithm passes this threshold for values of R equal to, and higher than 70%. For $R > 70\%$ a huge part of the networks links has already been removed, this explains why the VI found tends to be closer to the single partitions, as the network has fewer links, the communities start to become smaller and less defined.

Focusing at $R < 70\%$ we see that CPM and Surpriser have the best performances. Notice that the VI found by Infomap and RN stabilizes for values of $R \geq 50\%$. In this range these algorithms begin to join all the nodes in the same community, obtaining a VI identical to the One partition, which hints that this is a limit behavior of these algorithms and therefore, they are actually failing beyond this point.

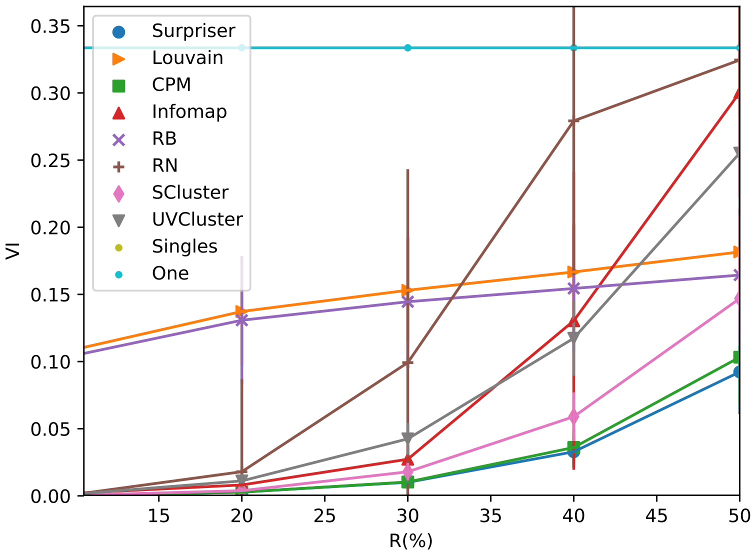
Methods that use Modularity (Louvain and RB) perform poorly in this benchmark as even for small R the VI is already big. This has to deal with the resolution limit of Modularity and will be discussed later. The limit also explains why they have a nearly identical performance for $R \leq 50\%$.

About the LFR networks (figure 26), RB and Louvain show an improvement in performance compared to the RC benchmark, with RB outperforming Louvain for the entire range of parameters. The best performance, though, is reached by Infomap. This algorithm still tends to find only one community when $\mu \geq 0.8$. Surpriser and CPM, on the other hand, stay in the middle of the pack. Finally, SCluster and UVCluster show the worst results. In contrast with the RC benchmark, the difference in VI between Singles and One partitions is much smaller. Splitting the nodes in individual communities is actually better than merging all the nodes in a single community in the LFR benchmark. We don't aim to do a quantitative analysis as to why this is the case, but these results match our intuition. The number of communities and the Pielou's index are higher for the LFR benchmark, this means that the partition is made of many small communities, in such a way that splitting the nodes individually is closer to the real community than merging all the nodes together, hence the VI should be lower for the Singles partition. Later we will show that this is the case in our benchmark as well.

Figure 27 summarizes the results from both benchmarks. Here we take the average VI over a range of parameters, $\mu \leq 0.5$ for the LFR benchmark and $R \leq 50\%$ for the RC benchmark. A pattern is evident here: algorithms that are great in one benchmark suffer in the other one, with two exceptions, our own Surpriser and CPM are able to function in

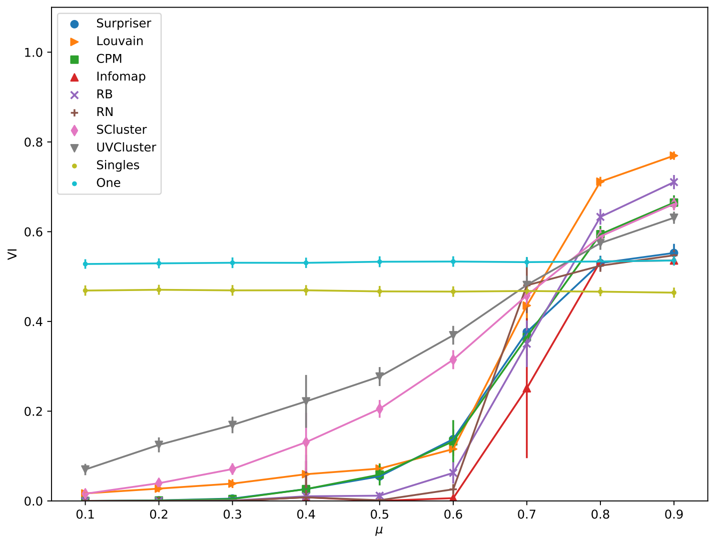


(a) RC Benchmark $10\% < R < 90\%$.

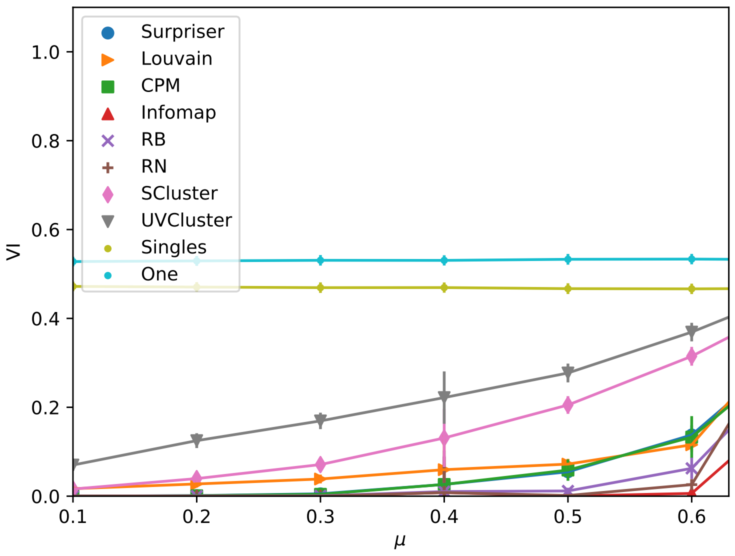


(b) RC Benchmark $10\% < R < 50\%$

Figure 25 – Performance of the algorithms in the RC benchmark.



(a) LFR Benchmark $0.1 < \mu < 0.9$.

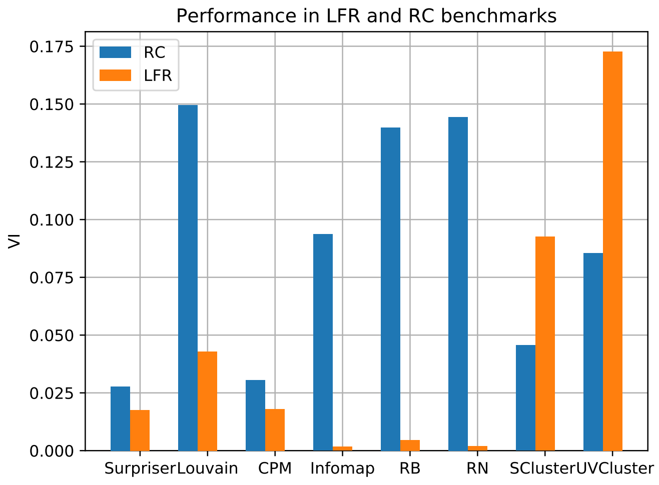


(b) LFR Benchmark $0.1 < \mu < 0.6$

Figure 26 – Performance of the algorithms in the LFR benchmark.

both LFR and RC networks. It's important to note that Infomap, RB and RN, perform superbly in the LFR benchmark, being better than CPM and Surpriser in those cases, however no single algorithm surpasses them in the RC benchmarks.

We also analysed the number of communities found by each algorithm compared to the number of communities created by each benchmark. Figures 28-a) and b) show this ratio c'_k/c_k, where c'_k is the average number of communities found by the algorithms and c_k is the average number of communities created by the benchmarks. We seek to find a



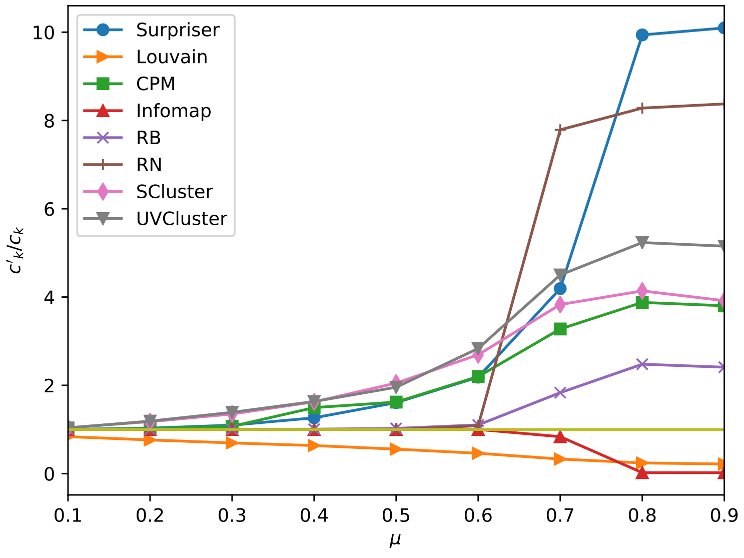
(a) $0.1 < \mu < 0.5$ and $10\% < R < 50\%$.

