Community Detection in Multi-Layer Networks

Valentin Dalibard
St John's College

Supervisor: Eiko Yoneki

A dissertation submitted to the University of Cambridge in partial fulfilment of the requirements for the degree of Master of Engineering in Advanced Computer Science

University of Cambridge
Computer Laboratory
William Gates Building
15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom

Email: vd241@cam.ac.uk

June 14, 2012

Abstract

Recent years have seen a surge of interest in community detection and a large range of algorithms have been designed and applied to this specific task. However, the vast majority of the work done in the area has been concerned with simple weighted graphs. This results in some of the real world systems being oversimplified to fit this model. In this project, I design a new community detection algorithm capable of detecting communities in multi-layer networks. Multi-layer networks are constituted of a finite set of graphs, called layers, which share the same nodes but differ in their edges. I construct a modeling approach whereby I try to approximate the network using communities. The results show that this approach can reveal multi-layer communities that would not be found using standard community detection tools.

Declaration

I, Valentin Dalibard of St John's College, being a candidate for Part III of the Computer Science Tripos, hereby declare that this dissertation and the work described in it are my own work, unaided except as may be specified below, and that the dissertation does not contain material that has already been used to any substantial extent for a comparable purpose.

Signed

Date June 14, 2012

Word Count: 10 747

Contents

Chapter 1

Introduction

The aim of my project was to extend community detection mechanisms to work on multi-layer networks. In this chapter, I explain what community detection and multi-layer networks are. I then go over why implementing community detection on multi-layer networks is an interesting and worthwhile effort and describe some of the challenges involved.

1.1 Motivation

Fields such as sociology, biology or computer science often use graphs to represent real world systems. Most often, these graphs are inhomogeneous, with high concentration of edges within special groups of vertices and low concentration between these groups. The task of detecting these structures is called *community detection* or *clustering*. Community detection has been of high interest in recent years [1], and many algorithms, based over a variety of heuristics have been proposed. However, the vast majority of this work has been concerned with simple weighted graphs. The aim of this project is to design community detection mechanisms that can be applied on multi-layer networks.

A multi-layer network, is a network made of multiple graphs, called layers, which share the same set of vertices, but differ in their edges. Most real world systems can gain in accuracy by being represented as a multi-layer network. This is due to the fact that, often, edges are a function of one or more variables, and attempting to represent them as a single scalar leads to a loss in accuracy.

The initial motivation for this project comes from Pocket Switched Networks (PSNs), in which a multitude of devices carried by people are dynamically networked. Routing packets in a PSN is difficult as human mobility is often unpredictable. However, recent work has shown that detecting and using communities for routing can bring significant improvements [2]. This was done by applying a standard community detection algorithm to a contact graph based on the number of contacts and contact durations between devices. The results were then used by forwarding packets to devices in the same community as the recipient. However, this technique makes no use of the existence of different types of relationships between people. For example, a person's contact graph will be very different at 10am and 8pm. Therefore representing this contact network as a multi-layer graph, each corresponding to a time slice, may lead to more

accurate routing techniques.

Furthermore, there has been an astonishing increase in the use of graph structured information over the last few years. This has lead to a trend in research to use the underlying graph properties of networks as a basis for the construction of high performance architectures. Specialized graph processing platforms such as Google's Pregel [3], and graph databases such as Microsoft's Trinity [4] are good examples of this phenomenon. There has been a need for tools to understand the properties of graph structured data, such as densely connected clusters, and how they can be taken advantage of. This project aims to be a step in the direction of how to construct those tools.

1.2 Challenges

In attempting this project, I faced a number of challenges. First I had to survey the literature on community detection which is surprisingly vast. The recent status of community detection as a "hot topic" has lead to a surge of publications in the area, and a huge variety of of different methods have been proposed by an extremely interdisciplinary community, including physicist, computer scientists, mathematicians, biologists engineers and social scientists. To quote one of the most recent and well reviewed survey of the field: "it appears that the field has grown in a rather chaotic way, without a precise direction or guidelines" [1].

Using this knowledge, I had to decide on which clustering mechanism I was going to extend to multi-layer networks. As it turns out, I found none of the approaches that had been proposed was well suited enough to multiple layers. I therefore created a new community detection mechanism by merging together different aspects of previously published methods.

1.3 Overview of the project

The aim of the project was to construct a community detection mechanism that would work on multiple layers. The project was a success. I built a new community detection model that could be applied on multiple layers. I then extended some of the most used methods for non-overlapping community detection to overlapping community detection in order to detect those communities under my model and found a way to ensure the statistical significance of the communities produced. Finally, I applied community detection to real world data from pocket switched networks and was able to analyze how different communities exhibited different behaviors.

Chapter 2

Background

In this chapter, I summarize the underlying community detection problems necessary to understand the approach used in the implementation. I start by formalizing the problem of community detection, and give an example of one of the most widely used detection mechanism: modularity optimization. I then describe some of the different features that can be required from a community detection mechanism.

2.1 Basics definitions

A graph G is a pair of set (V, E), where V is a set of vertices and E is a set of edges, where each edge is an unordered pair of vertices. A multi-layer network is a finite set of graphs $\{G_1, G_2, \ldots, G_L\}$ such that $G_1 = (V, E_1), G_2 = (V, E_2), \ldots, G_L = (V, E_L)$. The whole topology of a graph with n vertices is entailed in its *adjacency matrix* $\mathbf{A}$, which is an $n \times n$ matrix whose element A_{ij} is 1 if there is an edge joining vertices i and j, and zero otherwise. A multi-layer network can thus be represented as a set of adjacency matrices $\{A^1, A^2, \ldots, A^L\}$ where $A^l_{ij} = 1$ if there is an edge joining i and j on layer l.

The first difficulty of community detection is to find a quantitative definition of community. Intuitively, a community is a group of vertices where more edges lie "inside" the community than edges linking vertices of the community with the rest of the graph, but no definition is universally accepted. A useful distinction is the one between *overlapping* and *non-overlapping* communities. In non-overlapping community detection, one tries to split the vertices into groups so that each vertex belongs to exactly one group, the final result is called a *partition*. In the case of overlapping community detection, vertices may belong to an arbitrary number of communities, which yields a *cover* of the graph.

Most algorithms are able to identify an number of meaningful communities, that does not mean the communities found are equally good. It is therefore often helpful to have a quantitative evaluation of the goodness of a community: a *quality function*. In the case of non-overlapping community detection, one often assigns a quality function to asses the goodness of the partitioning itself. In the next subsection, I give an example of a partitioning quality function: modularity.

2.2 An example: Modularity

The most popular quality function used for partitioning is the modularity of Newman and Girvan [5]. It is based on the idea that a random graph is not expected to hold community structures, so the presence of clusters is revealed by comparing the actual and expected density of edges in a subgraph. This expected density depends on the chosen *null model* of the graph. Modularity can be written as follows

$$Q = \frac{1}{2m} \sum_{ij} (A_{ij} - P_{ij}) \delta(C_i, C_j)$$

where the sum runs over all pairs of vertices. A is the adjacency matrix and P_{ij} the expected number of edges between vertices i and j in the null model. $\delta(C_i, C_j)$ yields 1 if i and j belong in the same community, and zero otherwise. Therefore, the quality function really only sums over pairs of vertices that belong in the same community. Finally, m is the total number of edges.

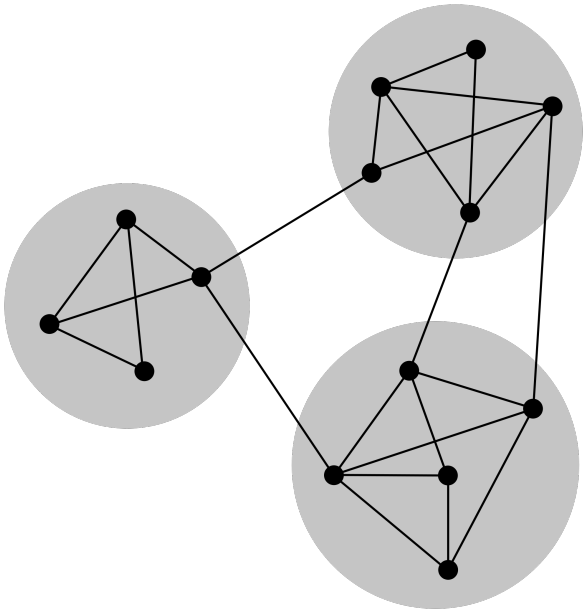


Figure 2.1: Modularity optimization applied on a small network.

Several possibilities exist for choosing the null model. In the simplest model, we only ensure that the graph keeps the same number of edges and set $P_{ij} = 2m/[n(n-1)]$. Often, the null model will be chosen so that the expected degree sequence matches the actual degree sequence of the graph. This leads to a model where $P_{ij} = k_i k_j / 2m$ where k_i and k_j are the degrees of vertices i and j respectively.

When using modularity to perform community detection, one has to devise a way to maximize the quality function for a graph. Unfortunately, finding this maximum is **NP**-complete. A number of techniques have been proposed to obtain high values of modularity in a reasonable amount of time, but performance can be a bottleneck, sometimes taking several hours to cluster 100 nodes.

