Community Detection in Undirected Graphs Using a New Quality Measure

A Thesis

submitted to the designated
by the General Assembly of Special Composition
of the Department of Computer Science and Engineering
Examination Committee

by

Nikolaos Koufos

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE
WITH SPECIALIZATION
IN SOFTWARE

University of Ioannina

July 2017

DEDICATION

To my family and my friends.

ACKNOWLEDGMENTS

I would like to express my sincerest thanks and gratitude to my advisor Prof. Aristidis Likas for the valuable guidance, advice he has offered during the elaboration of this thesis. For his excellent work ethic. Our collaboration has been a pleasant and memorable experience that has helped me develop strong research skills as well as develop my critical thinking.

I would also like to thank my colleagues for creating a pleasant and friendly environment at the office and for the useful conversations we had during the past years. It has been a privilege to conduct my research among them.

I would also like to thank my parents, Charalampos and Theodora, and my siblings Giorgos and Fani for always believing and supporting me.

Ioannina, July 2017

Nikolaos Koufos

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Nikolaos Koufos
MSc, Computer Science and Engineering, University of Ioannina, Greece
July 2017
Title: Community Detection in Undirected Graphs Using a New Quality Measure
Supervisor: Aristidis Likas

The detection of communities is of great significance in sociology, biology, computer science and other disciplines where complex systems are often represented as graphs or networks. One of the most interesting properties of graphs representing real systems is community structure, i.e. the partitioning of graph nodes into clusters, with many edges joining nodes of the same cluster and comparatively few edges joining nodes of different clusters. This hard but important problem has attracted an increasing scientific interest over the past few years and several techniques have been proposed, especially for the case where the number of communities is not known in advance.

The most popular family of community detection methods is based on the optimization of the so called 'modularity' criterion using various clustering approaches. The modularity of a community is defined as fraction of the edges that fall within a given group minus the expected fraction if edges were distributed at random. However, it has been shown that modularity has several drawbacks, such as for example the 'resolution limit', i.e., it is unable to detect small communities.

We introduce a new quality measure to evaluate a partitioning of a graph into communities that is called 'inclusion'. This quality measure evaluates how well each node is 'included' in its community by considering both its existing and its non-existing edges. We have implemented several techniques to optimize the inclusion criterion. A first technique follows the agglomerative principles, as it starts with every node in a separate community and iteratively merges communities so that inclusion is improved. A second technique is similarly initialized, but instead of community merging, it improves the inclusion of the partitioning by moving each time a single node to another community. Another method is based on evaluating the solutions provided by spectral clustering. In the experimental evaluation we conducted, it has been shown that the inclusion measure is very effective in evaluating communities and usually leads to improved community detection results without requiring the a-priori specification of the number of communities.

ΕΚΤΕΤΑΜΕΝΗ ΠΕΡΙΛΗΨΗ ΣΤΑ ΕΛΛΗΝΙΚΑ

Νικόλαος Κουφός
MSc, Τμήμα Μηχανικών Η/Υ και Πληροφορικής, Πανεπιστήμιο Ιωαννίνων
Ιούλιος 2017
Τίτλος: Εντοπισμός κοινοτήτων σε μη κατευθυνόμενα γραφήματα με ένα νέο κριτήριο ποιότητας διαμέρισης
Επιβλέπων: Αριστείδης Λύκας

Ο εντοπισμός κοινοτήτων παίζει σημαντικό ρόλο στην κοινωνιολογία, βιολογία, επιστήμη υπολογιστών καθώς και σε όλους τους τομείς όπου πολύπλοκα συστήματα , συχνά αναπαρίστανται ως γραφήματα η δίκτυα. Μία από τις πιο ενδιαφέρουσες ιδιότητες που έχει η αναπαράσταση με γραφήματα, είναι η δομή του σε κοινότητες, δηλαδή, η διαμέριση του γράφου σε συστάδες που απαρτίζονται από κόμβους που συνδέονται με πολλούς κόμβους της ίδιας συστάδας μέσο ακμών, και όσο δυνατόν λιγότερους κόμβους που ανήκουν σε άλλες συστάδες. Αυτό το πρόβλημα, παρά την δυσκολία του, έχει κεντρίσει το ενδιαφέρων διάφορων επιστημών τα τελευταία χρόνια, με αποτέλεσμα να προταθούν αρκετές τεχνικές επίλυσης του προβλήματος, κυρίως για τις περιπτώσεις όπου ο αριθμός των συστάδων δεν είναι γνωστός εκ των προτέρων.

Η πιο γνωστή οικογένεια μεθόδων εντοπισμού κοινοτήτων είναι βασισμένη στην βελτιστοποίηση του κριτηρίου ‘modularity’ με διάφορες τεχνικές ομαδοποίησης. Το modularity για μια κοινότητα ορίζεται ως ένα κλάσμα των ακμών μέσα σε μία συστάδα μείον τον κλάσμα των αναμενόμενων ακμών αν οι ακμές είχαν τοποθετηθεί τυχαία. Παρόλα αυτά, το κριτήριο modularity, έχει αρκετά μειονεκτήματα όπως για παράδειγμα η ανικανότητά του να ανιχνεύσει μικρές σε μέγεθος κοινότητες.

Σε αυτήν την εργασία, προτείνουμε ένα καινούρια κριτήριο διαμέρισης, εν ονόματι ‘inclusion’. Αυτό το κριτήριο εκτιμάει πόσο καλά ένα κόμβος ‘συμπεριλαμβάνεται’ στην κοινότητα του εξετάζοντας την ύπαρξη ακμών αλλά και την μη-ύπαρξη ακμών. Έχουμε υλοποιήσει αρκετές τεχνικές για την βελτιστοποίηση του κριτηρίου. Η πρώτη τεχνική ακολουθεί την agglomerative λογική , καθώς ξεκινάει τοποθετώντας κάθε κόμβο σε ξεχωριστή κοινότητα και έπειτα συνενώνει κοινότητες έτσι ώστε να βελτιωθεί το inclusion. Η επόμενη τεχνική που υλοποιήσαμε, έχει παρόμοια αρχική κατάσταση με την πρώτη, αλλά αντί να συνενώνει κοινότητες, μετακινεί ένα κόμβο κάθε φορά σε άλλη κοινότητα. Μία άλλη τεχνική, ήταν να αξιολογήσουμε τις λύσεις που παρήγαγε ο αλγόριθμος του spectral clustering. Στην πειραματική αξιολόγηση που κάναμε, τα αποτελέσματα έδειξαν πως το κριτήριο inclusion είναι αρκετά αποδοτικό στον

εντοπισμό κοινοτήτων και συνήθως οδηγεί σε καλύτερες λύσεις του προβλήματος χωρίς να χρειάζεται να προσδιορίσουμε τον αριθμό των κοινοτήτων εκ των προτέρων.

CHAPTER 1.

INTRODUCTION

1.1 Introduction

Graph theory is the study of graphs, which are mathematical structures used to model pairwise relations between objects. A graph is made up of nodes and edges. The origins of graph theory dates back to 1736, where Euler proposed a solution for the puzzle of Königsberg's bridges [Eule36]. Since then, we have learned a lot about graphs and their mathematical properties [Boll98].

Through the years, graph models became an extremely useful representation of a wide variety of systems in different scientific areas. Social, Biological and Telecommunication are some of the networks that have been studied as graphs and helped researchers extract some valuable features for these systems. For example, social network analysis started in the 1930's and since then, it has become one of the most important topics in sociology [WaFa94].

Due to the fact that we are living in the computer revolution era, scientists and researchers are provided with a huge amount of data, as well as computational resources. Those enormous data, can lead to graph models with millions or even billions of nodes and edges. So, the need of analysis rose, to determine helpful insights about the data.

Graphs that represent real systems are not always regular, meaning that each node does not have the same number of neighbors. The first attempt on modeling those graphs was introduced by Erdös and Rényi [ErRé59]. In their method, the probability of having an edge between a pair of nodes is equal for all possible pairs. Their model is quite simple and powerful with many applications. Although, their model does not have two important properties of real-world networks. Triadic closure, which is the property among three nodes

A, B, and C, such that if a strong relationship exists between A and B as well as A and C, there is a weak or strong relationship between B and C. The second property not present in Erdös–Rényi model, is the power-law distribution on the nodes degree which is commonly observed in real-world networks. Erdös–Rényi graphs converge to a Poison distribution. To tackle the aforementioned problems, Watts and Strogatz proposed a different approach based on interpolation between an Erdös–Rényi graph and a regular ring lattice [WaSt98].

In a random graph, the degree distribution is highly homogenous, which can lead to a problem, as some of the real networks follow a power law distribution as previously mentioned. Furthermore, real networks show high edge concentration within some group of nodes and low edge concentration between those groups. That property is called clustering or community structure [GiNe02].

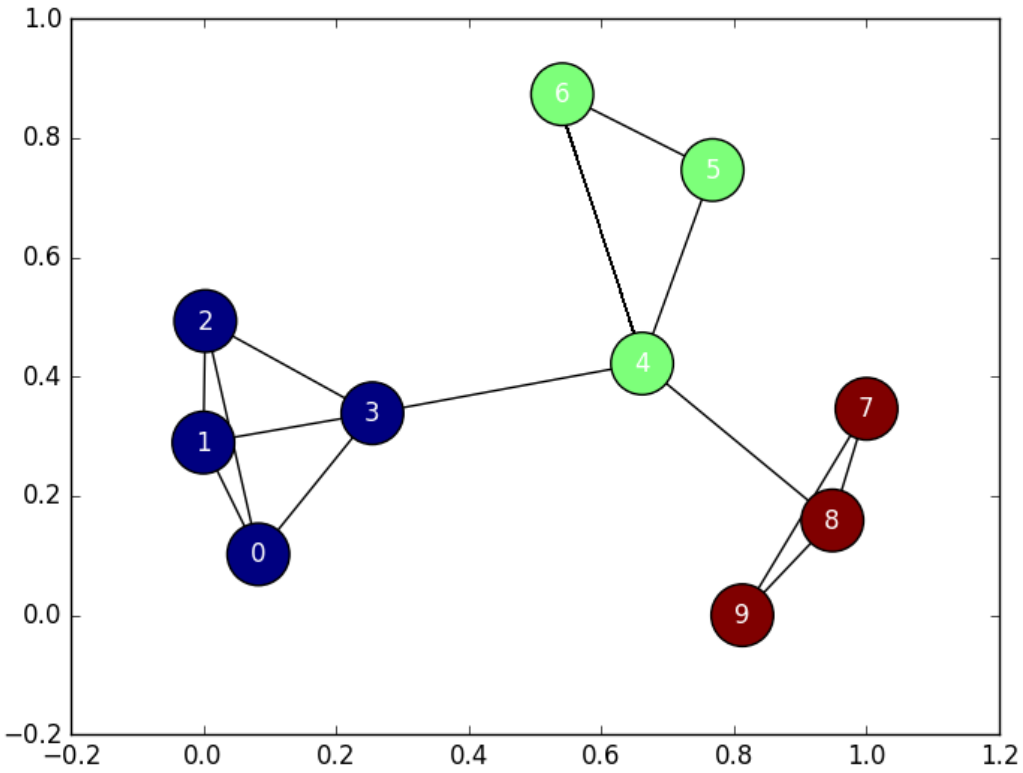

Figure 1 Simple graph clustering.

The need to create communities, is in the human nature. From families and friendship circles to alliances between countries at times of war. So, the need for community study inevitable rose. Communities also appear in many networked systems like computer science, economics, politics, biology, etc. For instance, in World Wide Web there are corresponding

group of pages that may deal with the same or related topic i.e. American Presidential Election.