Figure 27 – Performance of the algorithms in both benchmarks, we see that only CPM and Surpriser have a consistently good performance in both benchmarks.

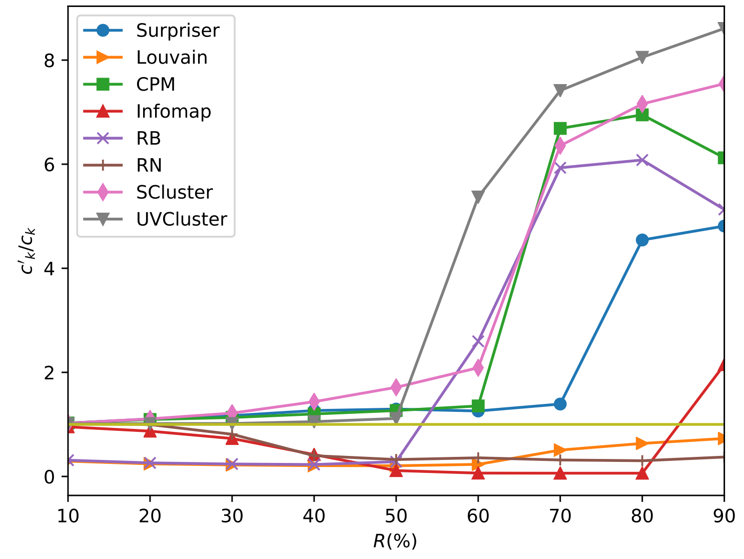
value close to one (the yellow line in both figures).

Infomap further proves its great performance on the LFR benchmark by matching the number of communities perfectly for small values of μ, however we see that this algorithm has a tendency to start merging communities together when the structure gets fuzzier, this problem boils down to joining every node in a single community as seen in figures 25 and 26. The difference in VI found by CPM and Surpriser in the LFR benchmarks seem to be caused by an overestimation of the number of communities in those networks, this starts slightly but as we cross the line at $\mu = 0.6$ and begin to have a weak definition of community (for values of $\mu > 0.5$ each node has more connections outside its initial community) both algorithms grossly miss evaluate the number of communities, with Surpriser finding 10 times more communities for $\mu = 0.9$. In the RC benchmark, conversely, Surpriser is the algorithm that stays closest to the initial number of cliques, only passing a factor of two for $R > 70\%$, this correlates to the strong performance of this algorithm in the benchmark. Even CPM, the second best performing algorithm, already finds 6 times more communities for $R = 70\%$, because of this we see the spike in the VI at $R = 70\%$ in figure 25.

Both RB and Louvain are based on maximization of Modularity, but they show very different behaviors in the different benchmarks, with Louvain consistently underestimating the number of communities in RC and LFR networks, while RB is able to solve the LFR benchmark with ease. This distinction is closely related to the resolution limit of Modularity. To show that let's analyse the values of γ_{RB} obtained in both benchmarks (figure 29). The higher the value of γ_{RB} the smaller are the communities found. Furthermore, whenever



(a) LFR Benchmark.



(b) RC Benchmark

Figure 28 – Ratio between the average number of communities found by the algorithms, c'_k, with the average number of communities seeded by the benchmarks c_k. A ratio of 1, where both c'_k and c_k match perfectly, is represented by the yellow lines.

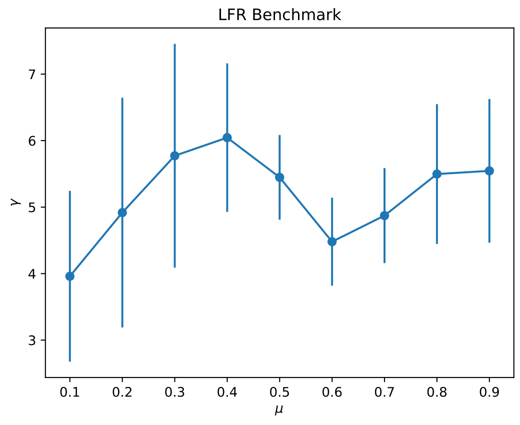
γ_{RB} is equal to one we recover the definition of Modularity used in the Louvain algorithm. We see in figure 29 that the values of γ_{RB} are way higher than one, this means that RB finds smaller communities and by extension more communities than Louvain in those networks. This matches perfectly with the plot in 28 a), where Louvain under evaluates the number of communities in the entire range of μ. This means that $\gamma_{RB} = 1$ is too small for those networks and that smaller modules are being wrongfully merged into bigger ones by the Louvain algorithm.

On the other hand, we see that $\gamma_{\text{RB}} = 1$ for smaller values of R (figure 29 b) which explains the mostly identical performance of both RB and Louvain in the RC benchmark (figure 25). As we increase R we should expect the communities to get smaller, as many links are being removed from the network (and some nodes might even become disconnected), the RB algorithm is sensitive to that and the ideal value of γ_{RB} also increases.

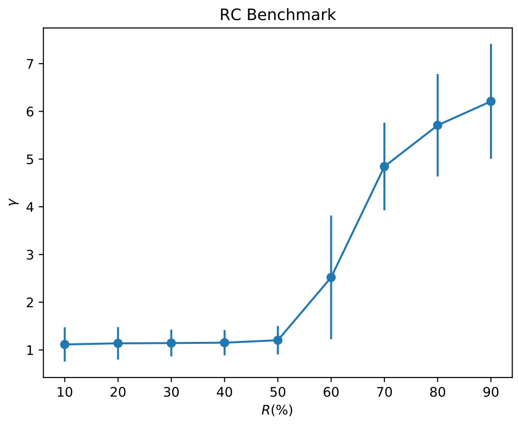
If we combine our analysis of the values of γ_{RB} (figure 29), with the number of communities found by the algorithms (figure 28) and the distribution of community sizes in both benchmarks (figures 20 and 21), we can clearly see the impact of the resolution limit in the performance of those algorithms. Networks in the LFR benchmark have a smaller range in community sizes, (also justified by the very high Pielou's Index) so that the RB algorithm is able to find a value of γ_{RB} that perfectly balances the tension between splitting larger communities into small ones and joining smaller communities into larger ones. That's why the VI found is so small for this algorithm. The ideal value of γ_{RB}, though, is way larger than the one used in Louvain, which is fixed at one. Because of that Louvain finds a smaller number of communities and has a worse performance in the LFR benchmark overall. The RC benchmark, conversely, has a wide range of community sizes with a $PI = 0.75$, because of that RB is not able to find a resolution that manages to solve both the very small communities and the very large ones successfully, hence the resolution limit truly hampers the application of Modularity based methods.

In relation to γ_{CPM}, the most interesting detail is in figure 30 b), where we see a spike in the value of γ_{CPM} at $R = 70\%$. This peak coincides with the increase in the VI figure 25 a) and with the increase in the number of communities detected figure 28. As γ_{CPM} is the threshold for the density of links inside communities ρ_{in} and the density of links between communities ρ_{out}, $\rho_{\text{out}} \leq \gamma_{\text{CPM}} \leq \rho_{\text{in}}$, a higher value of γ_{CPM} and a larger number of communities means that CPM finds smaller communities but more densely connected ones.

In general, the best performing algorithms in the LFR benchmark, Infomap, RN and RB, tend to underestimate the number of communities in the RC benchmark, with Infomap and RN being the most egregious cases as both algorithms tend to group every node in the same community for large values of R. From those two, the tendency to merge communities together seems to be more pervasive in the Infomap's case as it has

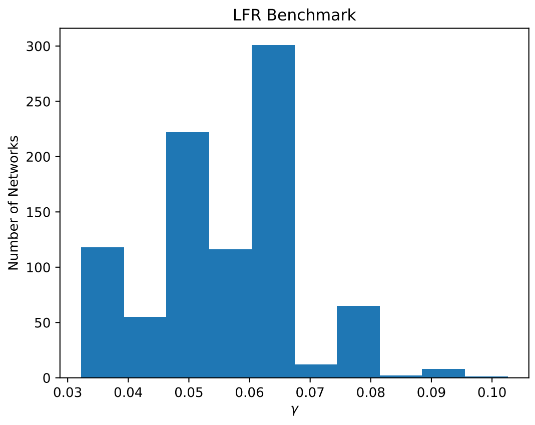


(a) γ_{RB} in the LFR Benchmark.