2.3 Features and Difficulties of Community Detection

In this subsection, I will review the different problems associated with community detection and start to draw the picture for the algorithm I will describe in the implementation section.

2.3.1 Overlapping communities

Most of the literature on community detection is concerned with non-overlapping community detection, where each vertex is associated with a single cluster. However, it is well understood that in many cases, such as social networks, people are naturally characterized by multiple community memberships. For example, a person usually has a strong relationship with the social groups formed by his family, friends, colleagues. Kelly *et al.* showed that allowing groups to overlap is a natural and essential part of social network analysis [6].

Despite representing a fraction of the publications in the field, overlapping community detection has seen a diverse number of approaches including local expansion and optimization, fuzzy detection, dynamic algorithms and game theoretic approaches [7]. Reviewing them here would take too long. However, since I end up using a local expansion and optimization technique, I will mention one of the main difficulties with the approach. In this case, one has a quality function that rates how good of a community a subset of vertices is, and tries to optimize this function by starting with a few vertices and iteratively adding and removing some of them. The difficulty with this technique, is that as we try to find more communities, nothing stops us from finding communities very similar to the ones already produced. One of the best reviewed publication in the area, OSLOM [8], has this issue and solves it by merging similar communities after they have been detected.

Another feature of real world networks, is that they are often hierarchical, with some communities which include, or are included by, other communities. Hierarchies are an important part of real world networks and communities of all hierarchy level should be detected. However, in my opinion, constraining higher level communities to contain exact unions of lower level communities is too restrictive. For this reason, in my approach, I treat hierarchical communities as a special case of overlapping communities, therefore allowing higher level communities to contain only subsets of lower level communities.

2.3.2 Statistical Significance

One of the recent trend in community detection research has been to consider the statistical significance of communities. In statistics, statistical significance is used to show that a result is unlikely to have occurred by chance. When applied to community detection, this means it should be highly unlikely that the communities revealed by a detection algorithm could have been generated at random. Interest in this area was spawned by the realization that, in fact, a lot of the algorithms published, including modularity, would find clusters with high values of their quality function even in random networks.

Designing a measure of statistical significance for community detection is difficult. Ideally, one would like to take a measure of the edge densities in the communities, and asses the probability to to get such high measures at random. However, it is often not possible to do

this directly. The detection algorithm usually works in a way that maximizes some measure of the edge densities of the communities found. Therefore, it would be cherry picking to evaluate the statistics of a correlated measure also based on edge density. Two solutions can be used. In the first case, one designs a measure that is independent to the one being optimized. A good example of this is OSLOM [8], which I describe in section 5.1, and uses the statistics of the worse node in the community to evaluate significance. In the second case, the measure being optimized and the one being evaluated for significance are correlated, but the algorithm is applied to random networks to evaluate the statistics of the significance measure. This bootstrapping allows one to make the difference between a community that could happen at random and one that is a structural property of the data.

The use of statistical significance for community detection is in my opinion a step in the right direction. However, this research trend has lead to the publication of techniques that attempt to maximize the significance of the communities that are found. In other word, they use the statistical significance of a community as a quality function. This in my opinion is taking things the wrong way. While it is important that the communities produced are significant, this should be seen as a constrain on the detection mechanism rather than a goal.

2.3.3 Multi-Layer Networks

There has been very little study on how to handle multi-layer networks for community detection. Two general approaches have been proposed, in each case by generalizing a standard community detection mechanism to multiple layers. Each time, the result corresponds to some "average" communities over all layers. I believe this is the wrong approach. The reason why we use multi-layer networks is because the behavior of the nodes on the different graphs changes from one layer to another. Therefore, when designing a multi-layer community detection mechanism, one should embrace the fact that the communities should vary on the different layers. Of course, this does not mean the solution should be to apply community detection on each layer individually, some middle ground must be found.

2.3.4 Difficulty of Validation

Finally, one of the obstacles of community detection is the difficulty of validating both the algorithms and the data. In both cases, the problem comes from the blurriness of the definition of community.

Validating that a set of nodes in a graph is a community is difficult. One can show that it has statistical significance, but it may not correspond to the intuitive human notion of community. This is especially true in the context of social networks, where users have they own perspective of which communities they belong to.

For the same reason, it is hard to validate a community detection algorithm. One can use benchmarks whereby a synthetic network is generated with labeled communities. But, there are a large number of benchmarks and each corresponds to a different view of what a community should be. Furthermore, creating a synthetic networks that has the same properties as a real world graph is difficult. Using real world data to validate an algorithm also raises issues, some of the underlying conditions may not be visible, and labeling the communities

may be hard which goes back to the first point.

Chapter 3

Multi-Layer Community Detection

This chapter describes the design and implementation of community detection for multi-layer networks. I start with a discussion on which of the ideas offered by publications in community detection I decided to use. From there, I design a new community detection scheme and show how it is well suited to handle multiple layers. Finally, I explain how I implemented it in an efficient way that could also be decentralized.

3.1 Basic Ideas from Previous Work

From previous work I take the following directions:

- The community detection mechanism should allow for overlapping communities. This is especially necessary since we will want to handle multiple layers, which are likely to increase the diversity of communities a node may belong to.
- The communities found should be statistically significant. This means applying my algorithm on a random graph should return no communities.
- Detecting the communities should be done hierarchically. Real world networks are often highly hierarchical which can be used to greatly reduce the search space: if some low level communities have been found, we can search for higher level communities by merging them together. This approach has shown to offer high performance.

With this is mind, I design a single layer model for detecting communities in section 3.2. I then extend it to multiple layers in section 3.3 and describe the optimization techniques used in section 3.4.

3.2 A Single Layer Model

In this section I describe a single layer model that I will extend to multiple layers in the next section. I first explain the behavior of the model and define the optimization problem along

with some constrains. I then go over how the values of some of the variables used in the model are found.

3.2.1 Single Layer Detection

In this subsection, I explain the basic concepts of the scheme. They will be refined in the following subsections.

I decide to take a modeling approach, whereby I try to represent the graph as well as possible through the use of communities. I start by explain how the graph is represented with those communities and then explain what function is actually going to be optimized.

I create a model whereby each pair of nodes ij has an associated random variable E_{ij} representing the likelihood that i and j share an edge. The random variables E_{ij} have a Bernoulli distribution: they take value 1 with probability P_{ij} and 0 with probability $1 - P_{ij}$. I assume the variables are independent. On a high level, the aim of modeling is to adapt the values P_{ij} to fit the graph as well as possible subject to some restriction.

To construct this model, I associate a probability P_{C_k} with each community. I then model the probability of an edge between a pair ij of nodes as a succession of Bernoulli trials from each of the communities they share. For example, in figure 3.1, the three possible pairs of nodes belonging to both communities are each associated the probability $\frac{2}{3}$, as it corresponds to the probability of success after two Bernoulli trials with probability $\frac{1}{2}$ and $\frac{1}{3}$.

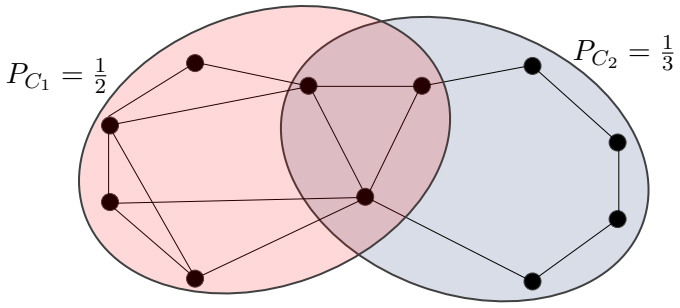


Figure 3.1: An example of the model. The left community is associated probability $P_{C_1} = \frac{1}{2}$ and the right community $P_{C_2} = \frac{1}{3}$.

In the general case, a pair of node may share an arbitrary number of communities leading to the equation:

$$P_{ij} = 1 - \prod_{\substack{C_k: \\ i,j \in C_k}} (1 - P_{C_k}) \,. \tag{3.1}$$

In order to set the probability associated with each community I enforce the constrain that for each community C_k, the expected number of edges within the community should be equal to the actual number of edges ind_{C_k}. This means we have:

$$\begin{aligned} \forall k \; ind_{C_k} &= \sum_{i,j \in C_k} \mathrm{E}\left(E_{ij}\right) \\ &= \sum_{i,j \in C_k} P_{ij}. \end{aligned} \tag{3.2}$$

For example, in figure 3.1, this constrain is enforced. There are 21 possible pairs of nodes in each community, three of which are modeled by the overlapping probability $\frac{2}{3}$. The expected number of edges is therefore $18 \times \frac{1}{2} + 3 \times \frac{2}{3} = 11$ for community 1 and $18 \times \frac{1}{3} + 3 \times \frac{2}{3} = 8$ for community 2. This corresponds to the actual number of edges.

Finally, in order to make sure that every pair of node is associated a probability, I systematically include all the nodes in a graph wide community which I name community 0.