Finding those aforementioned communities, has many applications. Identifying clusters of clients based on their previous purchases, improves significantly the recommendation system, which in return increases business opportunities. Another application is in parallel computing. For example, it is critical to allocate group of tasks to different processors in order to minimize communication between them and thus enabling rapid performance. The mathematical formalization of this problem falls under the category of NP-hard problems.

1.2 Contribution and Roadmap

In this thesis, we study the community detection problem as well as algorithmic techniques that try to approach it. More specifically, in Chapter 2, we present widely used algorithms such as Hierarchical clustering, Partitional clustering, Statistical models, etc. Finally, we thoroughly present a state-of-the-art quality function, called modularity.

In Chapter 3, which is the thesis contribution, we present a new quality measure, called inclusion. Our quality measure can be considered as a multi-criteria score function, since it focuses both on groups inter and intra edge density. Furthermore, we present two optimization techniques for our inclusion criterion. The first one follows the agglomerative principles, as it starts with every node in a separate cluster and iteratively merges clusters in a greedy way, that best improve the inclusion criterion. The second technique has the same initialization process as the first one, but instead of cluster merging, it moves a single node at a time to a new cluster that yielded the best value of inclusion.

Chapter 4 is dedicated on the experimental comparison between inclusion and modularity measures. It contains results from both of the aforementioned criteria on various synthetic graphs with different properties and sizes. Moreover, we also tested both criteria on real-world datasets such as Zachary's Karate Club and American College Football.

Finally, Chapter 5 summarizes the results which indicate that the inclusion measure is very effective in evaluating communities and usually leads to improved community detection results. Furthermore, we provide some future work/open issues regarding the exploitation of the inclusion criterion.

CHAPTER 2.

RELATED WORK

2.1 Basics

The problem of graph clustering, has a major semantic problem thus making it actually not well defined. The main reason behind that problem is the definition of community itself. There are many suggestions on the definition of a community, but scientists tend to always disagree which led to a rich literature regarding this problem.

It is important to stress that the identification of communal structure is possible when the graphs are sparse enough. That means that the number of edges m is of the order of the number of nodes n of the graph. Otherwise, the distribution of edges is too homogeneous for communities to make sense.

In this section, we will present many algorithms for graph clustering, but before that some basic definitions are essential:

A graph G with nodes $|V| = n$ and edges undirected and unweighted $|E| = m$ i.e. $e_{ij} = e_{ji}$, $e_{ij} \in \{0,1\}$.

2.2 Graph Partitioning

Graph partitioning is the problem of dividing the nodes of a graph in k groups, such that the number of edges between the groups is as small as possible. The number of edges between clusters is called cut size. The next figure is from Fortunato's survey on community detection in graphs and shows a graph with 14 vertices and 2 clusters [Fort10].

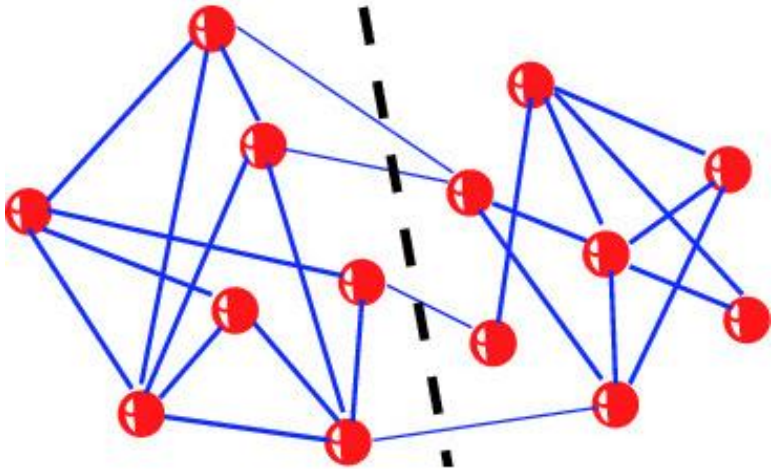

Figure 2 Graph Partitioning Example |V| = 14, number of clusters 2.

One major problem with the graph partitioning is that you need to specify the number of groups. If one does not specify the number of clusters, then the problem becomes quite trivial in the sense that you can group all nodes in one big cluster which will minimize the cut size. This problem can be actually avoided by choosing a different measure to optimize for the partitioning, which accounts for the size of the clusters as well. Specifying the size is also necessary, as otherwise the most likely solution of the problem would be a two-way partition where the lowest degree node will be in one cluster and all the other nodes in another. But this case is also quite simple and uninteresting.

Most variants of the graph partitioning problem are NP-hard. However, there are several algorithms that can produce some heuristic solutions with good results Many of those algorithms perform a bisection of the graph. To achieve further partitioning into more than two clusters, the technique of iterative bisectioning is used. Furthermore, in most cases there is a constraint that suggests that all clusters are of equal size. This problem is known as the minimum bisection and is NP-hard.

One of the first proposed algorithms that is still widely used is the Kernighan-Lin algorithm [KeLi70]. The motivation behind this algorithm was the partitioning of electronic circuits

onto boards. More specifically, the nodes contained in different boards need to be linked to each other with the minimum number of connections. The first thing they did was to define a benefit function Q. That function, quantifies the difference between the number of edges inside the modules and the number of edges lying between them. Then they tried to optimize it as follows: The initialization was the partition of the graph into two clusters of predefined size. This partition can be either random or suggested by some information regarding the graph. Then, subsets of equal numbers of nodes are swapped between the two clusters, so that maximum Q increase is achieved. To reduce the risk of Q's local maxima, the process may include some transitions that reduce the Q value. After a series of swaps with positive and negative Q values, the partition with the largest value of Q is selected and used as starting point of a new series of iterations ($O(n^2 \log(n))$).

As far as the complexity of the algorithm, the Kernighan-Lin algorithm is considered quite fast if you use a constant number of swaps at each iteration. The solution is heavily dependent on the initial configuration, thus is most commonly used to improve the partitions found by other techniques.

Algorithms for graph partitioning are not very suitable for community detection. That is because it is necessary to provide the number of clusters and in some cases even their sizes. Instead, it is preferable to have an algorithm to be capable of providing this kind of information as its output.

2.3 Hierarchical Clustering

Community structure of a graph, is an uncharted territory in general. It is most unlikely to know the number of clusters or any information indicating connections between nodes beforehand. In cases like that, which are the most common ones, graph partitioning algorithms cannot be helpful.

In order to handle those cases, one must make some reasonable assumptions about the clusters structure. One major assumption is that the graph may have a hierarchical structure. For instance, a graph may display levels of grouping of nodes, with small clusters inside larger ones, which in respect they are included within even larger clusters.

In cases like the aforementioned ones, we can use any of the hierarchical clustering algorithms, which reveal the multilevel structure of the graph. Hierarchical graph clustering, has been successfully used in several areas such as: Biology, Marketing, Social Network analysis etc.

The first step of every hierarchical clustering algorithm is the definition of the similarity function. Cosine, Euclidean and Manhattan are some of the most commonly used similarity functions. After the function is well defined, the next step is to compute the pairwise similarity between all n nodes. This step will result in a $n \times n$ matrix S, also known as the similarity matrix.

This kind of clustering techniques aim at finding groups of nodes with high similarity, and are generally distinguished into two categories:

1. Agglomerative algorithms, in which clusters are merged iteratively if their similarity is high enough.

2. Divisive algorithms, in which clusters are split iteratively by removing edges connecting nodes with low similarity.

These two categories reflect on opposite processes: agglomerative algorithms are bottom-up, as the process starts from the nodes in separate clusters and ends up with the graph as a unique cluster. On the other hand, divisive algorithms are top-down as they follow the opposite direction. They begin with all the nodes in one big cluster and they end up in a graph with several clusters.

Since the clusters are merged based on their mutual similarity, the number and quality of clusters is highly dependent on the nature of the similarity function. In agglomerative techniques such as single linkage clustering, the similarity between two groups C_1 and C_2 is the defined as minimum S_{ij} where $i \in C_1, j \in C_2$. This leads to iteratively combining two clusters that contain the closest pair of elements not yet belonging to the same cluster as each other. One major problem with this algorithm is that it usually produces long thin clusters in which nearby elements of the same cluster have small distances, but elements at opposite ends of a cluster may be more distant from each other than to elements of other clusters.

Another algorithm, is the complete linkage clustering. In this technique, the similarity of two clusters C_1 and C_2 is the similarity of their most dissimilar members, meaning the maximum S_{ij} where $i \in C_1, j \in C_2$. This is equivalent to choosing the cluster pair whose merge has the smallest possible diameter. Complete linkage clustering has its drawbacks as well. More specifically, a single node far from the center can increase the diameters of candidate merge clusters dramatically and result in completely changing the final clustering.

Finally, in average linkage clustering, the distance between two clusters is defined as the average of distances between all pairs of nodes, where each pair is made up of one node from each group. At each step, the algorithm merges the clusters with the minimum average value. As it can be observed, this algorithm lies in between single and complete linkage clustering, as it shares both their advantages and drawbacks.

One of the advantages of hierarchical clustering, is that if the number of data is fairly small, the clusters can be easily visualized by dendrograms. The figure below represents a simple example of hierarchical clustering solution illustrated by dendrograms.

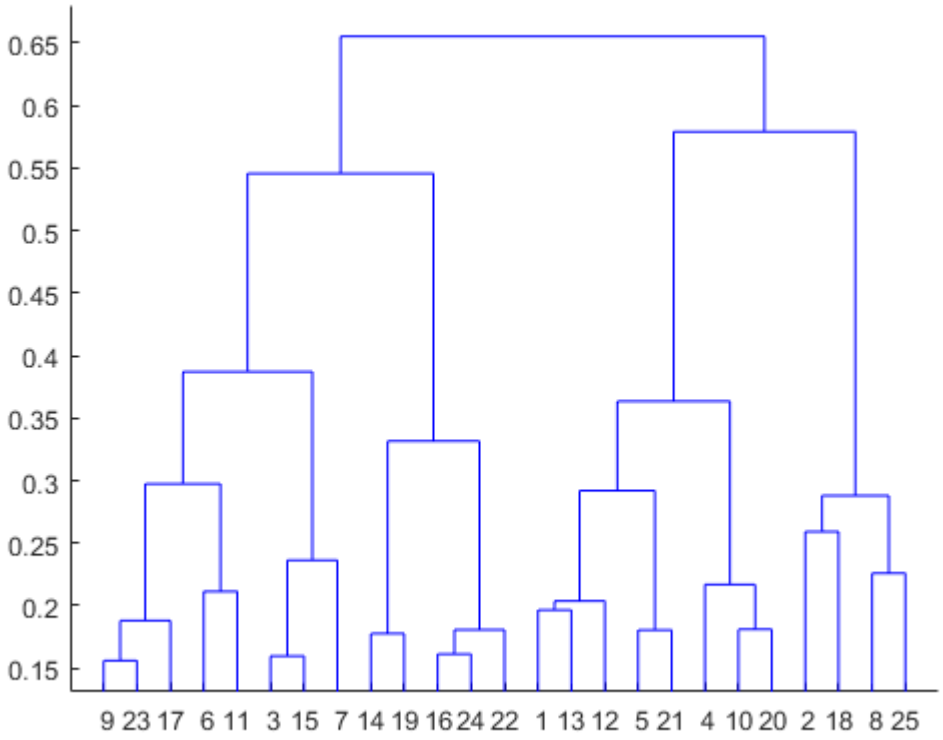

Figure 3 Hierarchical Clustering illustrated with dendrograms.