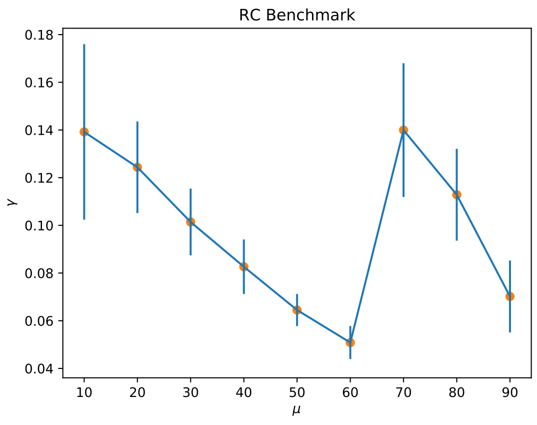


(b) γ_{RB} in the RC Benchmark.

Figure 29 – Values of γ_{RB} in the LFR and RC benchmarks. For reference $\gamma_{\mathrm{RB}} = 1$ recovers the definition of Modularity. The RB algorithm is able to solve LFR networks withe failing in the RC benchmark. This happens because in a), the distribution of community sizes is not broad so that the algorithm is able to find a value of γ_{RB} that balances the tension between splitting large communities and joining smaller ones. In b), conversely, the distribution of community sizes is wide, so that there's no clear value of γ_{RB} that balances the tension between dividing and merging communities. For small values of R we have the $\gamma_{\mathrm{RB}} = 1$ which explains the equivalent performance between RB and Louvain. Considering that $\gamma_{\mathrm{RB}} > 1$ in a) and that RB clearly outperforms Louvain, we see that the communities from the LFR benchmark are smaller than the ones found by Louvain, and that inserting a variation in γ_{RB} makes the algorithm sensitive to this change.



(a) γ_{CPM} in the LFR Benchmark



(b) γ_{CPM} in the RC Benchmark

Figure 30 – Values of γ_{CPM} in the LFR and RC benchmarks. In a) as the distribution of community sizes is even across the board we see little variation in the value of γ_{CPM}. In b) we see a major spike in the value of γ_{CPM} at $R = 70\%$, this coincides with the spike in the VI obtained and the spike in number of communities detected by the CPM algorithm. This means that CPM finds smaller communities but more densely connected ones.

this behavior in both benchmarks, whenever the community structure gets fuzzier the algorithm will lump all the nodes in a single group.

Surpriser and CPM, on the other hand, overestimate the number of communities in the LFR benchmark while providing the best results in the RC benchmark. These algorithms seem to be way more affected by changes in the value of μ. When we start to have $\mu > 0.6$ both algorithms have big spikes in the number of communities found, this isn't entirely bad as for those values of μ a strong definition of community isn't present anymore, so in a sense we would expect the algorithms to behave poorly. We don't see a resolution limit in these results, unlike those from Louvain that consistently underestimate the number of groups in both benchmarks.

The top algorithms in the RC benchmarks can, reasonably, solve the community structure of the LFR benchmarks, however, the opposite isn't true. The best algorithms in the LFR benchmark vastly underestimate the number of communities in the RC benchmark, even merging all the nodes in a single community for large values of R. From this analysis if the network has an unknown community structure both CPM and Surpriser are recommended over Infomap, RN and RB. Remember that the implementation of CPM obtains the community structure for different values of γ_{CPM} (figure 30 a and b) and then returns the partition with the highest value of Surprise.

The results of our benchmark will be presented as line plots for different values of q (figures 31 a,b,c and d), this will help with the readability and make it easier to compare with the results of both LFR and RC benchmarks. The the full heat maps for the VI in terms of both p and q are shown in appendix A. In figure 31 each point represents the average over 25 different networks with the same values of p and q.

We see that Modularity based methods perform very poorly in this benchmark. RB only finds partitions better than Singles for small values of both p and q while Louvain is always outperformed by the Singles partition. Methods that use the Surprise, directly or indirectly, seem to perform best with SCluster, CPM and Surpriser having the top three results.

In regard to the number of communities, recall figure 23, b), our benchmark has the highest number of communities with many of them representing single nodes, the ratio of communities found by the algorithms c'_k with the number of communities seeded by the benchmark c_k is in figures 32 a), b), c) and d). The algorithms underestimate the number of communities in our benchmark, as the value of p increases, however, we have fewer and fewer links in the network, as this happens nodes become disconnected and the community structure gets weaker, which in turn increases the number of communities found by the algorithms.

Figures 33 a), b) and c) as well as tables 4,5,6 summarize our results from all the

Algorithms	LFR	RC	New
Surpriser	$0.017528^{0.069310}_{0.014739}$	$\mathbf{0.027697^{0.057021}_{0.022564}}$	$\mathbf{0.067805^{0.050610}_{0.035050}}$
Louvain	$0.042910^{0.058849}_{0.019777}$	$0.149593^{0.044186}_{0.044537}$	$0.473898^{0.075747}_{0.151176}$
CPM	$0.018047^{0.055984}_{0.015756}$	$0.030582^{0.064456}_{0.025187}$	$0.075278^{0.055977}_{0.039609}$
Infomap	$\mathbf{0.001649^{0.807518}_{0.001635}}$	$0.093645^{0.216643}_{0.075500}$	$0.161008^{0.065963}_{0.058367}$
RB	$0.004599^{0.077874}_{0.004194}$	$0.139786^{0.057925}_{0.047264}$	$0.275888^{0.117593}_{0.099334}$
RN	$0.001926^{0.138579}_{0.001871}$	$0.144426^{0.187276}_{0.138909}$	$0.502669^{0.125536}_{0.463739}$
SCluster	$0.092519^{0.095622}_{0.056746}$	$0.045644^{0.082390}_{0.037831}$	$0.075117^{0.055880}_{0.038512}$
UVCluster	$0.172667^{0.085508}_{0.071940}$	$0.085508^{0.131145}_{0.068723}$	$0.101350^{0.070299}_{0.055560}$

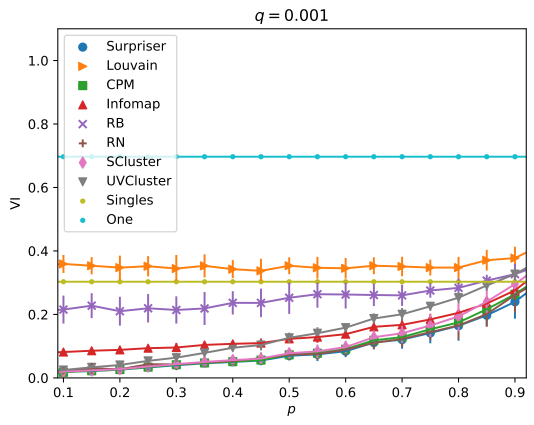
Table 4 – The VI found by each algorithm in the following range of parameters:$\mu \leq 0.5$,$R \leq 50\%$,$q \leq 0.015$ and $p \leq 0.50$. In bold are the best performances in each benchmark. The uncertainties shown around the average are standard deviations calculated for values bigger and smaller than the average, respectively. Visual representation in figure 33-a).