3.2.1.1 Optimizing the Communities

Since the aim is to find a model that fits the graph, I want to minimize the difference between the expected values of each edge and their actual value. Recall that the value of edges are stored in the adjacency matrix $\mathbf{A}$. The goal of the optimization therefore is:

$$\min \sum_{ij} (A_{ij} - P_{ij})^2. \tag{3.3}$$

3.2.1.2 Statistical Significance as a Constrain

As I mentioned in the previous section, I want to enforce the statistical significance of the communities I find. This constrain will also prevent the communities to over fit the graph, as only optimizing equation 3.3 would lead to having one community per edge, each with associated probability 1.

Establishing the statistical significance of a community is difficult. In practice, we want to measure a variable that corresponds to the goodness of the community and compare it with its probability distribution. However, the data the variable is based on can not have been used to decide which nodes were included in the community, as we would then only be measuring the efficiency of the algorithm at optimizing this variable. This makes things difficult as one would typically want to use some measure related to the edge density which will of course have been considered when finding the community. To go around this, I make the choice of using a variable that is data-dependent, but apply my algorithm to random networks to evaluate how unlikely it is to get such high values for it. This bootstrapping can be precomputed and the results stored so it does not have an impact on efficiency.

Even though this variable does not have to be related to significance, I choose to use the number of standard deviation away from the expected number of edges in my subgraph, as I expect it to be a good measure to compare the communities for significance. The formula for

this is:

$$S(U) = \frac{\mathrm{E}\left(\sum_{ij \in U} A_{ij} - E_{ij}\right)}{\sqrt{\mathrm{Var}\left(\sum_{ij \in U} E_{ij}\right)}} = \frac{\sum_{ij \in U} A_{ij} - P_{ij}}{\sqrt{\sum_{ij \in U} \mathrm{Var}\left(E_{ij}\right)}} \tag{3.4}$$

It is important to note that I asses the significance of a set of vertices U, not of a community. The reason for this is that I do not want a community to be part of the model as I assess its significance. Otherwise, due to constrain 3.2, the significance will be exactly zero. This also means whenever I need to test the significance of a community, I first need to remove it from the model and adapt the probabilities P_{C_k} of other communities. Then I can apply the formula above, taking U to be the set of nodes that belonged in the community.

3.2.1.3 Conclusion

Using this modeling approach has a number of advantages. First, it simplifies the problem of finding overlapping communities and assessing statistical significance. I mentioned in the preparation that a common problem in overlapping community detection was to find similar communities multiple times. Here however, this is not a problem since we change the null model after finding each community. Therefore, when looking at a set of nodes similar to a community that was previously found, the expected number of edges will be close the actual number of edges. For the same reason, the concept of significance is strengthened, as we are now assessing the significance of a community in the presence of the other found communities.

Second, it allows for the detection of different types of communities. In most community detection mechanism, each community is assumed to have the same properties. The variables P_{C_k} gives us some granularity to describe the behavior of the different communities. This will prove to be extremely useful when adapting the model to multiple layers.

3.2.2 Finding the Model

The previous section explained the basics of how the model could be used to find communities, in this section I explain how the probabilities associated with each communities are found.

Gathering together equations 3.1 and 3.2, the probabilities must satisfy the following function:

$$\forall k \in K \quad \sum_{i,j \in C_k} \left[1 - \prod_{\substack{C_l: \\ i,j \in C_l}} (1 - P_{C_l})\right] = ind_{C_k} \tag{3.5}$$

Where K is the set of community indexes. For m communities, this forms a set of m non-linear equations with m unknown which can not be resolved directly. To go around this, I solve it numerically using an iterative Newton method. This can be implemented very efficiently, as the partial derivative of the sum above simply is:

$$\frac{\partial}{\partial P_{C_k}} \sum_{i,j \in C_k} \left[1 - \prod_{\substack{C_l: \\ i,j \in C_l}} (1 - P_{C_l}) \right] = \sum_{i,j \in C_k} \prod_{\substack{C_l : l \neq k, \\ i,j \in C_l}} (1 - P_{C_l})$$

Once we have the probabilities P_{C_k}, we can easily compute each of the edge probability P_{ij} using equation 3.1. Since the random variables E_{ij} have a Bernoulli distribution, their variance used in 3.4 is

$$\mathrm{Var}\,(E_{ij}) = P_{ij}(1 - P_{ij}).$$

3.3 Extending the Scheme to Multiple Layers

In this section, I explain how the scheme that was presented in the previous section can be extended to multiple layers. I first go over how the model is extended to multiple layers and then explain how the statistical significance and optimization functions are generalized.

I mentioned in 2.3.3 that there were two extreme ways in which one could go about doing community detection on multiple layers. One extrema is to try to merge all the layers into one, discarding the fact that they contain different information and attempt to find some "average" communities. The other extrema is to apply community detection on each layer individually and find a different set of communities at each layer.

What makes the scheme presented above suitable to find a middle ground is that it allows communities to have a parameter that describes the density of edges inside the community. In the multiple layer community detection mechanism presented here communities are represented on each layer, but they are associated a different probability $P^l_{C_k}$ for each layer $l \in L$, where L is the set of all layers. Similarly, pairs of nodes are associated a different random variable E^l_{ij} at each layer. Figure 3.2 shows an example of this model on a two-layer network.

3.3.1 A Multi-Layer Model

Finding the P_{C_k}'s is done separately at each layer in the exact same way that it was done in the single layer case: the expected number of edge within each community matches the actual number of edge for each layer. Generalizing equation 3.2 gives:

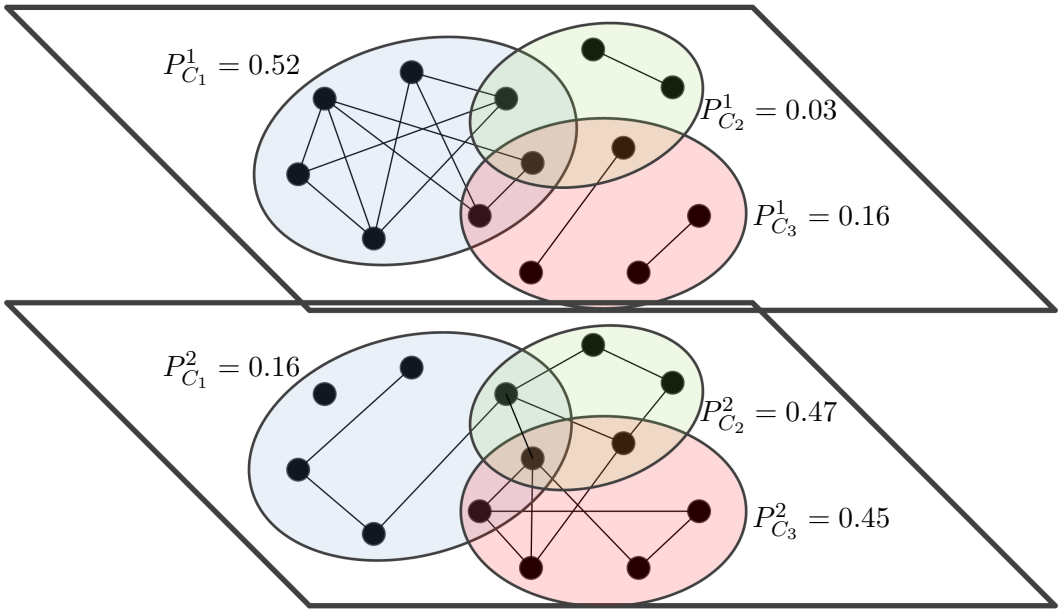


Figure 3.2: An example of the multi-layer model on a two layer network. Communities remain the same from one layer to another, but are associated different probabilities.

$$\forall k \in K, l \in L \quad ind^l_{C_k} = \sum_{i,j \in C_k} \mathrm{E}\left(E^l_{ij}\right) = \sum_{i,j \in C_k} P^l_{ij}.$$

In the next subsections, I will explain how I generalize statistical significance and graph fitting optimization to multiple layers.

3.3.2 Statistical Significance with Multiple Layers

Here I generalize the concept of statistical significance of a set U of nodes to multiple layers

What we want to avoid here, is to have a significance based on all the layers of the graph together, as this will lead to "average" communities that fail at representing each layer individually. However, we still want to use the fact that some layers may resemble each other. To solve this problem, I decide to use order statistics.

Given a cluster of which I need to assess the statistical significance, I start by calculating the single layer statistical significance of each layers. The significance of a subset U of nodes at layer l is given by:

$$S_l(U) = \frac{\sum_{ij \in U} A^l_{ij} - P^l_{ij}}{\sqrt{\sum_{ij \in U} \mathrm{Var}\left(E^l_{ij}\right)}}$$

Once this is done, I rank these significance in decreasing order. Then, using order statistics, I consider the probability of having such a high value for the highest significance, such a high value for the second significance and so on. A low probability corresponds to a high overall significance, it means it is unlikely to observe this configuration in a random graph. I keep the lowest of these probabilities as the overall significance of the cluster. Using this method allows me to only consider the layers on which the community is relevant. For a given layer, call $p_s = P(S_l(U) > s)$ the probability that the significance is higher than s on this layer. Given m layers, the order statistic of the q^{th} value is given by:

$$\Omega_q(s) = P(s_q > s) = \sum_{i=q}^{m} \binom{m}{i} p_s^i (1 - p_s)^{m-i}$$

Finding the value of p_s is complicated since I have once again picket the set U of nodes to evaluate in a data dependent way. For simplicity, I assume $S_l(U)$ is normally distributed with mean zero and standard deviation one. This would be the case if I had picked the set U at random. To compensate for this, I evaluate the multi-layer significance of random multi-layer networks after applying this procedure. This gives me a distribution for the value maximum value of $\Omega_q(s)$ which I can then use to asses the significance of my community.