Hierarchical clustering does not avoid the problem with the number of clusters. It simply constructs the tree spanning over all samples and let you manually chose the "right" number of clusters. However, the results highly depend on the adopted similarity function. Moreover, this technique constructs a partition of clusters assuming hierarchical structure, which in some cases the graph does not have. Another problem is that nodes with only one other node as neighbor are often classified in a separate cluster, which makes no sense. Finally, another major problem is the scalability of the agglomerative clustering. For example, the computational complexity of single linkage is $O(n^2)$ and $O(n^2 \log n)$ for complete and average linkage.

2.4 Partitional Clustering

Another popular class of methodologies is the partitional clustering. In this class, the number of clusters is pre-required. The data points are usually fixed in a metric space, so that each node is a point and a distance measure is defined between pairs of points in the space. The distance is usually a function of dissimilarity between nodes. The goal is to separate the points in k (number of clusters given) clusters in order to maximize/minimize a given cost function based on distances between points. Some of the most commonly used functions are listed below:

- Minimum k-clustering: The cost function here is the diameter of a cluster, which is the largest distance between two points of a cluster. The classification of the points is done in a way that the largest diameter of the k cluster diameters is minimized. The key idea behind this function is that the clusters are tightly compact.
- Average k-clustering: It is similar to the minimum k-clustering, but the diameter is replaced by the average distance between all pairs of points of a cluster.
- K-center: For every cluster, a representative is selected (centroid). Then the maximum distance d_i of the distances of each cluster point from the centroid is computed. The cluster and centroids are changing to minimize the largest value of d_i.
- K-median. Same as k-center, but the maximum distance from the centroid is replaced by the average distance.

One of the most popular techniques, if not the most popular, is the k-means clustering algorithm [MacQ67]. The cost function of k-means is the squared error function:

$$\sum_{i=1}^{k} \sum_{x_j \in C_i} \|x_j - m_i\|^2$$

Where C_i is the subset of points of the i_{th} cluster and m_i is its centroid. The steps of the algorithm are shown below

Algorithm 2.3.1 K-means Algorithm

1. Select K points as the initial centroids

2. **repeat**

3. Form K clusters by assigning each point to its closest centroid

4. Recompute the centroid for each cluster

5. **until** centroids do not change

The results of each K-means run may vary a lot, and that is because it is highly dependent on the initial positions of the centroids. To increase the performance of the algorithm, you can run multiple randomly initialized instances of the algorithm and keep the one that yields the minimum squared error.

Another major technique in clustering is spectral clustering [NgJW01]. The main characteristic of all spectral clustering techniques is that they partition the graph using the eigenvectors of similarity matrices. More specifically, the objects could be points in any metric space. Spectral clustering consists of a transformation of the initial set of objects into a set of points in space, whose coordinates are elements of eigenvectors. Then the set of points is clustered with standard techniques, usually k-means.

The reason that we do not apply k-mean directly, is that the change of representation induced by the eigenvectors makes the cluster properties much clearer. In this way, spectral clustering is able to separate data points that could not be resolved by applying directly k-means clustering. Finally, Spectral Clustering algorithm needs the similarity matrix S as its input, which as mentioned in the previous section, it contains all the pairwise similarities between all n nodes. If the S matrix is not given, the algorithm must calculate it first, using some similarity function (e.g. RBF)

Algorithm 2.3.2 Spectral Clustering Algorithm

Input: Similarity Matrix, number of clusters k

1. Compute the first k eigenvectors $v_1, \dots, v_k$ of the matrix S

2. Build matrix $V \in R^{n\,x\,k}$ with the eigenvectors as columns

3. Interpret the rows of V as new data points Z_i

4. Cluster the points Z_i with the k-means algorithm

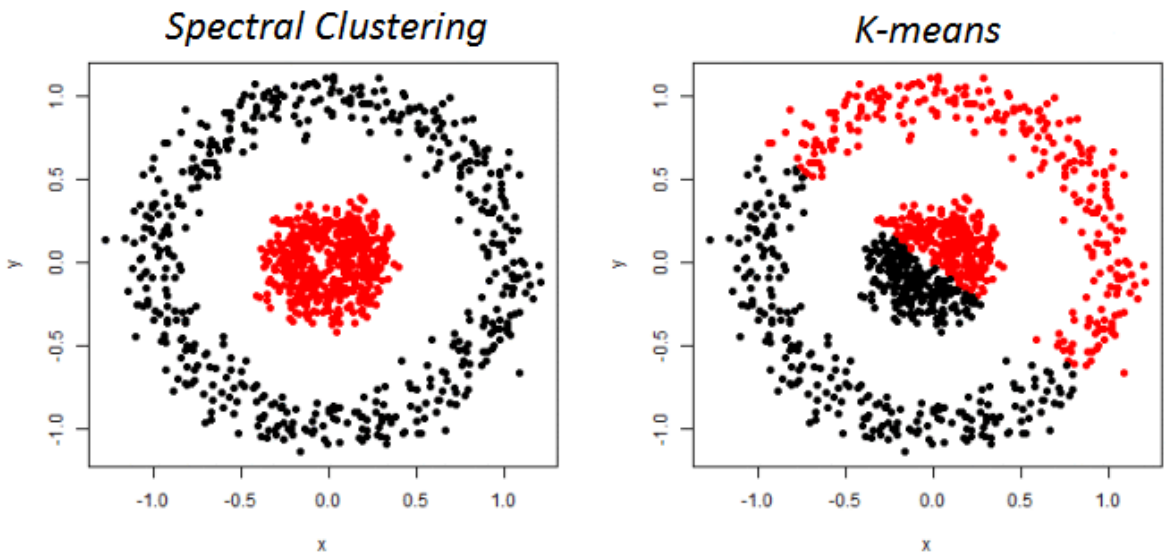

Figure 4 Spectral Clustering vs K-Means.

2.5 Methods Based on Statistical Inference

The statistical inference in general is the process of extracting properties of data sets, starting from a set of examples and model hypotheses [Mack03]. If the data are in a form of a graph, the model, which is based on the hypotheses on how nodes are connected to each other, has to fit the actual graph topology. In this section, we will focus specific on the Bayesian inference [Wink03].

Bayesian inference is a specific case of statistical inference where the Bayes' theorem is used update the estimation the probability that a given hypothesis is true, as more examples become available. It consists of two main characteristics: a statistical model with parameters $\{\theta\}$ and the examples which are expressed by system's information D. Bayesian inference starts by writing the likelihood $P(D|\{\Theta\})$ that the observed examples are produced by the model for a given set of parameters $\{\theta\}$. The aim is to determine the choice of $\{\theta\}$ that maximizes the posterior distribution $P(\{\Theta\}|D)$ of the given parameters of the model and the examples provided. With the use of Bayes' theorem, one has

$$P(\{\Theta\}|D) = \frac{1}{Z}P(D|\{\Theta\})P(\{\Theta\}),$$

where $P(\{\Theta\})$ is the prior distribution of the model parameters and Z is a normalizing constant

$$Z = \int P(D|\{\Theta\})P(\{\Theta\})\, d\theta$$

Unfortunately, computing the integral above is a major challenge, plus, the choice of the prior distribution $P(\{\Theta\})$ is not quite obvious. Different generative models distinguish themselves from each other by the choice of the model and the way they address the aforementioned issues.

Bayesian inference is frequently used in the analysis of real graphs, including social networks [Hart07]. One major part of this analysis is graph clustering. In graph clustering, the examples are represented by the graph structure in either adjacency or weight matrix form. One additional information, that one wants to extract is the partition of the nodes into groups, which is missing. Along with this information, the parameters of the model which is supposed to be responsible for the partition is required as well.

This idea is at the basis of several recent papers, which we discuss here. In all these works, one essentially maximizes the likelihood $P(D|\{\Theta\})$ that the model is consistent with the observed graph structure, with different constraints. We specify the set of parameters $\{\Theta\}$ as the triplet $(\{q\},\{\pi\},k)$ where $\{q\}$ indicates the community assignment of the nodes, $\{\pi\}$ the model parameters, and k the number of clusters.

In 2006, Hastings [Hast06] tried to approach the community detection problem as an inference problem: n nodes are assigned to q clusters with the following requirements: nodes of the same cluster are connected with an edge with a probability p_{in}, while nodes of different clusters are connected with an edge with a probability p_{out}. If $p_{in} > p_{out}$ the model shows community structure. The probability of a node correctly assigned to a cluster is given by the following model:

$$p(\{q_i\}) = cexp[\sum_{<i,j>} J\delta_{q_i q_j}] \, exp[\sum_{i \neq j} \frac{J'\delta_{q_i q_j}}{2}]$$

Where $\sum_{<i,j>}$ denotes a sum over pairs of i, j connected by an edge in the graph, δ is the delta function, with $\delta_{q_i q_j}$ is 1 if q_i, q_j are in the same community and 0 otherwise and

$$c = \exp[\frac{\log(1 - p_{out})\, N(N-1)}{2}]\exp[\sum_{<i,j>} \log(p_{out}/(1 - p_{out})]$$

$$\mathrm{J} = \log[(p_{in}(1 - \mathrm{p}_{out}))/((1 - p_{in})\mathrm{p}_{out})]$$

$$J' = \log[(1 - p_{in})/(1 - \mathrm{p}_{out})]$$

The above equations present the probability as a Potts model problem with combined short and long-range interactions, with coupling constants J, J'. So, the problem of community detection is reduced to finding the ground state of this Potts model. Hastings used belief propagation [Gall63] to find the ground state of the spin model.

The complexity of the model on sparse graphs, is expected to be $O(nlog^{\alpha}(n))$, where α needs to be estimated numerically. Also, in order for the model to work, one has to specify the p_{in}, p_{out}. However, these parameters turn out that they can be chosen arbitrarily, as any bad choices can be recognized and corrected.

2.6 Divisive Algorithms

The main idea behind divisive algorithms is to detect the edges that connect nodes of different clusters and remove them, so that the clusters get disconnected from each other. The most critical part of any divisive algorithm, is finding the property of edges between clusters that will lead to their identification.

Divisive algorithms are pretty similar to the traditional hierarchical top down clustering. The main difference is that the divisive algorithms remove inter-cluster edges instead of edges between pairs of nodes with low similarity. That process does not provide assurance that the edges removed connected nodes with low similarity.

Probably one of the most popular algorithm, if not the most one, was proposed by Girvan and Newman [GiNe02, NeGi04]. In their approach, edges are selected according to the values of measures of edge "betweenness", which is defined of each edge as the number of shortest paths between all pair of nodes that run along the edge. The steps of the algorithm are the following:

1. Calculate the "betweenness" for all edges in the graph.

2. Remove the edge with the highest betweenness.

3. Recalculate betweennesses for all edges affected by the removal.

4. Repeat from step 2 until no edges remain.

They considered three alternative definitions: geodesic edge betweenness, random-walk edge betweenness and current-flow edge betweenness but calculating edge betweenness was by far the fastest one ($O(n^2)$ vs $O(n^3)$ on sparse graphs). Moreover, in real-world applications the edge betweenness gives better results than adopting the other centrality measures.

A major problem they came across on their original work was that they had to deal with the whole hierarchy of the partitions, as they had not found a way to choose the best partition. On a later work [NeGi04], they introduced the modularity criterion, which sparked another kind of clustering technique that we discuss in the following section.

2.7 Quality Measures

2.7.1 General Methodology

As we have already mentioned in the previous chapter, a cluster or community is typically considered as a group of nodes with high edge connectivity among its members than with the nodes of different communities/clusters. The general methodology when trying to detect communities via quality measures, is usually following the next two major steps in their approach:

- Define a quality measure (objective function), that captures the definition of community structure in a way nodes in the same group have better internal than external connectivity.
- Use algorithmic techniques, so that the nodes of the network are assigned to specific communities, through optimization of the objective function.

In many cases, the optimization of the objective function leads to computational difficult problems. So, a common approach is to employ some kind of heuristic algorithms or other approximation techniques.