Algorithms	LFR	RC	New
Surpriser	$0.399241^{0.135472}_{0.203364}$	$0.528923^{0.216225}_{0.251377}$	$\mathbf{0.264694^{0.134667}_{0.109675}}$
Louvain	$0.508040^{0.234492}_{0.285229}$	$0.511509^{0.122748}_{0.248634}$	$0.556637^{0.112285}_{0.174050}$
CPM	$0.439201^{0.193429}_{0.229669}$	$0.596997^{0.131148}_{0.381965}$	$0.288294^{0.149499}_{0.120155}$
Infomap	$\mathbf{0.331695^{0.202853}_{0.269728}}$	$0.433725^{0.309244}_{0.101209}$	$0.370177^{0.206464}_{0.142379}$
RB	$0.439068^{0.234718}_{0.280001}$	$0.627685^{0.081953}_{0.353243}$	$0.410320^{0.099861}_{0.139858}$
RN	$0.394479^{0.130647}_{0.365198}$	$\mathbf{0.336153^{0.035244}_{0.007830}}$	$0.514620^{0.106431}_{0.308541}$
SCluster	$0.506017^{0.125927}_{0.141770}$	$0.608308^{0.114039}_{0.306444}$	$0.295673^{0.134964}_{0.120140}$
UVCluster	$0.514003^{0.093584}_{0.108356}$	$0.687288^{0.053606}_{0.106389}$	$0.349281^{0.104142}_{0.123330}$

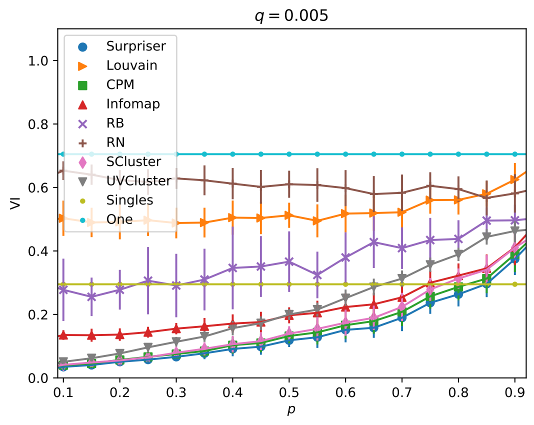
Table 5 – The VI found by each algorithm in the following range of parameters:$\mu \geq 0.6$, $R \geq 60\%$, $q \leq 0.030$ and $p \geq 0.55$. In bold are the best performances in each benchmark. The uncertainties shown around the average are standard deviations calculated for values bigger and smaller than the average, respectively. Replicated in figure 33-b).

Algorithms	LFR	RC	New
Surpriser	$0.187178^{0.312471}_{0.159115}$	$\mathbf{0.149206^{0.449037}_{0.126935}}$	$\mathbf{0.160425^{0.171839}_{0.094950}}$
Louvain	$0.249634^{0.417642}_{0.199084}$	$0.237330^{0.356396}_{0.096326}$	$0.517615^{0.108132}_{0.154250}$
CPM	$0.205227^{0.359638}_{0.177112}$	$0.167895^{0.488192}_{0.143230}$	$0.175922^{0.188736}_{0.103893}$
Infomap	$\mathbf{0.148336^{0.351064}_{0.144507}}$	$0.176089^{0.261329}_{0.151673}$	$0.277221^{0.242553}_{0.125291}$
RB	$0.197696^{0.401103}_{0.186573}$	$0.258064^{0.430485}_{0.128299}$	$0.336432^{0.135647}_{0.126166}$
RN	$0.176394^{0.346608}_{0.172141}$	$0.190905^{0.143340}_{0.185231}$	$0.516465^{0.110914}_{0.391555}$
SCluster	$0.276296^{0.267934}_{0.197138}$	$0.182048^{0.440071}_{0.150730}$	$0.178715^{0.184581}_{0.106975}$
UVCluster	$0.324372^{0.215433}_{0.170312}$	$0.231394^{0.388237}_{0.187417}$	$0.217415^{0.176365}_{0.128407}$

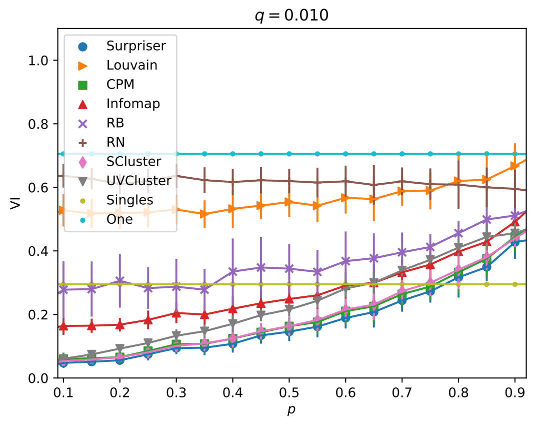
Table 6 – The average VI using every network. In bold are the best performances in each benchmark. The uncertainties shown around the average are standard deviations calculated for values bigger and smaller than the average, respectively. Numbers from figure 33-c).



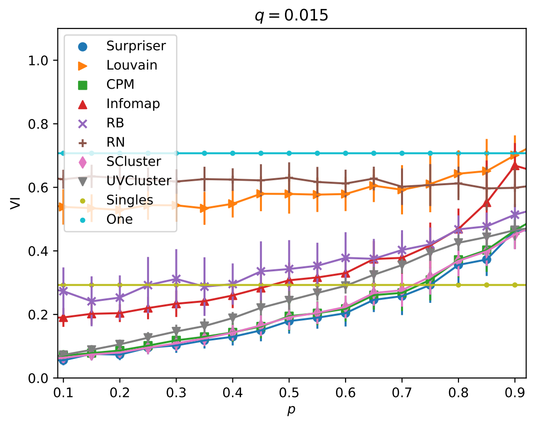
(a) q=0.001



(b) q=0.005



(c) q=0.010



(d) q=0.015

Figure 31 – Performance in our benchmark. CPM and Surpriser show the best results, while the Modularity based methods perform poorly in this benchmark. The results from Louvain are worse than the singles partition in the entire range of parameters. Of the three benchmarks, the algorithms perform worse in ours, which suggest that it's actually the toughest of the three.

benchmarks. In a) we consider only the cases where there's a clear definition of community structure, $\mu \leq 0.50$ and $q \leq 0.015$, and cases where the degradation is smaller than 50%, so $R \leq 50\%$ and $p \leq 0.50$, while in b) we have the complementary interval, $\mu \geq 0.60$, $R \geq 60\%$, $q \leq 0.030$ and $p \geq 0.55$, finally in c) we have the performance over all networks.

In figure 33 a), the algorithms that solve LFR superbly, Infomap, RN, and RB, have a very poor performance in the other benchmarks. Louvain, on the other hand, has an inferior performance in LFR than those three while being bad at RC and new. SCLuster and UVCluster form a second echelon, being somewhat consistent in the three benchmarks but having inferior performances than Surpriser and CPM, the clear winners when you consider consistency and performance together.

It's not particularly useful to rank the algorithms based on their performances

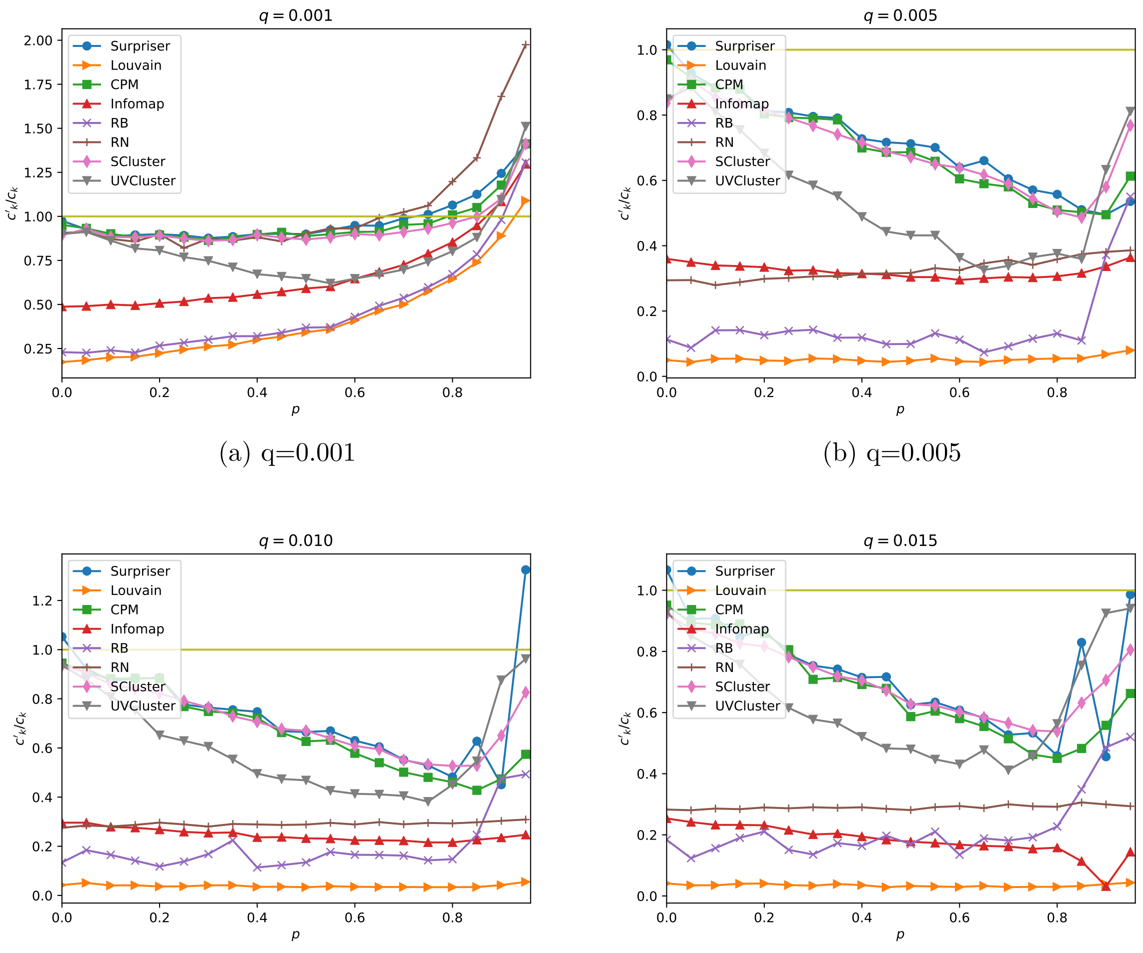
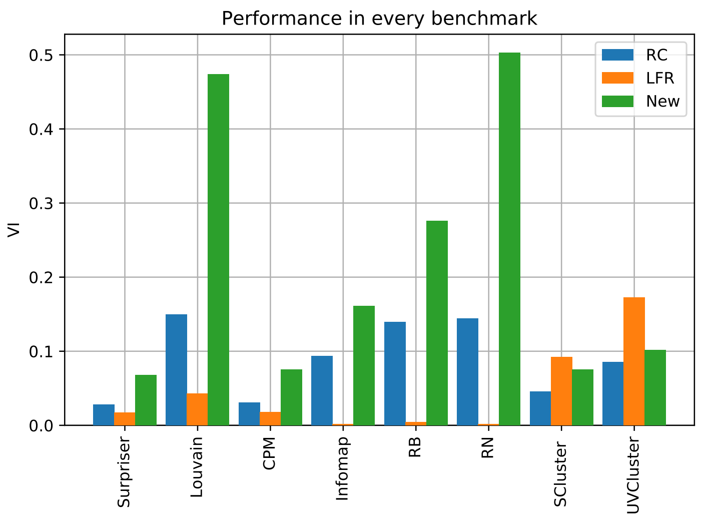