3.3.3 Optimizing the communities with multiple layers

To optimize the communities with multiple layers, I first need to generalize equation 3.3 to multi-layer networks. The same way that I was trying to minimize the difference between the adjacency matrix and the model in the single layer case, I decide to minimize the difference between the adjacency matrix and the model at each layer of the network. The aim of the optimization becomes:

$$\min \sum_{\substack{ij \in V \\ l \in L}} (A^l_{ij} - P^l_{ij})^2. \tag{3.6}$$

I have now fully generalized the tools available from the single layer model to multiple layers. In the next section I explain how these tools can be use to find and optimize communities.

3.4 Finding and Optimizing the Communities

I now describe how the different tools offered by the model can be used to detect communities. As I mentioned, the communities that are revealed by the method need to be statistically significant. My method works in two steps. First I look for statistically significant communities and add then to the model. Then, I optimize the communities to find a best fit for the graph, while ensuring they remain statistically significant.

3.4.1 Finding Statistically Significant Clusters

A number of approaches have been proposed to optimize overlapping communities subject to a quality function (here that would be the statistical significance). The mechanism that I want to use needs to satisfy a few criterion:

- The method needs to be greedy. One of the main motivation of this project is its application on pocket switched networks, where no element has a full view of the network. Therefore, a greedy method where decisions are only made with local information is necessary so that the scheme can be decentralized.
- The method should be able to find communities of different sizes and types. Real life networks may contain communities with high number of nodes and only slightly higher than average density.

After consideration, I decided instead of using one of the previously published methods to extend one of the non-overlapped community detection techniques: the Louvain method. The Louvain method was designed for modularity optimization and takes advantage of the hierarchy that is often present in real networks. It has been reviewed as one of the most efficient greedy methods and scales to graphs with billions of nodes. I explain how the original method works as part of the related work in section 5.1. I now go over the extended method I have implemented.

To give an overview, the method works by producing communities of different level. It starts by finding communities at level 0, containing only a small number of nodes. Once it can not find anymore communities at this level, it creates a new graph where nodes correspond to communities or intersections of communities when they overlap. The community detection mechanism is then applied again on this new graph and produces communities of level 1 and so on. The algorithm returns when it can not find anymore communities.

To find a single community, the algorithm starts with a random node. It then goes through an iterative procedure whereby it considers each node inside the community for removal, and each node outside the community for inclusion. Every time, what is being considered is the gain in statistical significance generated by the change. If removing any of the nodes inside the community produces an improvement, it removes the node for which that improvement is the greatest. Otherwise, if including any of the nodes outside the community produces an improvement, it adds the node for which that improvement is the greatest. The procedure keeps on running until no more improvements can be made in which case it stops.

Finding the change in significance can be easily done. Call U the set of nodes in the community, for each layer l we keep track of three values: the in-degree ind_U^l of U, the expected in-degree $eind_U^l = \sum_{ij \in U} P_{ij}^l$ and the variance in the in-degree $vind_U^l = \sum_{ij \in U} P_{ij}^l (1 - P_{ij}^l)$. The statistical significance of U on this layer is then:

$$S^l(U) = \frac{ind_U^l - eind_U^l}{\sqrt{vind_U^l}}$$

Then, for each node v, we also keep track for each layer l of three values: ${}_v ind_U^l$ the number of edges v has to members of U, the expected number of edges ${}_v eind_U^l = \sum_{i \in U} P_{iv}^l$ and the

variance in the number of edges ${}_v ind_U^l = \sum_{i \in U} P_{iv}^l (1 - P_{iv}^l)$. The new significance from adding a node v that is outside U or removing node v that is inside U can then be computed as:

$$S^l(U \cup \{v\}) = \frac{(ind_U^l - eind_U^l) + ({}_v ind_U^l - {}_v eind_U^l)}{\sqrt{vind_U^l + {}_v vind_U^l}}$$

$$S^l(U \backslash \{v\}) = \frac{(ind_U^l - eind_U^l) - ({}_v ind_U^l - {}_v eind_U^l)}{\sqrt{vind_U^l - {}_v vind_U^l}}$$

It is then easy to find the overall change in significance using the order statistics method described in 3.3.2. As you can see, this method has the advantage of only using local information to decide whether a node should be added.

If the community has a statistical significance above some threshold θ, it is stored and added to the model. Otherwise, it is discarded. It is important to note that a community may be significant when it is found but become insignificant after some other overlapping communities are found. For this reason, every time a community is stored, all previously stored overlapping communities are tested again for significance. If any of them fail, they are deleted and removed from the model.

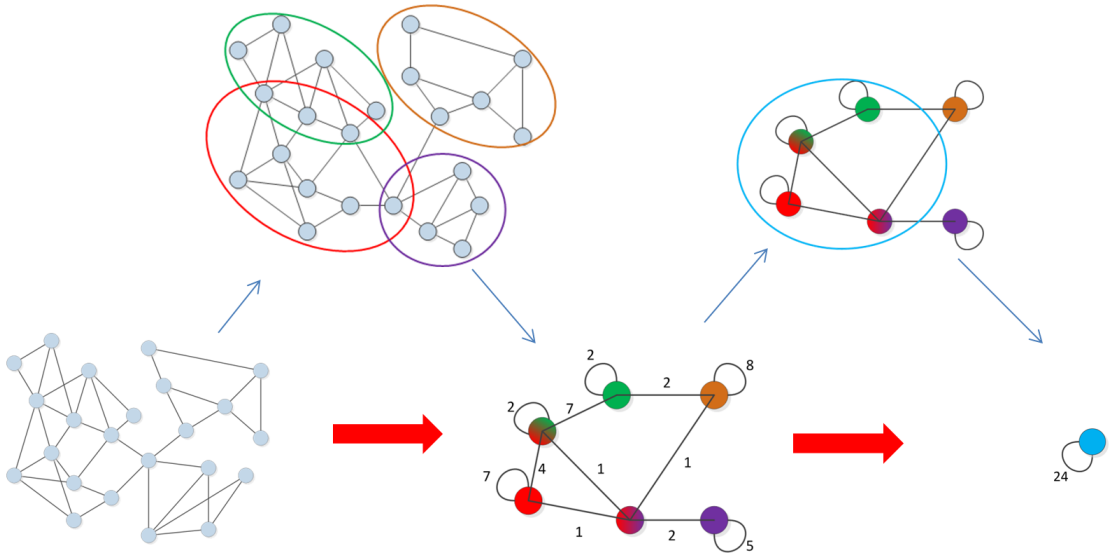


Figure 3.3: An example of applying the extended Louvain method on a single layer network. Note how nodes that were not included in communities are not represented on higher levels.

Once no more communities can be found, a new network is built based on the communities found during the previous phase. Because the communities are overlapping, each node in this new network corresponds either to a single community, or the intersection of overlapping communities. This is described by the example on figure 3.3.

The procedure of finding communities is then ran again on this new network, with one difference. Instead of considering the addition or removal of individual nodes, the procedure only

considers whole communities from the previous level. Adding or removing a community will result in adding or removing each of the nodes that constitute the community. This is useful to reduce the search space as a community may have a high number of overlaps, and therefore nodes, associated to it.

Once a community is found at this new level, and no more improvements can be made, I consider the addition or removal of individual nodes or communities from the level below the network being currently considered. Say I have found a new level 3 community constituted of level two nodes, I will then try to improve the community in the same way by adding or removing level 1 nodes, and once this is done, by adding or removing level 0 nodes. This allows me to restrict the search space while finding the bulk of the community, and then perform a more fined grained search but with a hopefully low number of changes.

In conclusion, the method described here offers a number of advantages:

- It can be decentralized. A few of the restrictions used would need some loosening such as always picking the best node to add in the community, but the fact that the decision is based only on local information along with some information from the community itself means it the procedure could be turned into a protocol whereby nodes themselves make the decision to join or leave the community.
- It is hierarchical. The reason why the original Louvain method performs extremely well for modularity optimization is that it takes advantage of the fact that real world networks are often hierarchical. This is what allows me to restrict the search space in the higher levels and therefore gain in performance without loosing quality in the final result.
- It detects overlapping communities. Since each community that is found is added to the model, communities that will be found afterwards will be forced to differ significantly. This avoid the problem I mentioned in the background of finding the same communities multiple times.
- It detects communities of different types. Often, when a level 0 community is found, it could be further improved by adding a small set of nodes into it, despite the fact that no individual node generates an improvement. The advantage of keeping this local maximum is that it forces the detection to initially find small and dense communities, leaving the discovery of larger communities for higher levels.
- It works well with multiple layers. This comes mostly from having adapted the multi-layer significance in a way that recognizes that some layers may be correlated and some others may not.