Some of the quality measures focus on both the intra as well as the inter cluster edge density (multi-criterion scores). Another kind of measures is the single-criterion score, where the measure focuses only in one of them (either inter or intra). An excellent example of that kind of measure is modularity, which we will get into detail in the next section.

2.7.2 Modularity Measure

Newman and Girvan introduced modularity Q as a stopping criterion for one of their previous algorithms [NeGi04]. Since then, modularity became one of the most popular and widely used measures to evaluate the quality of the graph partition. It is a classic example of one of the first attempts to achieve a better understanding of the community detection problem, as it presents key elements such as the definition of a community as well as its strength.

The main idea of modularity is that given a specific partition of a graph, it measures the number of edges that exist within a cluster compared to the expected number of edges of a random graph with the same degree distribution.

In other words, modularity is taking advantage of the fact that a random graph is not expected to have inherent community structure. So, comparing the density of a subgraph with the expected one of the same subgraph in a random graph, will determine a method for identifying clusters. More specifically, the modularity value Q is defined below:

$$Q = \frac{1}{2m}\sum_{i,j}\left[e_{ij} - \frac{k_i k_j}{2m}\right]\delta(c_i, c_j)$$

Where

i, j are the graph nodes

e_{ij} represents the weight of the edge between i and j

k_i is the sum of the weights of the edges attached to vertex i

m is the sum of all of the edge weights in the graph

c_i and c_j are the communities of nodes i, j respectively

δ is delta function, with $\delta(u, v)$ is 1 if $u = v$ and 0 otherwise

The assumption made by one, is that high values of modularity indicate good partitions. This indicates that the partition that corresponds to the maximum value of modularity is a very good one if not the best. The optimization of modularity Q through exhaustive search is not feasible due to the huge number of possible ways to partition a graph. In 2006, it has been proved that optimizing modularity is a NP-complete problem [BDG+06]. However, there are several algorithms able to find good approximations of the modularity maximum in a reasonable time.

The first algorithm to maximize modularity was a greedy agglomerative clustering method proposed by Newman himself [Newm04]. The algorithm starts by assigning n nodes to n different clusters, each containing exactly one node. The edges are not available all at once, as they are added one by one during the algorithm. However, the Q value of partitions is calculated with the full topology of the graph.

Adding the first edge to the set of disconnected nodes reduces the number of clusters from n to $n - 1$, creating a new partition of the graph. The edge is chosen in order for the partition to achieve the maximum increase of modularity. This process is repeated for all other edges. The number of partitions found during the procedure is n, each with a different number of clusters, from n to 1. The largest modularity value from those subsets is the solution given by the algorithm.

As for the complexity of the algorithm, at each iteration step, one needs to compute the difference ΔQ produced by the merging of any two communities of the current partition, and choose the best merge. An interesting thing is that, merging communities with no edges between them, can never lead to an increase of Q, so there is no need to check all the available communities, only the connected ones which are at most m. Since the calculation of each ΔQ can be done in constant time, this part of the calculation requires a time $O(m)$. After choosing the communities merging pair, the edge matrix update which expressing the number of edges between clusters i and j of the running partition can be done in $O(n)$ at worst-case. Since the algorithm requires $n - 1$ iterations to run to completion in order to merge all communities, its complexity is $O((m + n)n)$.

Clauset, Newman and Moore [ClNM04] in 2004 pointed out that a large amount of operations regarding the update of edge matrix, where redundant. They proposed a max-heap data structure to perform this operation, which stores the data in a binary tree form. They maintained the matrix of modularity ΔQ, in a max-heap containing the largest elements of each row as well as the corresponding communities. The optimization process is done in the same way as before, but much faster due to the new data structures.

The complexity of the algorithm is $O(md(log(n)))$, where d is the depth of the max-heap, which grows up to $log(n)$ for graphs with a strong hierarchical structure. This algorithm is still used to estimate the modularity maximum on such large graphs.

Finally, in 2008, Blondel, Guillaume, Lambiotte and Lefebvre [BGLL08] proposed a simple heuristic algorithm that outperformed the previous methods in terms of computational time and at some times in the achieved modularity value as well. This method is also known as

the Louvain method. In order to maximize the modularity value, the Louvain algorithm consisted of two steps that are repeated iteratively:

At first, each node in the graph is assigned to its own community. Then for each node i, the change in modularity is calculated by removing i from its own community and moving it into all the possible communities that the neighbors of i belong.

To avoid the intense computations of moving each node to different communities and then calculate modularity from the start, they came up with this function that yield the modularity change:

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

$\sum_{in}$ is the sum of all the weights of the links inside the community i is moving into

$\sum_{tot}$ is the sum of all the weights of the links to nodes in the community

k_i is the weighted degree of i

$k_{i,in}$ is the sum of the weights of the links between i and other nodes in the community

m is the sum of the weights of all links in the graph.

Once this value is calculated for all possible communities that i is connected to, i is placed into the community that yields the greatest modularity increase. If there is no possible increase, then i stays in its initial community. This process is applied repeatedly and sequentially to all nodes until no modularity increase can be achieved. After the local modularity maximum is hit, this phase has ended.

In the second phase, the algorithm groups all the nodes in the same communities and builds a new graph. Nodes are the communities from the first phase. The links between nodes in the same group, represented as self-loops on the new merged community. Also, links from multiple nodes in the same community to a node in a different community are represented by weighted edges between nodes. Once the new graph is constructed, phase one can be re-applied to it. This phase is optional and usually omitted.

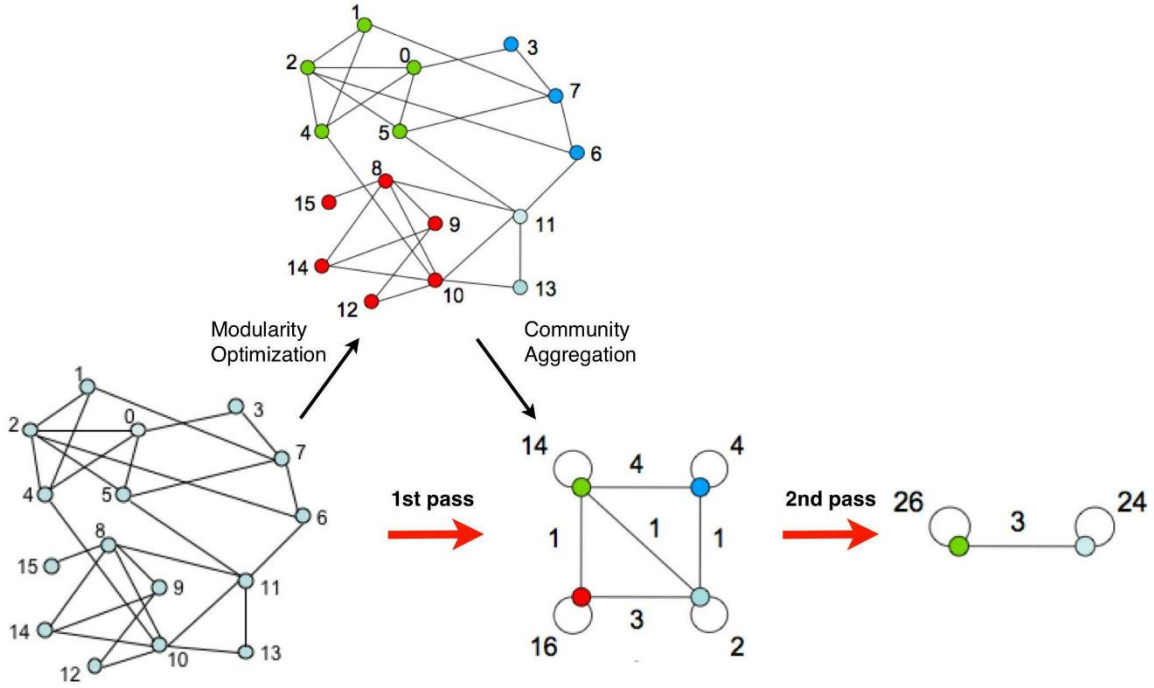

Figure 5 Visualizations of the steps used by Louvain's method [BGLL08].

Another technique for optimizing modularity is via simulated annealing [KiGV83]. Simulated annealing is used in many different problems and simply performs an exploration of all possible states, while trying to achieve global optimum of a given function F. The probability of a transition from one state to another is 1 if the function F increases and $e^{\frac{\Delta F}{T}}$ otherwise, where ΔF is the function decrease and T is the temperature which decreases over time. At some point, the system converges to a stable state, which can be a good approximation of the maximum of F.

Guimerà, Pardo and Amaral [GuSA04] were the first ones to use simulated annealing as an optimization technique for modularity. In his implementation two types of moves are used. The first one is local move where a single node is moved to another cluster at random. The second one is global move which contains communities' splits and merges. The split move is implemented in order to reduce the risk of trapping in local minima. The simulated annealing method can be potentially equal to the true modularity maximum, but it is very slow. The true complexity cannot be estimated, due to the heavy dependence on the parameters chosen for the optimization such as initial temperature and cooling factor. Simulated annealing is usually used for small graphs.

Although optimizing modularity has many advantages compared to other methods, it has some limitations as well. As noted by Fortunato and Barthélemy [FoBa07], modularity suffers from the resolution limit. More precisely, modularity optimization might fail to detect clusters smaller than a scale number, which is mainly dependent on the graph size. This limitation is important because real world networks, often contain communities of various sizes.

CHAPTER 3.

INCLUSION QUALITY MEASURE (I)

3.1 Introducing the New Quality Measure (I)

As we have presented previously, the modularity measure has many advantages but a major disadvantage as well. With our new quality measure named Inclusion, that we will present in detail below, we tried to approach the community detection problem from another perspective. Assume a graph G with nodes $|V| = n$ and edges undirected and unweighted $|E| = m$ i.e. $e_{ij} = e_{ji},\ e_{ij} \in \{0,1\}$

The first main difference of our measure compared to the existing ones, is that we value the absence of edges between two different clusters. The reason behind this idea is that for a cluster to be compact, the number of edges from/to different clusters should be minimum. The second and probably the most important difference, is that our measure focuses primarily on nodes and not on clusters. With this approach, we believe that we will be able to predict the number of main communities, but also discover the communities that are small compared to the size of the whole graph.

Given a community structure $C = \{C_1, C_2, \dots, C_k\}$, the definition for the Inclusion model is presented below:

$$I_i = \frac{1}{2}\left[\frac{W_1^i(in)}{d_i} + \left[\frac{W_0^i(out)+1}{N-d_i}\right]\right] \quad I_i \in [0.5, 1]$$

where

$W_1^i(in)$, is the number of existing edges between the corresponding node and the nodes of the same cluster

$W_0^i(out)$, is the number of non-existing edges between the corresponding node and nodes belonging in different clusters

d_i, is the degree of node i

N, is the total number of nodes

In other words, inclusion measures for each node, its existing edges inside its community and non-existing edges with the other communities. With the presented formula, it is now easier to understand that our criterion is node-centric and not cluster centric. Another property of our criterion is that it is in fact a multi-criterion score function as it focuses both on inter as well as intra edge density. The following formula expresses the inclusion measure on the whole graph:

$$I = \frac{\sum_{i=1}^{N} I_i}{N} \qquad I \in [0.5\ ,1]$$

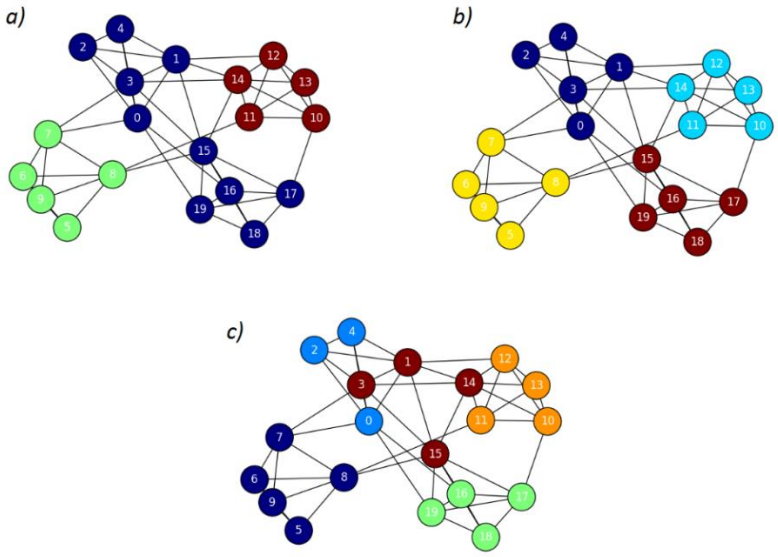

Figure 6 a) Graph clustered into three communities, I = 0.85 b) Graph clustered into four communities, I = 0.89 c) Graph clustered into five communities, I = 0.80.