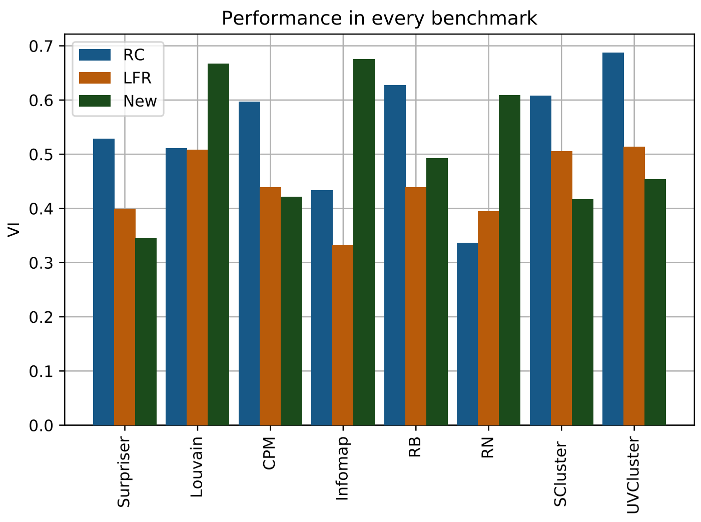
Figure 32 – Ratio between the average number of communities found by the algorithms, c'_k, with the average number of communities seeded by the benchmarks c_k. A ratio of 1, where both c'_k and c_k match perfectly, is represented by the yellow lines.

shown in figure 33 b) as we have extremely degraded networks and networks with a weakly defined comunity structure by default. This figure is most useful as a tie-break for the best performing algorithms in figure 33 a). From the best performing algorithms in the LFR benchmark (Infomap, RB and RN) Infomap is the best, followed by RB and RN. In the case of the RC benchmark, both RB and Infomap have a misleading performance, as those algorithms tend to stabilize in a structure were every node is in the same community, which does yield a smaller value of VI (around 0.3) but hardly represents the real structure of the system. The difference between Surpriser and CPM is more prominent here, with Surpriser outperforming CPM in every benchmark.

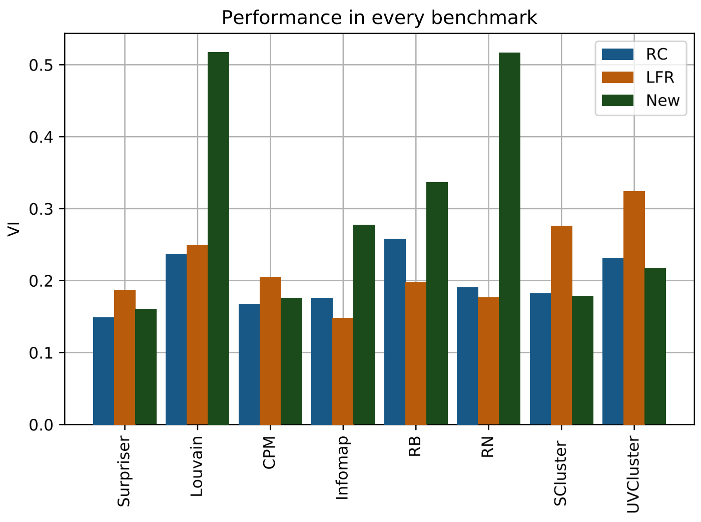
Figure 33 c) is the combination of all the results, here Surpriser is once again the best, followed by CPM and Infomap. One could argue that if we remove our new benchmark, Surpriser has a worse overall performance than Infomap and that it could



(a) $\mu \leq 0.5$,$R \leq 50\%$,$q \leq 0.015$ and $p \leq 0.50$.



(b) $\mu \geq 0.6$, $R \geq 60\%$, $q \leq 0.030$ and $p \geq 0.50$.



(c) Every network.

Figure 33 – Performances in every benchmark, darker colours represent regions with weaker community structures.

	New	LFR	RC
New	X	0.65 ± 0.02	0.43 ± 0.03
LFR	X	X	0.70 ± 0.02

Table 7 – Variation of Information between the initial partitions of the three different benchmarks. We see that RC and New are much more alike with a VI of 0.43 ± 0.03 between them.

suggest a bias from our benchmark towards our algorithm. This argument has a couple of problems. First, the results in figure 33 c) are tainted in much the same way as in figure 33 b), because we are considering regions with weak communities structures and vastly degraded networks, no algorithm has an optimal performance in such regions. The strongest and most representative results are those from figure 33 a). If we remove our benchmark in that scenario Surpriser vastly outperforms Infomap, and has an ever so slightly lead on CPM (in the order of 0.01).

Another point that must be raised is that from the methods that use the Surprise as a criteria to choose among different partitions (RB, CPM, UVCluster and SCluster), we see that only CPM is comparable to Surpriser and that a heads-on approach, that seeks to maximize the Surprise directly is preferable over the indirect methods. In general, the methods that use the Surprise function are the most consistent among the different benchmarks, with the exception of RB that is bound by the resolution limit of Modularity. In the last case, utilizing the Surprise as the basis to choose among different values of γ_{RB} does improve the results of Louvain significantly.

Our results show a correlation between the performances in the RC and New benchmark where algorithms that perform greatly in one have the same behavior in the other. Conversely, the best performing algorithms in the LFR benchmark fail to solve the community structure from the other benchmarks. To study why this is the case we calculated the VI between the initial partitions from each network and each benchmark. In order to make this work, we created 500 additional RC networks[2] with 1000 nodes divided in the same 16 cliques with the same Pielou's index of 0.75, the results are in table 7. The LFR benchmark is the most different among them, with a VI of 0.70 ± 0.02 in relation to the RC benchmark and a VI of 0.65 ± 0.02 in relation to our benchmark. RC and new have, on the other hand, a VI of 0.43 ± 0.03. This shows why some algorithms perform superbly in the LFR benchmark but poorly in the other ones, as the LFR is very different from RC and New.

[2] We can only measure the VI between two clusters if the number of elements is equal. The original RC networks had 512 nodes, whereas the New and LFR networks had 1000.

4.3 Degeneracy Problem

When different partitions yield a similar, close to the maximum value of the quality function we say that those solutions are degenerated. In a more formal way, it's the problem of not having a clear global maximum, and instead having many different local maxima very close together.