I have now described the first part of the detection method that finds statistically significant communities. In the next subsection, I show how these communities are optimized without loosing their significance to find the best fit for the graph.

3.4.2 Optimizing the Communities

As I said in 3.2.1.1, the aim of the project is to find communities that best represent the graph. Once we have found statistically significant communities, the aim then becomes to

make them represent the graph as well as possible without loosing their significance. As before, it is important to find a method that does this using only local information as this means the algorithm can be decentralized.

I choose to use a very simple greedy method, where each node locally makes the decision of joining or leaving a community, depending on whether this improves the approximation of the graph. The method follows the algorithm shown in figure 3.4

```
for v ∈ V do
    for k ∈ K do
        if v ∈ C_k then
            TRYREM(v, C_k)
        else
            TRYADD(v, C_k)

function TRYREM(v, C):
    if Q(C\{v}) >Q(C) then
        if S(C\{v}) > θ then
            remove(v, C)
        else
            delete(C)

function TRYADD(v, C):
    if Q(C ∪ {v}) >Q(C) then
        if S(C ∪ v) > θ then
            add(v, C)
```

Figure 3.4: Algorithm used to optimize the communities of the graph.

Here, I take the function Q to represent the quality of the graph representation as is described by equation 3.6. In the implementation, this is done by looking at the edges linking vertex v with the vertices in C_k and checking if they are better approximated if v is inside or outside of C_k. For a node v inside C_k, the decision to leave the community is taken if:

$$\sum_{\substack{i\in C_k \\ l\in L}} (A^l_{iv} - \frac{P^l_{iv}}{P^l_{C_k}})^2 < \sum_{\substack{i\in C_k \\ l\in L}} (A^l_{iv} - P^l_{iv})^2.$$

Similarly, if v is outside C_k, the decision to join the community is taken if:

$$\sum_{\substack{i\in C_k \\ l\in L}} (A^l_{iv} - P^l_{iv}P^l_{C_k})^2 < \sum_{\substack{i\in C_k \\ l\in L}} (A^l_{iv} - P^l_{iv})^2.$$

Again, this method has the advantage of being decentralizable. The only information necessary for the decision of joining or leaving the community is local to the concerned node. This means the procedure could be turned into a protocol whereby nodes take the decision of joining or

leaving communities themselves. It also fulfills our requirement of optimizing the communities whilst enforcing the constrain of retaining their statistical significance.

3.5 Conclusion

This concludes the implementation chapter. Here, I have used a modeling approach to implement multi-layer community detection. I described a single-layer model along with some constrains and a function to be optimized. I then generalized this model to multiple layers in a way that took account of the fact that some but not all layers may be correlated. Finally I devised two algorithm. The first one was a generalization of a famous non-overlapping community detection algorithm. I use it to find statistically significant communities. The second one was a simple and efficient way to optimize the communities without loosing their statistical significance.

Chapter 4

Evaluation

In this chapter I perform the evaluation of my multi-layer community detection mechanism. I first start by finding the threshold θ for statistical significance by applying my algorithm to random networks. I then test my algorithm on a synthetic network, where the communities are known, and check the results are the expected ones. Finally, I apply my method to real world data from Pocket Switched Networks (PSNs).

4.1 Finding the Significance Threshold

In this section, I find the threshold θ that is necessary to ensure the communities found are significant. θ depends upon the following parameters:

- The number of vertices $\mid V \mid$. In a graph with many nodes, it is more likely that the algorithm will find communities with high significance. θ should therefore increase with the number of nodes

- The number of layers L. The significance used is based on the best of L order statistic and therefore the distribution of θ may be dependent on the number of layers.

- The correlation between the layers. It is possible that the algorithm may be fed a graph where individual layers are uncorrelated, showing no community structures, but layers are correlated to each other. In this case, no communities should be found, despite the fact that the order statistic should return a high value. To go around this, I evaluate the correlation between layers before running the algorithm and use it as an input to find the appropriate θ.

I first define a function to evaluate the correlation between layers. I then evaluate the best communities found on random networks with varying number of nodes, number of layers, and correlation between layers.

4.1.1 Finding the Correlation Between Layers

In this subsection, I introduce a parameter ρ that rates how correlated the layers of the network are to each other.

It is important to note that this parameter should be adapted to the fact that I use order statistics over the layers to rate the significance of the network. Therefore ρ should be based on a set of "best" layers that are highly correlated. The larger the set, and the higher the correlation, the higher the value of ρ should be. There is no need to worry about the quantitative value of ρ, as it will later be rescaled by measuring the corresponding θ. What is important is that the ρ parameter of a random network is highly correlated with the significance of the communities found in that network.

I start by defining a correlation coefficient $\rho_{G_xG_y}$ between two graph G_x and G_y. I define it as the Pearson's correlation between two pairs of nodes in the two graphs. That is:

$$\rho_{G_xG_y} = \rho(E^x_{ij}, E^y_{ij}) \tag{4.1}$$

$$= \frac{\mathrm{Cov}\left(E^x_{ij}, E^y_{ij}\right)}{\sqrt{\mathrm{Var}\left(E^x_{ij}\right)\mathrm{Var}\left(E^y_{ij}\right)}} \tag{4.2}$$

which can be easily computed. Note that $\rho_{G_xG_y}$ is a value between -1 and 1. Now I need to define a value of ρ for the whole graph based on a set of worse layers that are highly correlated. Given a set S of layers, I call:

$$\rho_S = \mid S \mid \times \frac{2\sum_{xy\in S}\rho_{G_xG_y}}{\mid S \mid \times(\mid S \mid -1)} \tag{4.3}$$

$$= \frac{2\sum_{xy\in S}\rho_{G_xG_y}}{\mid S \mid -1}. \tag{4.4}$$

That is, the number of elements in S times the average value of the correlation between the graphs in S. I finally define ρ of the network to be the maximal such value of ρ_S. Unfortunately, finding this maximum value is NP-complete (funnily enough, it is a community detection problem), but I use a simple greedy approach to try and find a maximum. Again, it is not important to find the exact value of ρ but rather, we want an estimate of how correlated the layers are.

4.1.2 Application to Random Networks

In this subsection, I apply my algorithm to random networks in order to find the distribution of the maximum statistical significance that can be found in each network. I use an Erdős-Rényi random model to test my method on as it does not naturally produces communities.

I start by evaluating the inpact of the number of nodes and number of layers to the maximum significance found in the network. To do this, I generate networks, with the number of nodes in the range $10, 20, \ldots, 100$, and layers in the range $1, 2, \ldots, 5$. Each layer is generated as a separate Erdős-Rényi graph. For each of those data points, I do 20 simulations in order to evaluate the distribution of the maximum significance. Surprisingly, the number of layers had

small to no influence on the results obtained. For this reason, I only show the results obtained for networks with one layer and five layers. Those are shown in figure 4.1.

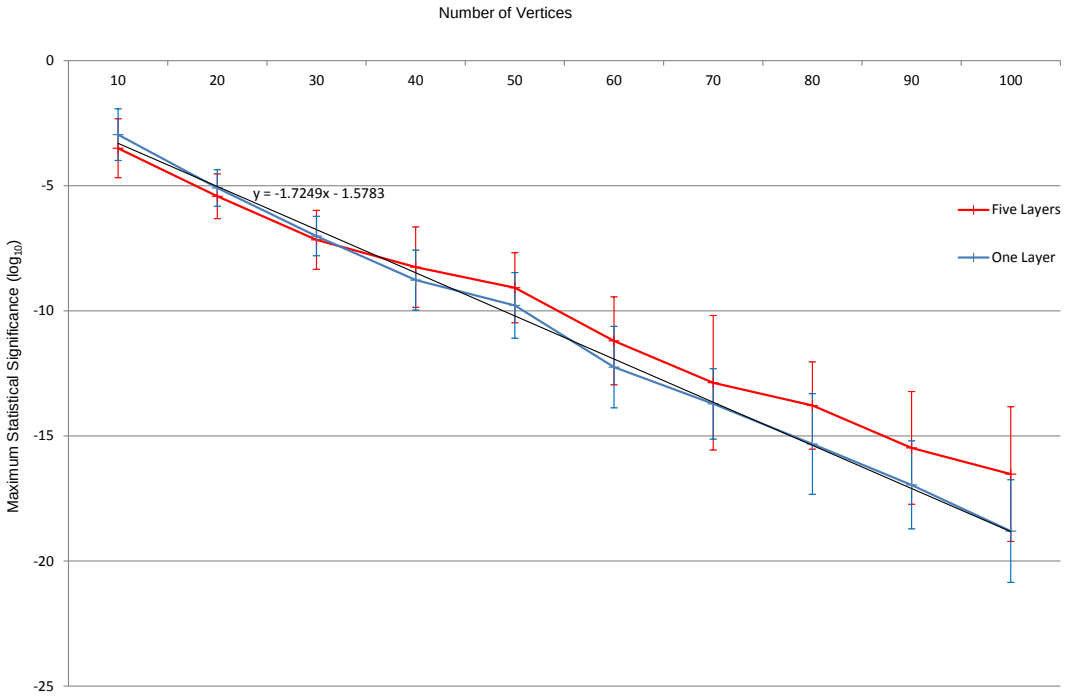


Figure 4.1: Maximum significance found in a randomly generated network. Each data point corresponds to a sample of twenty simulations.