As you can easily observe from Figure 6, I tends to increase as the quality of the clusters increases. In the first case, the graph is separated in three clusters which is a visually fine solution. That partition has I = 0.85. For the second case, the graph is separated in four clusters, which is the visually the best one. That partition has I = 0.89. In the third case, graph is separated in five clusters, which seems kind of over-partitioning it. That results in I = 0.80. The visual results from the three cases, align with the I values of each graph partition.

To examine the extreme cases where all nodes into one cluster or every node on a separate cluster, assume a fully connected graph, where all nodes are connected to each other. In the case where all nodes are in the same cluster, the first part of inclusion measure, $\frac{W_1^i(in)}{d_i}$, is one for every node because in a fully connected graph, the degree of every node is N-1 and the since all the nodes are in the same cluster, the intra-edges are also N-1. For the second part of the inclusion formula, $\frac{W_0^i(out)+1}{N-d_i}$, is also one cause as we already established the degree of every node is N – 1 thus the denominator is N – (N-1) = 1. Moreover, the non-existing edges to other clusters is zero because there is only one cluster thus the numerator is also 1. So, the inclusion value for this case is 1.

In the case of every node belonging in a separate cluster the first part of the inclusion formula, $\frac{W_1^i(in)}{d_i}$, is zero because there are no other nodes in each node's cluster. As for the second part, $\frac{W_0^i(out)+1}{N-d_i}$, is one as it does not differ from the previous case. So, the inclusion value for this case is 0.5.

3.2 Optimizing Inclusion

After explaining how our criterion works, it is only natural to try to optimize our criterion in order to detect the underlying communities. To achieve that, we tried two major techniques which are presented in detail below.

Agglomerative Cluster Merging:

Our first approach on the optimization problem, was based on cluster merging in each step. More precisely, at the start of the algorithm, each node is in a separate cluster containing only the node itself. Then for every possible cluster merge, we calculate all the corresponding I values and store the cluster pair that yielded the maximum I. Afterwards, we merge the aforementioned clusters and repeat the second step for finding the max I. Our stopping criterion was the improvement of I. If there was no pair that improves I, then we stop the algorithm.

Algorithm 3.2.1 Agglomerative Cluster Merging

1. C = Set all nodes into separate clusters (Initialization)

2. **repeat**

3. Set $maxI$ to 0

4. **for** cluster c_i in C

5.. **for** cluster c_j in C

6. Calculate corresponding I value

7. Store c_i, c_j if corresponding $I > maxI$

8. Update $maxI = I$

8. Update C by merging the c_i and $c_j$that resulted in $maxI$

9. **until** I does not improve

Our first results on the algorithm were encouraging, but the computational time was huge. That is because for every possible cluster pair we need to computed I from scratch ($O(n^2)$). To tackle this issue, we compiled a delta function to compute this increase without calculating the I from the whole graph. The nodes belonging in same cluster as the node n, who is about to move to a new cluster are labeled as N_{old} as on the other hand the nodes on cluster that the node n is about to be moved to are labeled as N_{new}. The definition for our delta function is presented below:

$$\Delta I_1 = \sum_{i \in N_{old}} \sum_{j \in N_{new}} e_{i,j} \left(\frac{1}{2d_i}\right) - (1 - e_{i,j}) \frac{1}{2(N - d_i)}$$

$$\Delta I_2 = \sum_{i \in N_{New}} \sum_{j \in N_{old}} e_{i,j} \left(\frac{1}{2d_i}\right) - (1 - e_{i,j}) \frac{1}{2(N - d_i)}$$

$$\Delta I = \frac{\Delta I_1 + \Delta I_2}{N}$$

Although, there was a major improvement in computational time, the method was still pretty slow. So, we abandon this technique and start experimenting with a new one presented below.

Greedy Node Movement:

The second approach on the problem, was based on moving nodes between clusters instead of whole clusters. The initiation process was the same, with every node belonging in a separate cluster. Then for every node in our graph we calculate the ΔI value for every possible cluster it can move to and store the maximum ΔI as well as the node and cluster that yielded that ΔI. Then we implement the best transition and start the search for the next move. Our stopping criterion was the same, the lack of improvement of I.

Algorithm 3.2.2 Greedy Node Movement

1. C = Set all nodes into separate clusters (Initialization)
2. $G = (V, E)$
3. **repeat**
4. Set $max\Delta I$ to 0
5. **for** node n in V
6. **for** cluster c in C
7. Calculate corresponding ΔI value
8. Store n, c if corresponding $\Delta I > max\Delta I$
9. Update C by moving node n to cluster c that resulted in $max\Delta I$
10. **until** I does not improve

After running several experiments, we observed that the results were highly dependent on the processing order of the nodes. So, to manage this problem, our first try was to examine nodes based on their degree. We tried to process nodes with descending and ascending order regarding their degree. The results did not differ from a random selection, so we abandon this technique. Furthermore, we slightly change the ΔI function to fit the nature of our method (node-centric). As in our previously presented delta function, the nodes belonging in same cluster as the node i, who is about to move to a new cluster are labeled as N_{old} as on the other hand the nodes on cluster that the node i is about to be moved to are labeled as N_{new}.

$$\Delta I_1 = \sum_{j \neq i, j \in N_{old}} \left\{ (1 - e_{i,j}) \left[\frac{1}{2} \left(\frac{1}{N - d_j} + \frac{1}{N - d_i} \right) \right] - e_{i,j} \left[\frac{1}{2} \left(\frac{1}{d_j} + \frac{1}{d_i} \right) \right] \right\}$$

$$\Delta I_2 = \sum_{j \neq i, j \in N_{new}} \left\{ e_{i,j} \left[\frac{1}{2} \left(\frac{1}{d_j} + \frac{1}{d_i} \right) \right] - (1 - e_{i,j}) \left[\frac{1}{2} \left(\frac{1}{N - d_j} + \frac{1}{N - d_i} \right) \right] \right\}$$

$$\Delta I = \frac{\Delta I_1 + \Delta I_2}{N}$$

Our final approach with this technique, was to consider three major decisions for searching. The first one, was either to search nodes sequentially or in a random order every time. The second decision, rose from the nature of our criterion. Our criterion positively values the existence of edges inside the cluster and the non-existence of edges between different clusters. So, we chose to either search adjacent clusters only, which are clusters that contain at least one neighbor of the respective node processed at the time, or every possible cluster. The last decision was either to examine all the possible clusters and find the best or chose the first better that we found.

Those three decisions, led us to develop eight optimization methods for inclusion that we will compare against modularity on both real datasets as well as artificial ones with certain properties. More specifically, we labeled these methods three parts separated with an underscore. The first part is about node selection. The 'All' tag refers to exhaustive node search as on the other hand, 'Rnd' refers to randomly selecting a node for examination. The second part of the name is about the candidate clusters that a node can move to. The 'Adj' tag means that a node can move to adjacent clusters, which are clusters that contain at least one neighbor of the respective node processed at the time. 'All' tag means that a node can move to every possible cluster. Finally, the third tag refers to choosing either the best of the available clusters ('B' tag) or the first cluster that improves inclusion ('F' tag).

CHAPTER 4.

DATASETS & RESULTS

4.1 Synthetic & Real-World Data

In order to test our data, we used both synthetic and real-world graphs. To produce synthetic graphs, we implemented a function that creates a graph given the following parameters:

N number of nodes

C number of clusters

$Cluster\_size$ list of percentages regarding the number of nodes for each cluster

$External\_probability$ the probability of each node to have an edge with nodes from other clusters

$Probability\_list$ list of probabilities for intra-edges in each cluster

After tweaking those parameters, we created five different categories of synthetic graphs to examine our criteria against modularity. The first sub-category of our synthetic data, consists graphs with equally distributed $Cluster\_size$ $(\frac{1}{C})$ and $Probability\_list$ values ranging from 80 to 100% percent for each cluster. This results in graphs that are separated in a clear way.

The next category contains of graphs with $Probability\_list$ values ranging from 80 to 100% percent for each cluster and descending $Cluster\_size$. This results in graphs with the dense large clusters and sparse small clusters.

The third category contains graphs with equally distributed $Cluster\_size$ $(\frac{1}{c})$ and a descending $Probability\_list$ values which starts ranging from 90 to 100 percent and reduced for each cluster by a constant amount (15%). That category creates equally dense clusters regarding the number of nodes, but addresses the various intra edge density.

The fourth category contains graphs with descending $Probability\_list$ values as described above. Also, the $Cluster\_size$ is descending as well. That parameter tweaking leads to clusters with stable ration between number of nodes and intra edge density.

The final sub-category includes once again graphs with a descending $Probability\_list$ values as described in the two previous categories. However, the $Cluster\_size$ is ascending. That leads to graphs with large sparse clusters and small dense clusters.

Due to the large running time for our optimization techniques on large graphs, on a commercial personal computer, we implemented a different approach. In order to test both criteria on large scale networks, we implement an optimization via the spectral clustering algorithm. More specifically, we run the spectral clustering algorithm for k number of clusters ranging from 2 to 20 for each graph, and store the partition that maximized modularity and inclusion respectively. We did 20 independent runs for each category.

The second category, is actual real-world datasets. To determine which of the real-world datasets to choose, we had two basic criteria. The first one was that the dataset was small enough, so it could be processed in a simple personal PC. The second and the most important one, was that the dataset had the ground truth provided. Ground truth, is the actual partition of the nodes into clusters. That information was necessary to us, because a couple of our metrics were based on it.

The first dataset that we chose was the famous Zachary's Karate Club [Zach77]. This dataset is a social network of a karate club that was studied for three years. During the study, a conflict arose between the administrator and instructor, which led to the split of the club into two.

The second dataset was the American College Football dataset. This dataset is a network of American football games between Division IA colleges during regular season Fall 2000.

4.2 Results

In order to measure the quality of the solution given by inclusion and modularity, we measure the similarity of an obtained solution with the ground truth solution, using two different metrics that are presented below:

Normalized Mutual Information (NMI): Mutual Information score (MI) in general, is the measure of mutual dependence between two random variables, but in our case, is adjusted to measure dependence (similarity) between different partitions instead of random variables. So, NMI, is a normalization of the Mutual Information score (MI), which is a measure of the

mutual dependence between the two random variables, to scale the results between 0 (no mutual information) and 1 (perfect correlation).

Adjusted Rand Index (ARI): The Rand Index (RI) computes a similarity measure between two partitions by considering all pairs of samples and counting pairs that are assigned in the same or different clusters in the predicted and true partitions.