We will analyse this problem in the context of Modularity and Surprise. To do that we will go back to the network of figure 15 (reproduced again bellow), we've already shown that two different partitions attain the maximum value of Surprise, but in order to study the degeneracy problem further we'll need lots of different partitions that have a wide range of Surprise and Modularity values. We obtain those partitions by performing an annealing in the quality functions. Here we want a more throughout sample of the partition space, we obtain that by stopping the annealing process at different steps. In the next examples we used 1317 different partitions from the network in figure 34.

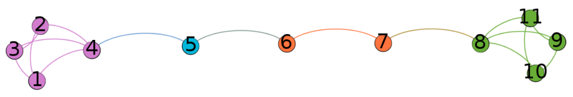


Figure 34 – We go back to the network shown in 15 in order to study the degeneracy problem.

The goal is to analyse the difference between the partitions in partition space and the relationship between their values of Surprise and Modularity. Comparing different partitions is easy, we will use the VI for that. In particular, we created a distance matrix d_{ij}, where every element is the VI between partitions i and j. To make the visualization of this matrix easier we made a linear embedding of all the partitions. A vector $\vec{r}_i = (x_i, y_i)$ is associated to each partition in such a way that the euclidean distance between two different vectors $\vec{r}_i$ and $\vec{r}_j$, each representing a different partition, is as close as possible to the VI between partitions i and j.

The embedding was created using a steepest descent algorithm that walks over the

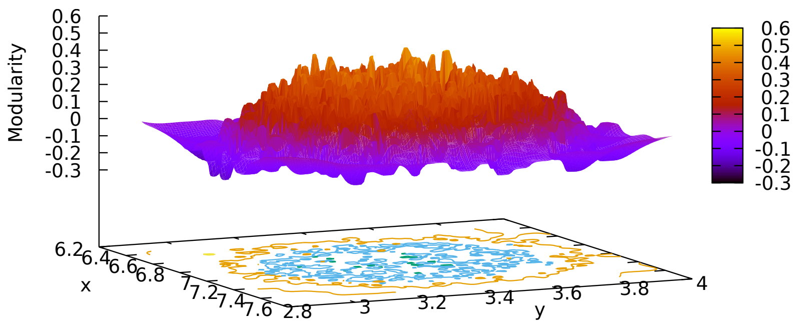
$\vec{r}_i$ parameter space searching for the minimum of χ^2 defined as:

$$\chi^2 = \sum_{i=1}^{N} \sum_{j>i|d_{ij}<d_{\text{lim}}} d_{ij}^{\gamma} \left(d_{ij} - \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \right)^2 , \tag{4.3}$$

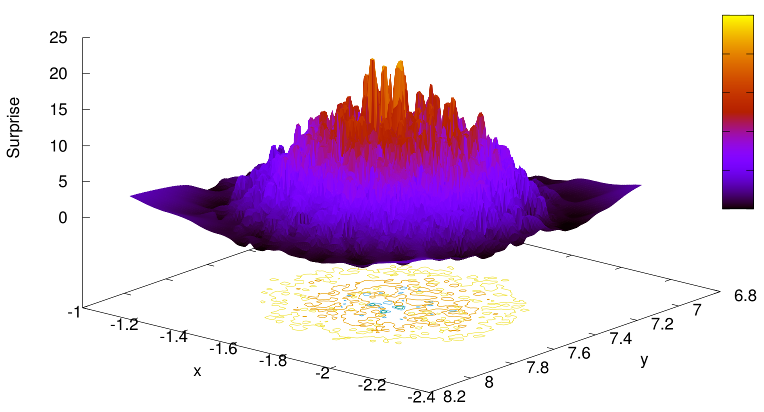
where the first sum runs over all partitions while the second sum runs over the partitions whose distance to the i partition is less than some predetermined value d_{lim}. The smaller the value of χ^2 the closer the euclidean distances are to the VI between partitions i and j (d_{ij}). To minimize χ^2 we evaluate the gradient of χ^2 in parameter space and then update the values of all x_i and y_i in the direction opposite of this gradient, so going towards smaller χ^2. Figures 35 a) and b) show the embedding, note that the z-axis represents the value of Modularity and Surprise respectively.

If we compare figures 35 a) and b) 35 we see that in Modularity's case the degeneracy problem is accentuated, as there's a plateau region where many different solutions lie really close to the global maximum in the z-axis, but are far apart in the xy-plane. Because the distance in the xy-plane mimics the VI between two partitions, this plateau region contain partitions that are close to the global maximum but that are very different from each other. Surprise, on the other hand, has a very sharp peak around the optimal value. As the points near the maximum are closer to each other, this means that the close-to-optimal partitions are also more similar.

This does not mean that Surprise does not suffer from a degeneracy problem at all, what it does mean is that partitions with a value close to the maximum are similar with each other so that even if we don't find the global maximum, the partitions that are nearby that point are representative of the networks true community structure. Both Surprise and Modularity suffer from the degeneracy problem in the sense that there's no guarantee that the global maximum is unique, however partitions close to the maximum of Surprise are similar with each other, unlike Modularity, where those partitions can represent very different structures (GOOD; MONTJOYE; CLAUSET, 2010)).



(a) Modularity



(b) Surprise

Figure 35 – Linear embedding of 1317 different partitions from the network in figure 34. Here the distances in the xy-plane represent the VI between the partitions in partition space, while the z coordinate gives us the value of Modularity (a) and Surprise (b) associated with each partition. In a) we see a plateau region, where lots of partitions are close to the global maximum of Modularity but far apart in the xy-plane. This means that the Modularity landscape is extremely degenerated, as partitions with a close to maximum value of Modularity can be structurally very different. In b), however, we see a peak around the maximum values of Surprise, which means that, unlike Modularity, the high scoring partitions in a Surprise landscape are more similar with each other and that a near-optimal partition is more representative of the networks true community structure.

Conclusion

Throughout the text we talked about the three main challenges for community detection methods. The first being that the number of possible partitions grows faster than exponentially for large networks (with the $N-$ th Bell number). This makes it computationally impossible to enumerate and to analyse every single partition of a network. Second, while the intuitive idea behind communities is well understood, there's no consensus formal definition. The discussion then resumes to which of the many heuristic functions would best represent the community structure of a network. The third and final challenge is that, in most cases, there's no information about the community structure of real world networks *a priori*. To circumvent this problem, various different benchmarks were proposed. Those benchmarks use our intuitive ideas about the concept of communities to create their networks with seeded community structure and make it easy to compare the differences between competing algorithms.

This work dealt with the last two challenges (as the first is inherent to the definition of a partition, so we just have to cope with it). In regard to the quality functions, we proposed a novel algorithm (Surpriser) based on the Surprise function. We compared the Surprise directly with Modularity, the most used function in the literature, and showed that the degeneracy problem is much less pervasive in a Surprise landscape. While both functions have degenerated solutions, *i.e.*, different partitions yield values very close or equal to the global maximum of the function, the near-optimal partitions found by Surprise are structurally much more alike than the ones found by Modularity. This bodes well for Surprise maximization algorithms because, even if different methods find slightly different partitions, if those partitions are close to the global maximum, those partitions will be similar with each other. Alongside that, Surprise does not seem to be affected by the resolution limit that haunts Modularity (more on that shortly).

With respect to the benchmarking problem, we proposed a novel benchmark. Our benchmark starts with sets of cliques, whose sizes are set according to a power law, and sets of isolated nodes. On one hand we have the strongest definition of a community, a clique, and on the other we have the weakest possible one, a single node. Alongside those intuitive ideas we created two parameters q, that controls how fuzzy the community structure is, and p that controls how degraded the communities are. While our benchmark networks are closer to the ones from the RC benchmark, our results show that this novel benchmark is the hardest of the three for the algorithms to solve.

Our Surpriser algorithm was tested against 7 other algorithms from the literature in three different benchmarks: LFR, RC and our own. As real word networks come in various shapes and sizes (figure 3), the best community detection algorithm should be able to perform reasonably in the widest range of conditions. Consistency is the name of the game. We seek an algorithm that is able to solve every benchmark. Our results, however, show that most algorithms tend to perform greatly in one benchmark but poorly in the other ones. This is the case of Infomap, RB and RN, that solve the community structure for the LFR networks but fail in the other ones, and of SCluster and UVCluster, that perform better in our own and RC networks but fail in LFR networks. Of note, Infomap has the best overall performance in LFR networks, with the caveat that as the networks community structure gets fuzzier, Infomap has a tendency to merge every node in one community.