Note that I have used a log scale to plot the significance. Here the maximum significance corresponds a probability: the probability of getting such a high density of edges in a random graph if the nodes of the community were selected at random. Each data point corresponds to the average value found in the log scale, and the error bars correspond to the standard deviation found within each sample. The results clearly show a linear relationship between the significance in the log scale and the number of vertices.

Usually, one refers to the significance as the probability of not getting such a value (e.g. 95% significance, rather than 5%), but the numbers used here make this notation impractical. For convenience however, I will keep on referencing to a low probability as a "high" significance.

Next I try to evaluate how the maximum significance found varies with ρ. To do this, I generate random networks and try to randomly correlate some of the layers with one another. All the networks generated have the same number of layers, 10. In each case, a number n of layers, where n was picked uniformly at random, are created by randomly modifying one original graph. The other $10 - n$ graphs are computed independently at random. This gives

me a range of values for ρ which I can use to estimate the relationship between ρ and the maximum significance found in the network. The results are shown in figure 4.2.

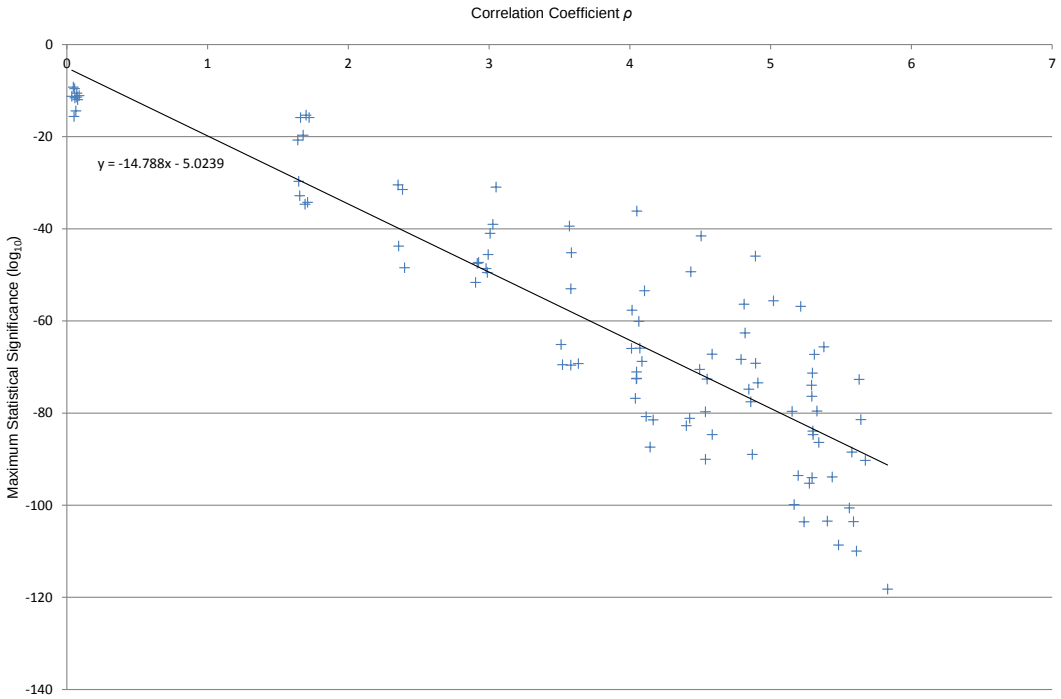


Figure 4.2: Maximum significance found in a randomly generated network with correlated layers. Each data point corresponds to the maximum significance found in a single network with 100 vertices and 10 layers.

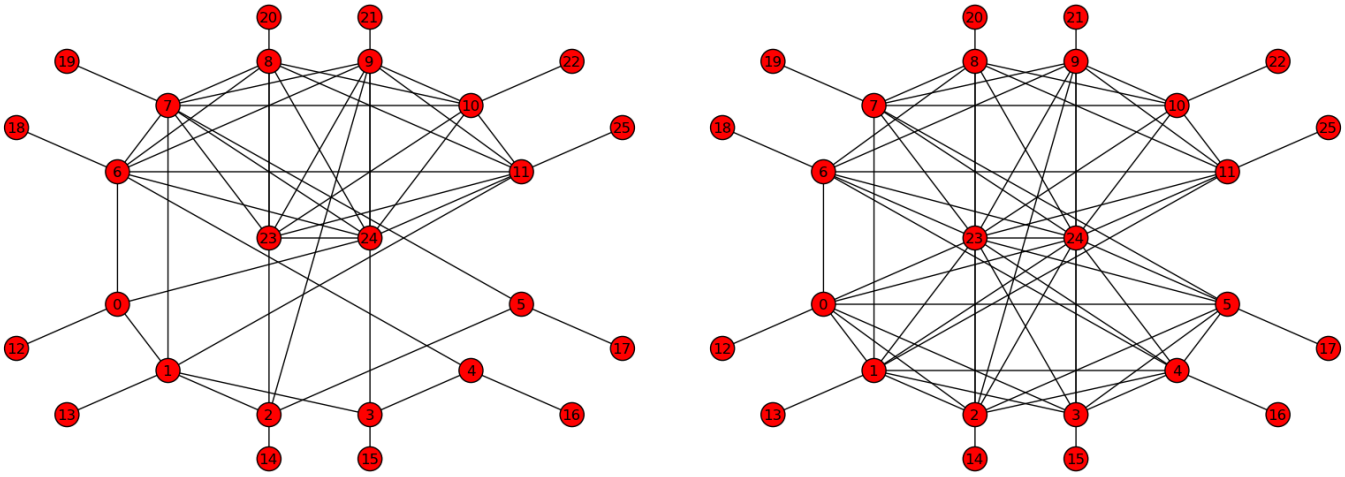
As you can see, there is a somewhat linear relationship between ρ and the maximum significance found in the log scale. In the next subsection, I use the results of both experiments to find a value for θ.

4.1.3 θ as a Function of the Network

From the two previous set of experiments, I can construct a function of $\mid V \mid$ and ρ that tells me the lowest significance that can be considered as statistically significant.

I decide to base the coefficient associated with ρ on the highest significance observed. ρ and the gradient of the worse point two standard deviation away from the mean

Since we saw that log of the significance grows linearly with both $\mid V \mid$ and ρ, I need to assign a coefficient to each of these variables to find the threshold θ. In the first case, I base the coefficient on the worse point that is two standard deviation away from the mean. In the



(a) First Layer *(b) Second Layer*

Figure 4.3: The two layers of the test network

second case, I base the coefficient on the worse value observed for ρ over the 100 experiment. Picking $\log_{10}\theta = -5$ as the maximum possible value, this leads to the equation:

$$\log_{10}\theta = -5 - 2.5 \mid V \mid -20\rho$$

Which is the formula I will use for the rest of the evaluation.

4.2 Evaluation on Synthetic Networks

In this section, I evaluate the performance of my algorithm at detecting communities in a synthetic network where the communities are known. Unfortunately, there are no benchmarks available to test multi-layer community detection. Therefore, I choose to apply my algorithm to hand made multi-layer network and show that the communities found are the expected ones. Of course, this does not constitute a rigorous evaluation, but in the absence of benchmarks it is not possible to fairly test the method proposed here. This section should therefore be viewed as a proof of concept.

I choose to apply my algorithm on the two layer network shown in figure 4.3. Two communities were clearly implemented in this network: 0, 1, 2, 3, 4, 5, 6, 23, 24 and 6, 7, 8, 9, 10, 11, 23, 24. This network tests two main features of my algorithm. First, the ability to detect overlapping communities as the nodes 23 and 24 are in both communities. Second, the ability to function over multiple layers. If both layers were merged, most community detection would return a single cluster, as there are many edges going from one community to another.

After running the algorithm on the data, the result is the one expected: both communities are found with significances 8.10^{-25} and 4.61^{-47}. This shows that my algorithm is successful at detecting communities. In the next section, I apply on real world data to test it's ability to find communities with different behaviors.

	Vertices
Community 1	16, 21, 22, 27, 28, 32, 34, 36
Community 2	3, 5, 7, 11, 12, 18, 21, 22, 25, 28, 29, 32, 34, 36
Community 3	6, 10, 14, 15, 18, 19, 20, 21, 23, 25, 33, 34
Community 4	24, 26, 35
Community 5	2, 3, 6, 11, 13, 16, 17, 29, 34

Table 4.1: Vertices per community in Test 1.