The adjusted Rand index is a normalization of RI that provides a value close to 0.0 for random labeling independently of the number of clusters and samples and exactly 1.0 when the clusters are identical.

Once we have defined the NMI and ARI performance measures, we will explain the contents in the tables that follow. For each category, we created a graph with 60 nodes and 4 clusters as well as a graph with 80 nodes and 5 clusters. For each type of graph, we conducted 100 runs and kept the average values for various metrics.

Starting with the table columns, the first one presents of the average cluster size. The second one gives the average Inclusion value. The third one gives the average Modularity. Finally, the last two correspond to the average NMI and average ARI respectively.

As for the table rows, the first one corresponds to the solution produced by the fast modularity algorithm. The next rows, present the results for our optimization function. Each row name consists of three parts separated with an underscore, which are thoroughly described at the end of chapter 3.

4.2.1 Equal Cluster Size – Large Intra Cluster Probability

Tables 1 and 2 present the results for the case where all clusters are of equal size and have high intra-edge density.

Table 1 Results for Graph Model with 60 Nodes, 4 Clusters, Equally Distributed Cluster Size, External Probability 15% and Probability List ranging from 90% to 100%.

	Avg Clusters	**Avg I**	**Avg Mod**	**Avg NMI**	**Avg ARI**
Modularity	**4**	**0.7672**	**0.2789**	**1**	**1**
All_Adj_B	3.81	0.7613	0.2720	0.9741	0.9435
All_Adj_F	**4**	**0.7672**	**0.2789**	**1**	**1**
All_All_B	3.81	0.7613	0.2720	0.9741	0.9435
All_All_F	**4**	**0.7672**	**0.2789**	**1**	**1**
Rnd_Adj_B	3.97	0.7663	0.2777	0.9960	0.9911

Rnd _Adj_F	3.66	0.7543	0.2645	0.9542	0.9000
Rnd _All_B	3.98	0.7664	0.2779	0.9973	0.9940
Rnd _All_F	3.43	0.7457	0.2545	0.9229	0.8316

Table 2 Results for Graph Model with 80 Nodes, 5 Clusters, Equally Distributed Cluster Size, External Probability 15% and Probability List ranging from 90% to 100%.

	Avg Clusters	**Avg I**	**Avg Mod**	**Avg NMI**	**Avg ARI**
Modularity	4.99	0.7306	0.2568	0.9991	0.9977
All_Adj_B	3.7	0.7069	0.2288	0.8779	0.7294
All_Adj_F	**5**	**0.7307**	**0.2570**	**1**	**1**
All_All_B	3.67	0.7064	0.2282	0.8750	0.7237
All_All_F	4.99	0.7306	0.2568	0.9991	0.9977
Rnd_Adj_B	4.87	0.7284	0.2542	0.9882	0.9711
Rnd _Adj_F	4.14	0.7139	0.2369	0.9212	0.8102
Rnd _All_B	4.9	0.7289	0.2548	0.9910	0.9774
Rnd _All_F	3.73	0.7042	0.2258	0.8817	0.7319

4.2.2 Equal Cluster Size – Variable Intra Cluster Probability

Tables 3 and 4 present the results for the case where all clusters are of equal size and have intra-edge densities that are gradually reduced.

Table 3 Results for Graph Model with 60 Nodes, 4 Clusters, Equally Distributed Cluster Size, External Probability 15% and Probability List ranging from 90% to 100% for the First Cluster Followed by a 15% Reduction for each Subsequent Cluster.

	Avg Clusters	**Avg I**	**Avg Mod**	**Avg NMI**	**Avg ARI**
Modularity	**3.99**	0.7335	**0.2508**	**0.9960**	0.9943

All_Adj_B	3.37	0.7164	0.2318	0.8891	0.7957
All_Adj_F	4.02	0.7335	**0.2508**	0.9954	0.9948
All_All_B	3.37	0.7164	0.2318	0.8891	0.7957
All_All_F	**4.01**	**0.7336**	**0.2508**	0.9955	**0.9951**
Rnd_Adj_B	**3.99**	0.7326	0.2498	0.9904	0.9852
Rnd _Adj_F	3.81	0.7252	0.2401	0.9585	0.9222
Rnd _All_B	3.98	0.7318	0.2488	0.9860	0.9760
Rnd _All_F	3.48	0.7165	0.2303	0.9088	0.8290

Table 4 Results for Graph Model with 80 Nodes, 5 Clusters, Equally Distributed Cluster Size, External Probability 15% and Probability List ranging from 90% to 100% for the First Cluster Followed by a 15% Reduction for each Subsequent Cluster.

	Avg Clusters	**Avg I**	**Avg Mod**	**Avg NMI**	**Avg ARI**
Modularity	4.73	0.6827	0.2124	0.9022	0.8642
All_Adj_B	3.75	0.6687	0.1952	0.7643	0.6293
All_Adj_F	5.03	**0.6840**	**0.2129**	**0.9309**	**0.9128**
All_All_B	3.74	0.6687	0.1951	0.7652	0.6298
All_All_F	**5.02**	0.6839	0.2128	0.9285	0.9084
Rnd_Adj_B	4.79	0.6796	0.2070	0.8674	0.8110
Rnd _Adj_F	4.59	0.6767	0.2034	0.8501	0.7766
Rnd _All_B	4.94	0.6803	0.2077	0.8932	0.8464
Rnd _All_F	4.18	0.6703	0.1945	0.8148	0.7064

4.2.3 Variable Cluster Size – Large Intra Cluster Probability

Tables 5 and 6 present the results for the case where clusters are of various size and have high intra-edge density.

Table 5 Results for Graph Model with 60 Nodes, 4 Clusters, Distributed Cluster Size with Descending Order (40%, 30%, 20%, 10%), External Probability 15% and Probability List ranging from 90% to 100%.

	Avg Clusters	**Avg I**	**Avg Mod**	**Avg NMI**	**Avg ARI**
Modularity	3.48	0.7850	**0.2581**	0.9545	0.9458
All_Adj_B	3.14	0.7797	0.2557	0.9180	0.8979
All_Adj_F	**4**	**0.7883**	0.2575	**0.9983**	**0.9990**
All_All_B	3.14	0.7797	0.2557	0.9180	0.8979
All_All_F	3.99	0.7883	0.2576	0.9976	0.9980
Rnd_Adj_B	3.93	0.7874	0.2572	0.9918	0.9896
Rnd _Adj_F	3.9	0.7877	0.2577	0.9892	0.9884
Rnd _All_B	3.9	0.7869	0.2569	0.9892	0.9856
Rnd _All_F	3.76	0.7855	0.2566	0.9764	0.9681

Table 6 Results for Graph Model with 80 Nodes, 5 Clusters, Distributed Cluster Size with Descending Order (30%, 25%, 20%, 15%, 10%), External Probability 15% and Probability List ranging from 90% to 100%.

	Avg Clusters	**Avg I**	**Avg Mod**	**Avg NMI**	**Avg ARI**
Modularity	4.36	0.7420	**0.2551**	0.9574	0.9327
All_Adj_B	3.36	0.7240	0.2383	0.8504	0.7511
All_Adj_F	4.91	0.7440	0.2548	0.9933	0.9874
All_All_B	3.37	0.7242	0.2383	0.8511	0.7519
All_All_F	**4.92**	**0.7443**	**0.2551**	**0.9947**	**0.9910**
Rnd_Adj_B	4.57	0.7415	0.2533	0.9673	0.9446
Rnd _Adj_F	4.3	0.7352	0.2468	0.9385	0.8831

Rnd _All_B	4.66	0.7419	0.2535	0.9729	0.9544
Rnd _All_F	3.88	0.7267	0.2375	0.8991	0.8082

4.2.4 Variable Cluster Size – Variable Intra Cluster Probability (Small Cluster High Density)

Tables 7 and 8 present the results for the case where clusters are of various size and have intra-edge densities that are gradually reduced, resulting in smaller sized clusters having larger intra-edge density.

Table 7 Results for Graph Model with 60 Nodes, 4 Clusters, Distributed Cluster Size with Ascending Order (10%, 20%, 30%, 40%), External Probability 15% and Probability List ranging from 90% to 100% for the First Cluster Followed by a 15% Reduction for each Subsequent Cluster

	Avg Clusters	**Avg I**	**Avg Mod**	**Avg NMI**	**Avg ARI**
Modularity	3.71	0.7242	**0.2285**	0.9684	0.9628
All_Adj_B	3.09	0.7168	0.2232	0.8930	0.8672
All_Adj_F	**3.96**	**0.7255**	0.2284	**0.9943**	**0.9938**
All_All_B	3.08	0.7166	0.2229	0.8904	0.8641
All_All_F	3.92	0.7254	0.2284	0.9890	0.9880
Rnd_Adj_B	3.78	0.7239	0.2276	0.9681	0.9641
Rnd _Adj_F	3.77	0.7238	0.2275	0.9692	0.9617
Rnd _All_B	3.91	0.7245	0.2276	0.9845	0.9797
Rnd _All_F	3.63	0.7217	0.2257	0.9536	0.9417

Table 8 Results for Graph Model with 80 Nodes, 5 Clusters, Distributed Cluster Size with Ascending Order (10%, 15%, 20%, 25%, 30%), External Probability 15% and Probability List ranging from 90% to 100% for the First Cluster Followed by a 15% Reduction for each Subsequent Cluster

	Avg Clusters	**Avg I**	**Avg Mod**	**Avg NMI**	**Avg ARI**
Modularity	4.56	0.6663	0.1938	0.9225	0.8944
All_Adj_B	3.5	0.6526	0.1765	0.7584	0.6477
All_Adj_F	**5.03**	**0.6678**	**0.1948**	**0.9504**	**0.9395**
All_All_B	3.5	0.6525	0.1764	0.7589	0.6486
All_All_F	4.9	0.6674	0.1945	0.9358	0.9192
Rnd_Adj_B	4.75	0.6637	0.1896	0.8898	0.8435
Rnd _Adj_F	4.52	0.6626	0.1887	0.8766	0.8251
Rnd _All_B	4.59	0.6639	0.1901	0.8845	0.8424
Rnd _All_F	3.99	0.6563	0.1804	0.8084	0.7212

4.2.5 Variable Cluster Size – Variable Intra Cluster Probability (Large Cluster High Density)

Tables 9 and 10 present the results for the case where clusters are of various size and have intra-edge densities that are gradually increased, resulting in larger sized clusters having larger intra-edge density.