Two Modularity based methods were tested: Louvain, with a fixed resolution parameter, and RB, with a varying resolution parameter. While allowing the resolution parameter to vary does improve the results, it's not enough to circumvent the known resolution limit of Modularity. Whenever the distribution of community sizes is wide (as in a system that has very large communities but also very small ones) Modularity based algorithms fail to find a balance between merging small clusters into larger ones and splitting big clusters into smaller ones. This is clearly reflected in the performance of both Louvain and RB. They perform best in the LFR networks, which have a very high Pielou's Index of 0.95. In those networks, the range of community sizes goes from as few as 7 nodes to as much as 53, because this range is so small RB is able to find a resolution parameter that perfectly fits those networks and thus has one of the top performances in this benchmark.

Conversely, our own and RC benchmarks have community sizes that range from one node to more than two hundred. Because of this unevenness, reflected in the smaller Pielou's indexes of 0.86 and 0.75 respectively, Modularity completely fails to find the community structure of those networks. It will wrongly merge smaller communities in to larger ones, in much the same way as in the ring of cliques network. Depending on the network size, Modularity stops to identify the smallest communities joining them into bigger ones.

Only two algorithms perform consistently in every benchmark: our own Surpriser and CPM. Both seek to maximize the Surprise function, while one does so directly (Surpriser) and the other indirectly (CPM). The heads-on approach slightly outperforms the indirect one in every case, with a difference in order of 0.01 in the VI. This difference gets bigger for fuzzier networks, as both algorithms perform admirably well for less degenerated, better defined community structures.

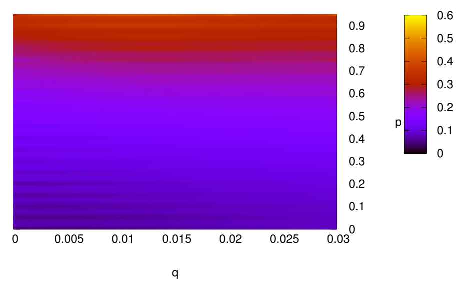
A Heat Maps

Our benchmark has two main degradation parameters: p and q. The former represents the probability that a link is removed from a clique, while the later represents the probability that a link is created between two nodes (from different cliques). We can represent the Variation of Information as a function of both p and q by constructing a heat map. Figure 36 from a) to d) show the top performers, while figure 37 from a) to d) show the worst performing algorithms.

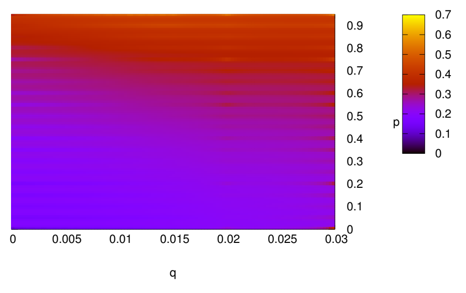
Once again we have Surpriser and CPM as the front runners, followed by SCluster,Infomap and UVCluster. RN,RB and Louvain perform miserably in this benchmark. The Modularity based methods suffer specially, with Louvain failing to solve the community structure in the whole range of parameters, and RB being able to solve the communities only for small values of both p and q.

(a) Surpriser

(b) CPM

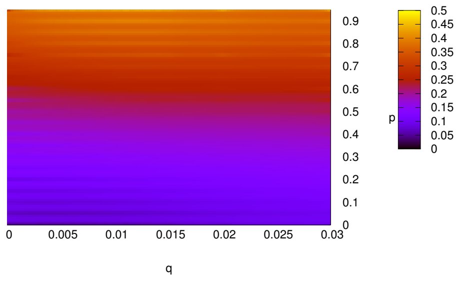


(c) SCLuster

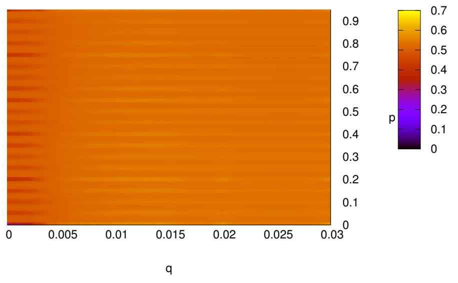


(d) Infomap

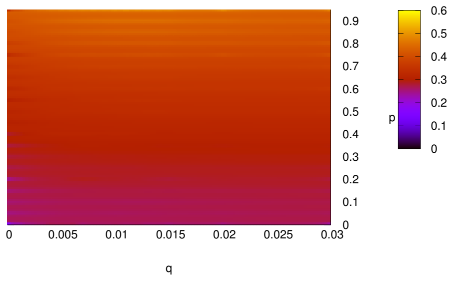
Figure 36 – Best performing algorithms in our benchmark.



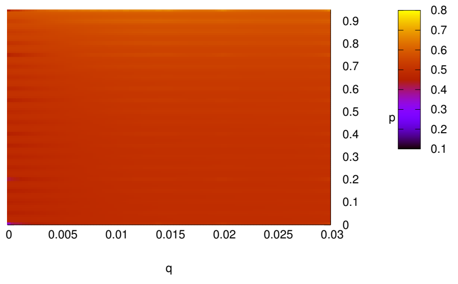
(a) UVCluster



(b) RN



(c) RB



(d) Louvain

Figure 37 – Worst performing algorithms in our benchmark.

Bibliography

ALDECOA, R.; MARÍN, I. Jerarca: Efficient analysis of complex networks using hierarchical clustering. *PloS one*, Public Library of Science, v. 5, n. 7, p. e11585, 2010. Citado 4 vezes nas páginas 27, 28, 41, and 57.

ALDECOA, R.; MARÍN, I. Deciphering network community structure by surprise. *PloS one*, Public Library of Science, v. 6, n. 9, p. e24195, 2011. Citado 6 vezes nas páginas 28, 29, 41, 51, 55, and 57.

ALDECOA, R.; MARÍN, I. Surprise maximization reveals the community structure of complex networks. *Scientific reports*, Nature Publishing Group, v. 3, p. 1060, 2013. Citado na página 44.

ALDECOA, R.; MARÍN, I. Surpriseme: an integrated tool for network community structure characterization using surprise maximization. *Bioinformatics*, Oxford University Press, v. 30, n. 7, p. 1041–1042, 2013. Citado 2 vezes nas páginas 41 and 68.

ARNAU, V.; MARS, S.; MARÍN, I. Iterative cluster analysis of protein interaction data. *Bioinformatics*, Oxford University Press, v. 21, n. 3, p. 364–378, 2004. Citado 2 vezes nas páginas 27 and 28.

AZUAJE, F. et al. Analysis of a gene co-expression network establishes robust association between col5a2 and ischemic heart disease. *BMC medical genomics*, BioMed Central, v. 6, n. 1, p. 13, 2013. Citado na página 28.

BARABÁSI, A.-L.; BONABEAU, E. Scale-free networks. *Scientific american*, JSTOR, v. 288, n. 5, p. 60–69, 2003. Citado na página 34.

BARABÁSI, A.-L.; GULBAHCE, N.; LOSCALZO, J. Network medicine: a network-based approach to human disease. *Nature reviews genetics*, Nature Publishing Group, v. 12, n. 1, p. 56, 2011. Citado 3 vezes nas páginas 27, 28, and 34.

BARABASI, A.-L.; OLTVAI, Z. N. Network biology: understanding the cell's functional organization. *Nature reviews genetics*, Nature Publishing Group, v. 5, n. 2, p. 101–113, 2004. Citado 2 vezes nas páginas 27 and 28.

BARRAT, A. et al. The architecture of complex weighted networks. *Proceedings of the national academy of sciences*, National Acad Sciences, v. 101, n. 11, p. 3747–3752, 2004. Citado 2 vezes nas páginas 27 and 34.

BLONDEL, V. D. et al. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, IOP Publishing, v. 2008, n. 10, p. P10008, 2008. Citado na página 41.

BROIDO, A. D.; CLAUSET, A. Scale-free networks are rare. *Nature communications*, Nature Publishing Group, v. 10, n. 1, p. 1017, 2019. Citado na página 34.

CALDARELLI, G. *Large scale structure and dynamics of complex networks: from information technology to finance and natural science.* [S.l.]: World Scientific, 2007. v. 2. Citado na página 34.

CARRON, P. M.; KENNA, R. Universal properties of mythological networks. *EPL (Europhysics Letters)*, IOP Publishing, v. 99, n. 2, p. 28002, 2012. Citado na página 25.

CLAUSET, A.; MOORE, C.; NEWMAN, M. E. Hierarchical structure and the prediction of missing links in networks. *Nature*, Nature Publishing Group, v. 453, n. 7191, p. 98, 2008. Citado na página 45.

EULER, L. Solutio problematis ad geometriam situs pertinentis. *Commentarii academiae scientiarum Petropolitanae*, p. 128–140, 1741. Citado na página 32.