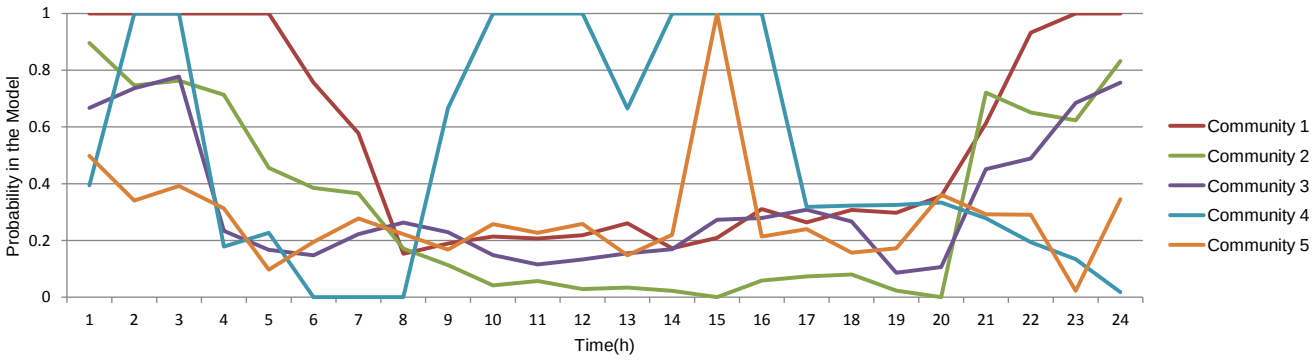


Figure 4.4: Probability model of communities per layer in Test 1

4.3 Evaluation on Real World Data

In this section, I evaluate the performance of my algorithm on real world data. I do this by applying my algorithm on collected data from PSNs.

Here, I use a dataset gathered by the Haggle Project [9] in the Cambridge Computer Laboratory. iMotes were distributed to thirty five undergraduate students from two different year groups. The data consists of contact graphs between the students at regular 10 minutes intervals for 11 days.

The aim of the evaluation here is to observe the ability of the algorithm to detect different types of communities that interact in different ways. In a recent paper [10], using joint diagonalization, Fay *et al.* have observed that this particular dataset contained different main "modes" of activity. In terms of my model, this means that the various communities should differ in the way the probability associated with their model behaves depending on the layer.

I perform two tests. In Test 1, I construct a network with 24 layers, one for each hour of the day. I assign an edge between two nodes on layer l if there has been a contact between the students at some point over the 11 days at the hour of the day associated to l. In Test 2, I consider five consecutive days which I cut into two hour slots. I assign an edge between two nodes if they have been in contact during the corresponding slot. The results of Test 1 are shown in figure 4.4 and table 4.1. The results of Test 2 are shown in figure 4.2 and table 4.2.

Looking at the results from Test 1, there are a number of observations that should be made. First, look at communities 1, 2 and 3 from Test 1. They represent the bulk of the network: communities 2 and 3 are the two biggest ones, and the number of edges a community represents

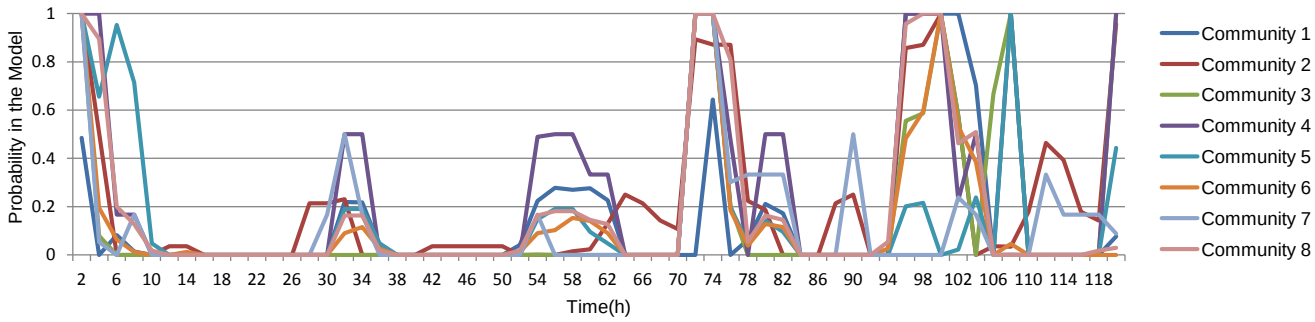


Figure 4.5: Probability model of communities per layer in Test 2

	Vertices
Community 1	22, 25, 27, 28, 32, 34, 36
Community 2	2, 4, 6, 14, 15, 18, 19, 20
Community 3	13, 19, 33
Community 4	3, 16, 21, 34
Community 5	3, 7, 12, 17, 28, 34, 36
Community 6	8, 11, 13, 15, 16, 21, 22, 25, 27, 28, 32, 34, 36
Community 7	14, 18, 29, 34
Community 8	3, 7, 16, 21, 22, 25, 27, 28, 32, 34, 36

Table 4.2: Vertices per community in Test 2.

grows quadratically with the number of vertices it contains. They also have the same general shape of being high in the ranges 1-4 and 21-24. This has already been observed in other publications analyzing this dataset and corresponds to the morning hours, where students go to lectures together. Now, consider communities 4 and 5. They have a completely different behavior, peeking in the evening and early hours of the morning. They most likely correspond to a group of friends who see each other independently of lectures. It is also interesting to note that most of the nodes in community 5 are also represented in community 1,2 or 3.

Turning our attention to Test 2 which was performed over the course of five consecutive days. Note how the peaks of contact happening at hours 34 and 58 are surprisingly low. These probably correspond to the week-end meaning students did not have lectures to attend. This is also supported by the fact that the peaks seem to happen during the afternoon and evening rather than in the morning when lectures happen. Again, all the communities do not behave in the same way. Community 3 for example has members which seem to sit together every lecture, but did not see each other over the week-end. The members of communities 2 and 7 also seem to have a tendency to see each other in the evenings.

In both cases, the method was able to find communities that had various behavior which was the aim of this project. To contrast these results with the tools currently available, I apply modularity on the data available for Test 1. To do this, I construct a weighted network based on the twenty four layers that were available to my algorithm, the weight of each edge being proportional to its number of occurrences over the twenty four layers. Once this is done, I feed the produced communities into my model to observe the distribution of edges per layers

	Vertices
Community 1	1, 2, 4, 5, 6, 9, 10, 13, 14, 15, 17, 18, 19, 20, 23, 24, 26, 30, 31, 33, 35
Community 2	3, 7, 8, 11, 12, 16, 21, 22, 25, 27, 28, 29, 32, 34, 36

Table 4.3: Vertices per community after applying Modularity to Test 1.

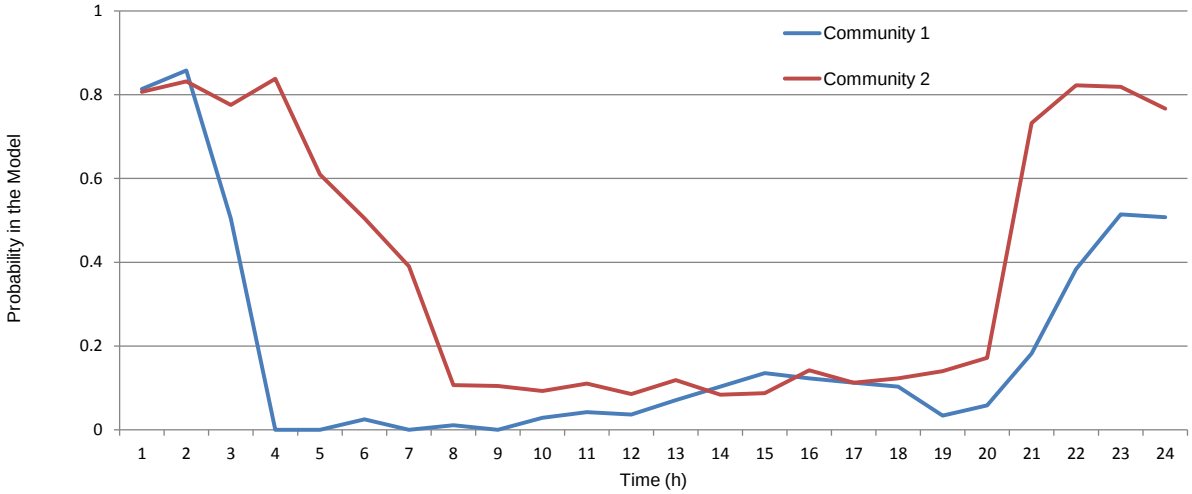


Figure 4.6: Probability model per layer of the communities obtained with modularity with the Test 1 data

in these communities. The results are shown in figure 4.6 and table 4.3.

The two communities clearly correspond to the two different year groups. As you can see, both communities behave similarly, and no conclusions can be made about the different types of interactions. The fact that, using my algorithm, we can detect different communities with different behaviors shows the success of the approach proposed here. However, it has to be pointed out that the communities produced by modularity are "cleaner". For example, a few of the communities generated by Test 2 behave similarly and overlap significantly. I believe this is due to the greedy optimization algorithms not performing as well as I initially hoped.

4.4 Conclusion

In this chapter, I applied my algorithm to random networks in order to get a distribution for the statistical significance of the communities found. From there, I deduced a formula for the minimum statistical significance one should use. I then tested my algorithm on a synthetic network to verify its capacity to detect communities. Finally, I applied the algorithm to real world data generated from pocket switched networks. The results were very good as I managed to detect different types of communities that interacted at different times. I contrasted this with what would be the current network analysis approach when given a multi-layer graph: merging the layers and applying a standard community detection method. This showed to give much less useful results in which all communities behaved in the same way.