Table 9 Results for Graph Model with 60 Nodes, 4 Clusters, Distributed Cluster Size with Descending Order (40%, 30%, 20%, 10%), External Probability 15% and Probability List ranging from 90% to 100% for the First Cluster Followed by a 15% Reduction for each Subsequent Cluster

	Avg Clusters	**Avg I**	**Avg Mod**	**Avg NMI**	**Avg ARI**
Modularity	3.14	0.7686	**0.2447**	0.9069	0.9002
All_Adj_B	3.31	0.7675	0.2429	0.9039	0.8972

All_Adj_F	4.09	0.7724	0.2422	**0.9748**	0.9825
All_All_B	3.31	0.7678	0.2429	0.9078	0.9010
All_All_F	4.06	**0.7725**	0.2422	0.9761	**0.9831**
Rnd_Adj_B	3.96	0.7721	0.2427	0.9641	0.9692
Rnd _Adj_F	3.97	0.7721	0.2425	0.9699	0.9749
Rnd _All_B	**3.98**	0.7719	0.2423	0.9654	0.9704
Rnd _All_F	3.83	0.7705	0.2419	0.9538	0.9536

Table 10 Results for Graph Model with 80 Nodes, 5 Clusters, Distributed Cluster Size with Descending Order (30%, 25%, 20%, 15%, 10%), External Probability 15% and Probability List ranging from 90% to 100% for the First Cluster Followed by a 15% Reduction for each Subsequent Cluster

	Avg Clusters	**Avg I**	**Avg Mod**	**Avg NMI**	**Avg ARI**
Modularity	3.95	0.7181	**0.2338**	0.8795	0.8598
All_Adj_B	4.17	0.7130	0.2272	0.8472	0.8073
All_Adj_F	4.98	**0.7200**	0.2317	**0.9282**	**0.9324**
All_All_B	4.18	0.7132	0.2275	0.8467	0.8087
All_All_F	**4.99**	**0.7200**	0.2317	0.9270	0.9323
Rnd_Adj_B	4.96	0.7192	0.2310	0.9152	0.9170
Rnd _Adj_F	4.71	0.7174	0.2299	0.8951	0.8840
Rnd _All_B	4.93	0.7193	0.2312	0.9177	0.9171
Rnd _All_F	4.55	0.7133	0.2246	0.8776	0.8485

4.2.6 Summary of Results

For the first category, where the all the clusters high equal size and high intra-edge density, all methods performed generally well. Modularity, 'All_Adj_F' and 'All_All_F' performed the best on both graphs on all metrics, with almost perfect results every time. 'Rnd_Adj_B'

and ‘Rnd_All_B’ performed slight worse than the previous ones but still pretty great with results for both NMI and ARI above 97%. Furthermore, their average number of communities found 4.1 instead of 4. Finally, the rest of the methods performed great on the first category were the graph had 60 nodes and 4 clusters, but on the next category with 80 nodes and 5 clusters, their percentage on NMI dropped about 10% to approximately 88% while the ARI dropped even more to almost 20% on some cases. Moreover, their number of detected communities was 3.7 for some cases instead of 5.

On the next category, where clusters are of equal size but the intra-edge density is dropping gradually for every cluster, in graphs with 60 nodes and 4 clusters, ‘All_Adj_F’, ‘All_All_F’, ‘Rnd_Adj_B’, ‘Rnd _All_B’ and modularity performed really well with both NMI and ARI nearly at 99%. Their number of detected communities was deviated by 0.01. ‘Rnd _All_F’, ‘All_Adj_B’ and ‘All_All_B’ performed worse than the others with NMI and ARI at 90%, while their average number of communities deviated by 0.6. On the other hand, on graphs with 80 nodes and 5 clusters all methods dropped their percentages on all metrics except from ‘All_Adj_F’, ‘All_All_F’ which maintained their great performance.

On the next category, where clusters are of various size and have high intra-edge density, ‘All_Adj_F’, ‘All_All_F’ , ‘Rnd _Adj_B’, ‘Rnd_Adj_F’ and ‘Rnd _All_B’ performed remarkably well on graphs with 60 nodes and 4 clusters, with average NMI and ARI reaching 99% while their average number of clusters found had a very small deviation of 0.08. Modularity and ‘Rnd_All_F’, performed slightly worse than the previous ones but still very good with an average of 96% on both NMI and ARI. As for their number of not detected clusters, it is about 0.35. ‘All_Adj_B’ and ‘All_All_B’ performed the worst once again with an average of 92% on NMI and 90% on ARI while their number clusters found was 3.15 instead of 4. On the contrary, on graphs with 80 nodes and 5 clusters, all methods suffered from a significant reduction on all metrics, with both ‘All_Adj_F’, ‘All_All_F’ outperforming the other methods.

In the category where the clusters are of various size and have intra-edge densities that are gradually reduced, resulting in smaller sized clusters having larger intra-edge density, methods performed really well for the case of graphs with 60 nodes and 4 clusters. More specifically, ‘All_Adj_F’, ‘All_All_F’ and ‘Rnd _All_B’ performed the best with an average of 98% on both NMI and ARI, while their number of clusters found deviated by 0.08. Modularity, ‘Rnd_Adj_B’, ‘Rnd _Adj_F’ and ‘Rnd _All_F’, performed slightly worse than the previous ones but still very good with an average of 96% on both NMI and ARI. As for their number of clusters found, it is about 3.7 instead of 4. ‘All_Adj_B’ and ‘All_All_B’ performed the worst once again with an average of 89% on NMI and 86% on ARI while their number of clusters found was approximately 3.1 instead of 4. For the case of 80 nodes and 5 clusters, all methods suffered a 5-10% reduction on both NMI and ARI except from ‘All_Adj_B’, ‘All_All_B’ and ‘Rnd _All_F’ which suffered a significant reduction up to 15%.

For the last category, where clusters are of various size and have intra-edge densities that are gradually increased, resulting in larger sized clusters having larger intra-edge density, almost all methods performed well on graphs with 60 nodes and 4 clusters. More specifically, ‘All_Adj_F’, ‘All_All_F’, ‘Rnd _Adj_B’, ‘Rnd_Adj_F’ ,‘Rnd _All_B’ and ‘Rnd _All_F’ performed really well with an average of 96% for NMI and 97% for ARI.

Furthermore, their number of detected clusters deviated by 0.05. Modularity, 'All_Adj_B' and 'All_All_B' performed the worst with 90% on both NMI and ARI, while their average number of detected clusters deviated by 0.8. On the graphs with 80 nodes and 5 clusters all methods suffered a 5-10% on every metric.

4.2.7 Large Graphs – Optimization via Spectral Clustering

In this section, we present the results for the spectral optimization. More specifically, we run the spectral clustering algorithm for k number of clusters ranging from 2 to 20 for each graph, and store the partition that maximized modularity and inclusion respectively. We did 20 independent runs for each category.

Table 11 Results for Graph Model with 1000 Nodes, 8 Clusters, Equally Distributed Cluster Size, External Probability 15% and Probability List ranging from 90% to 100%.

	Avg Clusters	**Avg I**	**Avg Mod**	**Avg NMI**	**Avg ARI**
Modularity	**8**	**0.6684**	**0.2122**	**1**	**1**
Inclusion	**8**	**0.6684**	**0.2122**	**1**	**1**

Table 12 Results for Graph Model with 1000 Nodes, 8 Clusters, Equally Distributed Cluster Size, External Probability 10% and Probability List ranging from 90% to 100% for the First Cluster Followed by a 10% Reduction for each Subsequent Cluster

	Avg Clusters	**Avg I**	**Avg Mod**	**Avg NMI**	**Avg ARI**
Modularity	7.3	0.6200	**0.1604**	0.9694	0.9087
Inclusion	**8**	**0.6213**	0.1600	**0.9973**	**0.9977**

Table 13 Results for Graph Model with 1000 Nodes, 8 Clusters, Distributed Cluster Size with Descending Order (20%, 20%, 15%, 15%, 10%, 10%, 5%, 5%), External Probability 15% and Probability List ranging from 90% to 100%.

	Avg Clusters	**Avg I**	**Avg Mod**	**Avg NMI**	**Avg ARI**

Modularity	6.9	0.6877	**0.2264**	0.9786	0.9733
Inclusion	**8**	**0.6883**	0.2260	**1**	**1**

Table 14 Results for Graph Model with 1000 Nodes, 8 Clusters, Distributed Cluster Size with Ascending Order (5%, 5%, 10%, 10%, 15%, 15%, 20%, 20%), External Probability 10% and Probability List ranging from 90% to 100% for the First Cluster Followed by a 10% Reduction for each Subsequent Cluster

	Avg Clusters	**Avg I**	**Avg Mod**	**Avg NMI**	**Avg ARI**
Modularity	**8**	**0.6054**	**0.1400**	**0.9997**	**0.9997**
Inclusion	**8**	**0.6054**	**0.1400**	**0.9997**	**0.9997**

Table 15 Results for Graph Model with 1000 Nodes, 8 Clusters, Distributed Cluster Size with Descending Order (20%, 20%, 15%, 15%, 10%, 10%, 5%, 5%), External Probability 10% and Probability List ranging from 90% to 100% for the First Cluster Followed by a 10% Reduction for each Subsequent Cluster

	Avg Clusters	**Avg I**	**Avg Mod**	**Avg NMI**	**Avg ARI**
Modularity	5	0.6613	**0.2057**	0.8932	0.7827
Inclusion	**7**	**0.6661**	0.2031	**0.9756**	**0.9774**

For the spectral optimization on graphs with equally distributed and highly dense clusters, both methods yielded perfect results, as they found the ground truth solution every time. For the next category where the cluster sizes where equally distributed but there was various inter-edge probability, the modularity failed to detect the correct number of communities on some cases, resulting in a 96% NMI and 90% of ARI while its number of detected communities deviated by 0.7. On the other hand, inclusion found the ground truth solution nearly every time with an average of 99% on both NMI and ARI.

On the next category where the cluster sizes are created with various sizes, inclusion found the ground truth partition on every occasion. On the contrary, modularity failed to detect the correct number of communities once again providing an average of 6.9 communities instead of 8.

On graphs where smaller clusters are denser, both methods yielded perfect solutions every time. Finally, on the last category, where smaller clusters are sparser, both methods failed to detect the correct number of communities. Inclusion though, found 7 instead of 8

communities with 97% NMI and ARI while modularity found 5 instead of 8 communities with 89% on NMI and 78% on ARI.

Table 16 Results for Graph Model with 2000 Nodes, 16 Clusters, Equally Distributed Cluster Size, External Probability 15% and Probability List ranging from 95% to 100%.

	Avg Clusters	**Avg I**	**Avg Mod**	**Avg NMI**	**Avg ARI**
Modularity	**16**	**0.5959**	**0.1294**	**1**	**1**
Inclusion	**16**	**0.5959**	**0.1294**	**1**	**1**

Table 17 Results for Graph Model with 2000 Nodes, 16 Clusters, Equally Distributed Cluster Size, External Probability 15% and Probability List ranging from 95% to 100% for the First Cluster Followed by a 5% Reduction for each Subsequent Cluster

	Avg Clusters	**Avg I**	**Avg Mod**	**Avg NMI**	**Avg ARI**
Modularity	13	0.5648	**0.0926**	0.9333	0.7021
Inclusion	**14.8**	**0.5658**	0.0919	**0.9630**	**0.8995**

Table 18 Results for Graph Model with 2000 Nodes, 16 Clusters, Distributed Cluster Size with Descending Order (10%, 10%, 10%, 8%, 8%, 8%, 7.5%, 6.25%, 6.25%, 6.25%, 5.25%, 5%, 3.5%, 2%, 2%, 2%), External Probability 15% and Probability List ranging from 95% to 100%.

	Avg Clusters	**Avg I**	**Avg Mod**	**Avg NMI**	**Avg ARI**
Modularity	13.9	0.6095	**0.1461**	0.9853	0.9749
Inclusion	**15.7**	**0.6097**	0.1460	**0.9982**	**0.9977**

Table 19 Results for Graph Model with 2000 Nodes, 16 Clusters, Distributed Cluster Size with Ascending Order (2%, 2%, 2%, 3.5%, 5%, 5.25%, 6.25%, 6.25%, 6.25%, 7.5%, 8%, 8%, 8%, 10%, 10%, 10%), External Probability 15% and Probability List ranging from 95% to 100% for the First Cluster Followed by a 5% Reduction for each Subsequent Cluster

	Avg Clusters	**Avg I**	**Avg Mod**	**Avg NMI**	**Avg ARI**
Modularity	13.8	0.5563	**0.0795**	0.9598	0.8629
Inclusion	**15.4**	**0.5566**	0.0793	**0.9818**	**0.9421**

Table 20 Results for Graph Model with 2000 Nodes, 16 Clusters, Distributed Cluster Size with Descending Order (10%, 10%, 10%, 8%, 8%, 8%, 7.5%, 6.25%, 6.25%, 6.25%, 5.25%, 5%, 3.5%, 2%, 2%, 2%), External Probability 15% and Probability List ranging from 95% to 100% for the First Cluster Followed by a 5% Reduction for each Subsequent Cluster

	Avg Clusters	**Avg I**	**Avg Mod**	**Avg NMI**	**Avg ARI**
Modularity	10.1	0.5925	**0.1291**	0.9085	0.7074
Inclusion	**13.4**	**0.5944**	0.1276	**0.9763**	**0.9639**

To even further harden the problem, we test both methods on even bigger graphs with more clusters. On the first category where all cluster sizes are equally distributed, both methods found perfect solutions every time. For the next category where the cluster sizes where equally distributed but there was various inter-edge probability, inclusion outperformed modularity with 96% on NMI and 90% on ARI while the average number of communities found was 14.8 instead of 16. Modularity, found 13 communities instead of 16 with 93% NMI and surprisingly low ARI 70%.