FORTUNATO, S. Community detection in graphs. *Physics reports*, Elsevier, v. 486, n. 3-5, p. 75–174, 2010. Citado na página 27.

FORTUNATO, S.; BARTHÉLEMY, M. Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, National Academy of Sciences, v. 104, n. 1, p. 36–41, 2007. ISSN 0027-8424. Disponível em: <http://www.pnas.org/content/104/1/36>. Citado 2 vezes nas páginas 28 and 42.

GAMERMANN, D.; TRIANA, J.; JAIME, R. A comprehensive statistical study of metabolic and protein-protein interaction network properties. *Physica A: Statistical Mechanics and its Applications*, Elsevier, p. 122204, 2019. Citado na página 34.

GONÇALVES, B. et al. Exploring team passing networks and player movement dynamics in youth association football. *PloS one*, Public Library of Science, v. 12, n. 1, p. e0171156, 2017. Citado na página 25.

GOOD, B. H.; MONTJOYE, Y.-A. de; CLAUSET, A. Performance of modularity maximization in practical contexts. *Physical Review E*, APS, v. 81, n. 4, p. 046106, 2010. Citado 5 vezes nas páginas 28, 42, 43, 44, and 84.

GREGORY, S. Finding overlapping communities in networks by label propagation. *New Journal of Physics*, IOP Publishing, v. 12, n. 10, p. 103018, 2010. Citado na página 61.

GUIMERA, R.; AMARAL, L. A. N. Functional cartography of complex metabolic networks. *nature*, Nature Publishing Group, v. 433, n. 7028, p. 895, 2005. Citado na página 28.

GUIMERA, R. et al. The worldwide air transportation network: Anomalous centrality, community structure, and cities' global roles. *Proceedings of the National Academy of Sciences*, National Acad Sciences, v. 102, n. 22, p. 7794–7799, 2005. Citado na página 25.

HAGBERG, A.; SWART, P.; CHULT, D. S. *Exploring network structure, dynamics, and function using NetworkX*. [S.l.], 2008. Citado na página 41.

HARTWELL, L. H. et al. From molecular to modular cell biology. *Nature*, Nature Publishing Group, v. 402, n. 6761supp, p. C47, 1999. Citado na página 27.

HUFFMAN, D. A. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, IEEE, v. 40, n. 9, p. 1098–1101, 1952. Citado na página 47.

JONES, J. H.; HANDCOCK, M. S. An assessment of preferential attachment as a mechanism for human sexual network formation. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, The Royal Society, v. 270, n. 1520, p. 1123–1128, 2003. Citado na página 27.

JUNKER, B. H.; SCHREIBER, F. *Analysis of biological networks*. [S.l.]: John Wiley & Sons, 2011. v. 2. Citado na página 28.

KANEHISA, M.; GOTO, S. Kegg: kyoto encyclopedia of genes and genomes. *Nucleic acids research*, Oxford University Press, v. 28, n. 1, p. 27–30, 2000. Citado na página 25.

KING, A. D.; PRŽULJ, N.; JURISICA, I. Protein complex prediction via cost-based clustering. *Bioinformatics*, Oxford University Press, v. 20, n. 17, p. 3013–3020, 2004. Citado na página 41.

LANCICHINETTI, A.; FORTUNATO, S.; RADICCHI, F. Benchmark graphs for testing community detection algorithms. *Phys. Rev. E.*, v. 78, p. 046110, 2008. Citado 2 vezes nas páginas 29 and 55.

LOVÁSZ, L. *Combinatorial problems and exercises*. [S.l.]: American Mathematical Soc., 2007. v. 361. Citado na página 37.

MEILĂ, M. Comparing clusterings by the variation of information. In: *Learning theory and kernel machines*. [S.l.]: Springer, 2003. p. 173–187. Citado na página 37.

NEWMAN, M. E. Modularity and community structure in networks. *Proceedings of the national academy of sciences*, National Acad Sciences, v. 103, n. 23, p. 8577–8582, 2006. Citado na página 42.

NICOLINI, C.; BIFONE, A. Modular structure of brain functional networks: breaking the resolution limit by surprise. *Scientific reports*, Nature Publishing Group, v. 6, p. 19250, 2016. Citado 5 vezes nas páginas 27, 28, 29, 43, and 44.

NICOLINI, C.; BORDIER, C.; BIFONE, A. Community detection in weighted brain connectivity networks beyond the resolution limit. *Neuroimage*, Elsevier, v. 146, p. 28–39, 2017. Citado 3 vezes nas páginas 27, 28, and 29.

ORMAN, G.; LABATUT, V. A comparison of community detection algorithms on artificial networks. Citado na página 61.

PAPIN, J. A.; REED, J. L.; PALSSON, B. O. Hierarchical thinking in network biology: the unbiased modularization of biochemical networks. *Trends in biochemical sciences*, Elsevier, v. 29, n. 12, p. 641–647, 2004. Citado na página 28.

PENA, J. L.; TOUCHETTE, H. A network theory analysis of football strategies. *arXiv preprint arXiv:1206.6904*, 2012. Citado na página 25.

PIELOU, E. C. The measurement of diversity in different types of biological collections. *Journal of theoretical biology*, Elsevier, v. 13, p. 131–144, 1966. Citado na página 37.

RAMOS, M. et al. How does public opinion become extreme? *Scientific reports*, Nature Publishing Group, v. 5, 2015. Citado na página 27.

REICHARDT, J.; BORNHOLDT, S. Statistical mechanics of community detection. *Physical Review E*, APS, v. 74, n. 1, p. 016110, 2006. Citado 2 vezes nas páginas 41 and 48.

RIBEIRO, M. A. et al. The complex social network of the lord of rings. *Revista Brasileira de Ensino de Física*, SciELO Brasil, v. 38, n. 1, 2016. Citado na página 25.

RIVES, A. W.; GALITSKI, T. Modular organization of cellular networks. *Proceedings of the National Academy of Sciences*, National Acad Sciences, v. 100, n. 3, p. 1128–1133, 2003. Citado 2 vezes nas páginas 27 and 28.

RONHOVDE, P.; NUSSINOV, Z. Local resolution-limit-free potts model for community detection. *Physical Review E*, APS, v. 81, n. 4, p. 046114, 2010. Citado 2 vezes nas páginas 41 and 49.

ROSVALL, M.; BERGSTROM, C. T. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, National Acad Sciences, v. 105, n. 4, p. 1118–1123, 2008. Citado 4 vezes nas páginas 27, 41, 46, and 48.

SEGAL, E. et al. Module networks: identifying regulatory modules and their condition-specific regulators from gene expression data. *Nature genetics*, Nature Publishing Group, v. 34, n. 2, p. 166, 2003. Citado na página 28.

SOKAL, R. R. A statistical method for evaluating systematic relationship. *University of Kansas science bulletin*, v. 28, p. 1409–1438, 1958. Citado na página 46.

SPIRIN, V.; MIRNY, L. A. Protein complexes and functional modules in molecular networks. *Proceedings of the National Academy of Sciences*, National Academy of Sciences, v. 100, n. 21, p. 12123–12128, 2003. ISSN 0027-8424. Disponível em: <http://www.pnas.org/content/100/21/12123>. Citado na página 28.

TRAAG, V. A.; ALDECOA, R.; DELVENNE, J.-C. Detecting communities using asymptotical surprise. *Physical review e*, APS, v. 92, n. 2, p. 022816, 2015. Citado 2 vezes nas páginas 29 and 51.

TRAAG, V. A.; DOOREN, P. V.; NESTEROV, Y. Narrow scope for resolution-limit-free community detection. *Physical Review E*, APS, v. 84, n. 1, p. 016114, 2011. Citado 4 vezes nas páginas 41, 48, 55, and 61.

WATTS, D. J. *Small worlds: the dynamics of networks between order and randomness*. [S.l.]: Princeton university press, 2004. v. 9. Citado na página 29.

YANG, Z.; ALGESHEIMER, R.; TESSONE, C. J. A comparative analysis of community detection algorithms on artificial networks. *Scientific reports*, Nature Publishing Group, v. 6, p. 30750, 2016. Citado na página 55.

ZANZONI, A.; SOLER-LÓPEZ, M.; ALOY, P. A network medicine approach to human disease. *FEBS letters*, Wiley Online Library, v. 583, n. 11, p. 1759–1765, 2009. Citado na página 28.

ZHU, J. et al. Integrating large-scale functional genomic data to dissect the complexity of yeast regulatory networks. *Nature genetics*, Nature Publishing Group, v. 40, n. 7, p. 854, 2008. Citado na página 28.