In conclusion, the method I have proposed in Chapter 3 has shown to be efficient at detecting multi-layer communities. In particular, the use of order statistics on the different layers to rank the significance of a community has shown to be a good way to detect "unusual" communities, whose behavior differs from the average behavior of the network. However, it seems that the greedy optimization algorithms proposed could be improved. A number of the nodes were not included in any of the communities. Some of the communities also had high overlaps and behaved the same way, meaning it would have been more accurate to have them as a single community. I believe these issues are mostly to do with being trapped in local maxima, and future work should be done to establish how to improve the optimization algorithms.

Chapter 5

Related Work

In this chapter I describe the work my project has built upon. I first review similar work that has been done in the area of single layer community detection and then go over the different approaches that have been used for multi layer community detection. Finally, I describe the work that has been done in related fields.

5.1 Single Layer Community Detection

OSLOM [8] is a single layer overlapping community detection mechanism based on statistical significance. It uses the statistical significance measure introduced by Lancichinetti *et al.* [11] as a quality function and attempts to find clusters of nodes that optimize it. The statistical significance function is built using order statistics. It measures the probability that the worse node in the cluster would have such a high number of edges pointing inside the community, also referred to as in-degree.

In these papers, it is assumed that the probability distribution of the in-degree of the worse node in the community does not differ from the probability distribution of the in-degree the best node outside the community. This is extremely useful as the best node outside the community has not been selected in a data-dependent way and therefore this distribution can be found using simple statistics. This contrasts with the statistical measure I use which is data-dependent and therefore necessitated me to apply it to random networks in order to find the correct threshold. However, this comes at the cost of evaluating the whole community based on the worse node which can be inaccurate.

OSLOM uses a strict hierarchy mechanism whereby communities at the highest level are initially found, and the algorithm is then reapplied to the subgraph of that community to find lower level communities.

Long *et al.* [12] use graph approximation to detect communities. They create a *community prototype graph* A^* based on the edge density within and between communities, and turn the the problem of community detection into an optimization problem:

$$\min \parallel A - A^* \parallel^2 .$$

They require the number of communities to be given as input. They allow for overlapping

communities but with the restriction that each node has a belonging factor to each community, and the sum of all belonging factors for any given node must be 1.

The general approach I use in this dissertation can also be compared with generative models such as Bayesian inference [13], or model selection methods [14]. As it turns out, the techniques used for community detection in these areas are usually very different from the ones used in this project. I will therefore not review them here but see [1] for a survey of their application to community detection.

Delvenne *et al.* [15] introduce the concept of stability of a partition, using the dynamics of the graph, and use it as a measure of the quality of the partitioning. The method can be shown to generalize a number of classical partitioning measures such as modularity, and some of the most used spectral techniques.

Blondel *et al.* [16] published the Louvain method for modularity optimization. Recall that modularity is a non-overlapping quality function. The method works by initializing the graph with one community per node. Individual nodes may then move from one community to another as long as it improves the value of modularity. Once a local maximum is found, a new graph is created with each node as a community from the previous level, and the same procedure is applied.

This method performs extremely well for modularity optimization, mostly due to the fact that it takes advantage of the hierarchy that is often present in real-world networks.

5.2 Multi-Layer Community Detection

Two main approaches have been used for multi-layer community detection. Mucha *et al.* [17] generalize the modularity null model to multiple layers using a parameter to control the coupling between the different layers. Then, a standard modularity optimization heuristic can be used to detect these multi-layer communities.

Dong *et al.* [18] generalize spectral clustering algorithms – algorithms that use the eigenvectors of the Laplacian matrix of the graph – to multiple layers. They propose two different algorithms to obtain "average" eigenvectors that try to fit as well as possible with the Laplacian matrix of each layer.

It is important to note that in both cases, the communities generated are the same on each layer.

5.3 Work in Related Research Areas

Hui *et al.* [19] consider the design of decentralized community detection mechanisms for PSNs. They use the measure of local modularity introduced by Clauset [20] and introduce three algorithms to detect communities which tradeoff between accuracy and resource requirements.

Fay *et al.* [10] use joint diagonalization on contact graphs. They create an average eigenspace of the network and look at the distribution of deviations from this average. They find that the network goes over a number of *modes* which are found to correspond to particular times.

From the perspective of this project, these modes correspond to the fact that there are only a few different layer patterns.

Nicosia *et al.* [21] extend the notion of connectedness to the case of *time-varying networks*. They consider the problem of finding strongly connected components and show that, in this context, it is NP-Complete.

Chapter 6

Conclusion

This research project concerned the construction of a community detection method that could be applied on networks with multiple layers.

The project was a success. I designed a new modeling approach to community detection based on statistical inference. In this approach, the aim of the detection is to find the model that best fits the graph, under the constrain of having statistically significant communities. Using order statistics, I generalized the model to multi-layer networks. To find the communities, I extended one of the most used non-overlapping community optimization algorithm to detect overlapping communities. Finally, I constructed an algorithm to optimize the communities once they were found.

I had difficulties performing the evaluation on my method, due to the lack of benchmarking tools for community detection on multi-layer networks. However, the application of the algorithm on real world data from pocket switched networks showed it could successfully take advantage of the extra information available in the multiple layers, and I was able to detect multi-layer communities that could not have been found using standard community detection tools.

The methods used to find and optimize the communities were designed to be easily decentralizable, so that the method could be implemented as part of a protocol in pocket switched networks. An interesting future work would be to study whether removing this restriction could lead to more accurate algorithms to find the optimal communities.

Bibliography

[1] S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5):75–174, 2010.

[2] P. Hui, J. Crowcroft, and E. Yoneki. Bubble rap: social-based forwarding in delay tolerant networks. In *Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*, pages 241–250. ACM, 2008.

[3] G. Malewicz, M.H. Austern, A.J.C. Bik, J.C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski. Pregel: a system for large-scale graph processing. In *Proceedings of the 2010 international conference on Management of data*, pages 135–146. ACM, 2010.

[4] B. Shao, H. Wang, and Y. Li. The trinity graph engine. Technical report, Technical report, Microsoft Research, 2012.

[5] M.E.J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113, 2004.

[6] S. Kelley. *The existence and discovery of overlapping communities in large-scale networks*. PhD thesis, RENSSELAER POLYTECHNIC INSTITUTE, 2010.

[7] J. Xie, S. Kelley, and B.K. Szymanski. Overlapping community detection in networks: the state of the art and comparative study. *Arxiv preprint arXiv:1110.5813*, 2011.

[8] A. Lancichinetti, F. Radicchi, J.J. Ramasco, and S. Fortunato. Finding statistically significant communities in networks. *PloS one*, 6(4):e18961, 2011.

[9] J. Su, J. Scott, P. Hui, J. Crowcroft, E. De Lara, C. Diot, A. Goel, M.H. Lim, and E. Upton. Haggle: Seamless networking for mobile applications. In *Proceedings of the 9th international conference on Ubiquitous computing*, pages 391–408. Springer-Verlag, 2007.

[10] D. Fay, J. Kunegis, and E. Yoneki. On joint diagonalisation for dynamic network analysis. *Arxiv preprint arXiv:1110.1198*, 2011.

[11] A. Lancichinetti, F. Radicchi, and J.J. Ramasco. Statistical significance of communities in networks. *Physical Review E*, 81(4):046110, 2010.

[12] B. Long, X. Wu, Z. Zhang, and P.S. Yu. Community learning by graph approximation. In *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, pages 232–241. IEEE, 2007.

[13] R.L. Winkler. *An introduction to Bayesian inference and decision.* Holt, Rinehart and Winston, 1972.

[14] K.P. Burnham and D.R. Anderson. *Model selection and multimodel inference: a practical information-theoretic approach.* Springer Verlag, 2002.

[15] J.C. Delvenne, S.N. Yaliraki, and M. Barahona. Stability of graph communities across time scales. *Proceedings of the National Academy of Sciences*, 107(29):12755–12760, 2010.

[16] V.D. Blondel, J.L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008:P10008, 2008.

[17] P.J. Mucha, T. Richardson, K. Macon, M.A. Porter, and J.P. Onnela. Community structure in time-dependent, multiscale, and multiplex networks. *Science*, 328(5980):876–878, 2010.

[18] X. Dong, P. Frossard, P. Vandergheynst, and N. Nefedov. Clustering with multi-layer graphs: A spectral perspective. *Arxiv preprint arXiv:1106.2233*, 2011.

[19] P. Hui, E. Yoneki, S.Y. Chan, and J. Crowcroft. Distributed community detection in delay tolerant networks. In *Proceedings of 2nd ACM/IEEE international workshop on Mobility in the evolving internet architecture*, page 7. ACM, 2007.

[20] A. Clauset. Finding local community structure in networks. *Physical Review E*, 72(2):026132, 2005.

[21] V. Nicosia, J. Tang, M. Musolesi, G. Russo, C. Mascolo, and V. Latora. Components in time-varying graphs. *Arxiv preprint arXiv:1106.2134*, 2011.