On the next category where the cluster sizes are created with various sizes, both methods performed really well with 98% and 99% on both NMI and ARI, for modularity and inclusion, respectively. The average number of communities found was 13.9 for modularity and 15.7 for inclusion, instead of 16.

On graphs where smaller clusters are denser, both methods performed really well once again. More specifically, modularity reached 96% on NMI and 86% on ARI while its number of detected communities is 13.8 instead of 16. Inclusion, performed slightly better with 98% on NMI and 94% on ARI while its number of detected communities is 15.4 instead of 16.

Finally, on the last category, where smaller clusters are sparser, both methods struggled on the correct number of the underlying communities. More precisely, modularity found 10.1

communities on average instead of 16 and reached 90% on NMI and 71% on ARI. On the other hand, inclusion performed significantly better as the number of detected communities on average was 13.4 and its NMI and ARI is approximately 97%.

4.2.8 Real-World Graphs

Table 21 presents some basic statistics on the two real-world networks we used. Furthermore, Table 22 and 23 present the results on the Zachary's Karate Club and American College Football datasets respectively.

Table 21 Real-World Networks' Statistics

	Number of Nodes	**Number of Edges**	**Number of Clusters**
Karate Club	34	78	2
American Football	115	616	12

Table 22 Results for Karate Club Dataset (2 Clusters)

	Avg Clusters	**Avg I**	**Avg Mod**	**Avg NMI**	**Avg ARI**
Modularity	**4**	**0.8126**	**0.4188**	0.6176	**0.4619**
All_Adj_B	9	0.7547	0.2940	0.5303	0.2321
All_Adj_F	11	0.7352	0.2588	0.5021	0.1576
All_All_B	9	0.7547	0.2940	0.5303	0.2321
All_All_F	11	0.7352	0.2588	0.5021	0.1576
Rnd_Adj_B	9.54	0.7563	0.2932	0.5280	0.2142
Rnd _Adj_F	6.11	0.7901	0.3687	**0.6305**	0.4142
Rnd _All_B	9.78	0.7521	0.2854	0.5202	0.1992
Rnd _All_F	6.15	0.7905	0.3666	0.6046	0.3803

Table 23 Results for American College Football Dataset (12 Clusters)

	Avg Clusters	Avg I	Avg Mod	Avg NMI	Avg ARI
Modularity	10	**0.8367**	**0.6043**	0.8856	0.8035
All_Adj_B	7	0.8350	0.6006	0.7955	0.5784
All_Adj_F	**11**	0.8363	0.6031	**0.9115**	**0.8569**
All_All_B	7	0.8350	0.6006	0.7955	0.5784
All_All_F	10	0.8342	0.5998	0.8895	0.8006
Rnd_Adj_B	10.9	0.8317	0.5948	0.8940	0.8003
Rnd _Adj_F	10.42	0.8308	0.5931	0.8867	0.7854
Rnd _All_B	10.71	0.8330	0.5970	0.8927	0.7934
Rnd _All_F	6.98	0.8111	0.5568	0.77573	0.5262

Table 24 Spectral Optimization for Zachary's Karate Club Dataset

	Avg Clusters	Avg I	Avg Mod	Avg NMI	Avg ARI
Modularity	**4**	**0.8126**	**0.4188**	0.6176	**0.4619**
Inclusion	**4**	**0.8126**	**0.4188**	0.6176	**0.4619**

Table 25 Spectral Optimization for American College Football Dataset

	Avg Clusters	Avg I	Avg Mod	Avg NMI	Avg ARI
Modularity	**11**	**0.8363**	**0.6031**	**0.9115**	**0.8569**
Inclusion	**11**	**0.8363**	**0.6031**	**0.9115**	**0.8569**

For the Zachary's Karate Club dataset, modularity found 4 communities instead of 2 with an NMI of 62% and ARI of 46. 'Rnd_Adj_F' and 'Rnd_All_F' performed similar to modularity in regard to NMI and ARI with 62% and 41% respectively but the number of communities

found is on average 6.1 instead of 2. All the other methods performed significantly worst with NMI nearly at 50% and ARI at 20% while on some cases the number of communities found is 11. That results indicate that the Karate Club is quite a difficult problem and probably there more than 2 actual communities.

On American College Football dataset, the results were significantly better. 'All_Adj_B', 'All_All_B' and 'Rnd _All_F' failed to detect the 5 clusters which led to poor results with NMI of 78% and ARI of 55%. On the other hand, all the other methods found at least 10 of the actual clusters with 'All_Adj_F' performing the best with 11 clusters found and NMI of 91% and ARI of 86%.

For the spectral optimization on Zachary's Karate Club dataset, both methods chose the exact same solution. More specifically, they found 4 communities instead of 2 with an NMI of 62% and ARI of 46%. The same thing occurred with the American College Football, as both methods chose the exact same solution. The clusters found was 11 instead of 12, while the NMI was 91% and ARI was 86%.

CHAPTER 5.

CONCLUSION AND FUTURE WORK

5.1 Conclusion

In this thesis, we studied the community detection problem and introduced a new quality measure, inclusion. Furthermore, we presented several optimization techniques for this criterion and compare them to the most popular family of community detection methods which are based on the optimization of the so called 'modularity' criterion using various clustering approaches.

In the experimental evaluation we conducted, we deducted some valuable insights. First of all, in almost every case we examined, higher inclusion values led to better results, on both the quality of the solution (NMI and ARI) and actual number of communities detected. Thus, the optimization of the inclusion measure can help in solving the community detection problem.

Although on smaller graphs both methods optimizing inclusion and modularity performed remarkably well, when we scale out, modularity seems to struggle on detecting smaller communities. On the other hand, our methods and specifically 'All_Adj_F' and 'All_All_F', outperformed modularity on almost every occasion. Finally, when we used the spectral optimization, thus narrowing our selection of solutions to a specific set, inclusion outperformed modularity as a choice criterion.

5.2 Future Work

Given the encouraging community detection results obtained from the use of the proposed inclusion measure, there are several research directions to be followed. At first it would be interesting to test the approach on various community detection applications arising in biological, social and other types of networks.

It would be also interesting to conduct a more detailed analysis of the strengths and weaknesses of the proposed measure that will lead to the identification of graph cases where the method succeeds or fails.

Another research direction is to consider alternative approaches for optimizing inclusion, in analogy with the various techniques that have been proposed for optimizing modularity (e.g. simulated annealing, alternative greedy search schemes, etc.).

It would also be important if we could formulate the inclusion maximization problem as a trace maximization problem in analogy to the spectral clustering objective. In such a case, the solution could be obtained from the eigenvectors of the corresponding matrix.

Finally, another important research direction concerns the possible use of inclusion measure to detect communities in weighted graphs as well as in directed graphs. In such a case, an adaptation of the inclusion definition would be necessary to take into account the richer connection information included in the edge matrix.

REFERENCES

[Euler36] L. Euler, Solutio problematis ad geometriam situs pertinentis, Commentarii Academiae Petropolitanae 8, 1736.

[Boll98] B. Bollobas, Modern Graph Theory, Springer Verlag, New York, USA, 1998.

[WaFa94] S. Wasserman, K. Faust, Social Network Analysis, Cambridge University Press, Cambridge, UK, 1994.

[ErRé59] P. Erdös, A. Rényi, On random graphs. I., Publ. Math. Debrecen 6, 1959.

[WaSt98] D. J. Watts, S. H. Strogatz, Collective dynamics of small-world networks, Nature, 1998.

[GiNe02] M. Girvan, M.E.J. Newman, Community structure in social and biological networks, Proc. Natl. Acad. Sci. USA 99 12, 2002.

[Fort10] S. Fortunato, Community detection in graphs, Physics Reports, 2010.

[KeLi70] B.W. Kernighan, S. Lin, An efficient heuristic procedure for partitioning graphs, Bell Syst. Tech. J. 49 (1970) 291-307.

[MacQ67] J.B. MacQueen, Some methods for classification and analysis of multivariate observations, in: L.M.L. Cam, J. Neyman (Eds.), Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, University of California Press, Berkeley, USA, 1967, pp. 281-297.

[NgJW01] A.Y. Ng, M.I. Jordan, Y. Weiss, On spectral clustering: Analysis and an algorithm, in: T.G. Dietterich, S. Becker, Z. Ghahramani (Eds.), Advances in Neural Information Processing Systems, vol. 14, MIT Press, Cambridge, USA, 2001.

[Mack03] D.J.C. Mackay, Information Theory, Inference, and Learning Algorithms, Cambridge University Press, Cambridge, UK, 2003.

[Wink03] R.L. Winkler, Introduction to Bayesian Inference and Decision, Probabilistic Publishing, Gainesville, USA, 2003.

[HaRT07] M.S. Handcock, A.E. Raftery, J.M. Tantrum, Model based clustering for social networks, J. Roy. Statist. Soc. A 170 (2007) 1-22. Working Paper no. 46.

[Hast06] M.B. Hastings, Community detection as an inference problem, Phys. Rev. E 74 (3) (2006) 035102.

[Gall63] R.G. Gallager, Low Density Parity Check Codes, MIT Press, Cambridge, USA, 1963.

[NeGi04] M.E.J. Newman, M. Girvan, Finding and evaluating community structure in networks, Phys. Rev. E 69 (2) (2004) 026113.

[BDG+06] U. Brandes, D. Delling, M. Gaertler, R. Görke, M. Hoefer, Z. Nikolski, D. Wagner, On modularity np-completeness and beyond, 2006.

[Newm04] M.E.J. Newman, Fast algorithm for detecting community structure in networks, Phys. Rev. E 69 (6) (2004) 066133.

[ClNM04] A. Clauset, M.E.J. Newman, C. Moore, Finding community structure in very large networks, Phys. Rev. E 70 (6) (2004) 066111.

[BGLL08] V.D. Blondel, J.-L. Guillaume, R. Lambiotte, E. Lefebvre, Fast unfolding of communities in large networks, J. Stat. Mech. P10008 (2008).

[KiGV83] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Optimization by simulated annealing, Science 220 (1983) 671-680.

[GuSA04] R. Guimerà, M. Sales-Pardo, L.A.N. Amaral, Modularity from fluctuations in random graphs and complex networks, Phys. Rev. E 70 (2) (2004) 025101 (R).

[FoBa07] S. Fortunato, M. Barthélemy, Resolution limit in community detection, Proc. Natl. Acad. Sci. USA 104 (2007) 36-41.

[Zach77] W.W. Zachary, An information flow model for conflict and fission in small groups, J. Anthropol. Res. 33 (1977) 452-473.

SHORT CV

Nikolaos Koufos was born in Athens, Greece in 1992. In 2010, he enrolled in the Computer Science & Engineering department in University of Ioannina and in 2015 he received his diploma. In continuation to his studies, he joined the same department for his Master's Degree. After fulfilling his responsibilities as a post-graduate student, he presented his thesis in July 2017 in order to complete his Master's Degree.