Community Detection in Imperfect Networks

Johan Dahlin

Information Systems, Swedish Defence Research Agency.
Department of Physics, Umeå University.

June 3, 2011

Community Detection in Imperfect Networks
Master's thesis, Master of Science in Engineering Physics, Umeå University.
Johan Dahlin, johan.dahlin@foi.se or joda0009@student.umu.se.

This thesis is part of the project *Tools for information management and analysis*, which is funded by the R&D programme of the Swedish Armed Forces.

Supervisor: Pontus Svenson, Swedish Defence Research Agency.
Examiner: Sang Hoon Lee, Department of Physics, Umeå University.

Presented: Umeå University, May 30, 2011.
Approved for print: June 3, 2011.

Abstract

Community detection in networks is an important area of current research with many applications. Finding community structures is a challenging task and despite significant effort no satisfactory method has been found. Different methods find different communities in the same network and with different computational requirements. To counter this problem, several different methods are often used and the results compared manually. In this thesis, we present three different methods to instead merge the results from different methods (or several runs from the same algorithm) to find better estimates of the community structure.

Another problem in practical applications is noisy and imperfect networks with missing and false edges. These imperfections are natural results from the methods used to map the network structure and are often difficult to eliminate. In this thesis, we apply a Monte Carlo-sampling method in combination with the introduced methods for merging community detection results to find community structures in such networks. The method is tested by simulation studies on both real-world networks and synthetic networks with generated uncertainties and imperfections. We finally demonstrate how it is possible to generate confidence levels of the obtained community structure from the merging methods. This allows for a qualitative comparison of the robustness and significance of the network clustering.

Keywords: social network analysis, imperfect networks, uncertain edges, community detection, ensemble clustering.

Sammanfattning

Identifikation av grupperingar i nätverk är ett viktigt område inom aktuell forskning med många olika tillämpningsområden. Att finna grupperingar är ofta svårt och trots betydande ansträngningar har ingen tillfredsställande metod hittats. Olika metoder finner ofta olika grupperingar i samma nätverk och kräver varierande beräkningskraft. För att hantera dessa problem används ofta flera metoder vartefter resultaten jämförs manuellt. I detta examensarbete presenterar vi tre olika metoder att istället slå samman resultat från olika metoder (eller fler körningar från samma algoritm) för att hitta bättre uppskattningar av grupperingarna.

Ett annat problem i praktiska tillämpningar är brus och ofullständiga nätverk med saknade och falska kanter. Dessa brister är naturliga resultat från de metoder som används för att kartlägga nätverketstrukturen och det är ofta svåra att eliminera dessa. I detta examensarbete använder vi Monte Carlo-metoder i kombination med de introducerade metoderna för att slå samman funna grupperingar för att hitta grupperingar i det osäkra nätverket. Vi testar metoden genom simuleringstudier på både verkliga och syntetiska nätverk med genererade osäkerheter och brister. Slutligen demostrerar vi hur det är möjligt att skapa konfidensnivåer för noder i grupperingar med hjälp av metoderna för sammanslagning. Detta möjliggör en kvalitativ jämförelse av stabilitet och signifikans av identifierade nätverksgrupperingar.

Nyckelord: social nätverksanalys, ofullständiga nätverk, osäkra kanter, grupperingar, ensembleklustering.

Preface

This is my Master's thesis for the degree of Master of Science in Engineering Physics at Umeå University. The thesis has been written at the Department of Informatics, Swedish Defence Research Agency (FOI) during the spring of 2011.

This thesis was not written in a vacuum and I owe much gratitude to many people that have supported and aided my work. I would especially like to thank my supervisor Pontus Svenson for his guidance and support during the entire Master's thesis. The thesis work would not have progressed as good as it has without our creative discussions and his enthusiasm. Furthermore, I would like to thank my supervisor Sang Hoon Lee for all helpful comments, suggestions, and discussions during the project.

I thank the participants of the NORDITA-conference on *Applications of network theory: from mechanisms to large-scale structure* for the possibility of presenting my findings before world-class network scientists. The helpful comments and suggestions from the presentation greatly improved my thesis.

I am grateful for the support, interest, and encouragement from all the people at the Department of Informatics at FOI. Without you, my Master's thesis work would have been dreading and monotonous. At last, I owe much to my wonderful family and friends without their loving support and encouragement my life would not be the same.

Johan Dahlin,
Kista, Stockholm,
June 3, 2011.

Contents

Chapter 1

Introduction

Networks are everywhere in nature and models of them are useful in describing and analyzing many different kinds of relationships and interdependencies. Studies of these networks and their structures have recently been the focus of much research. Much effort has been given to study the problem of finding so called community structures in networks. A *community* is a group of nodes that are more densely connected to each other than to other nodes outside the group. Community structures are a distinct feature of some real-world networks that are rarely found in fully random networks. Community detection is useful in many important applications ranging from grouping metabolic and protein-interaction networks to identifying persons with similar shopping patterns and structural studies of the Internet. [1, 2, 3, 4, 5, 6]

Motivation In practical applications network data is often gathered by empirical methods and therefore some uncertainty is included. It is often the case that the full network with all nodes and edges is not known with certainty, as the network is not often fully observable directly. Note that problems with falsely included edges and nodes are equally important. An example of this is the use of communication data and observed behavior in constructing friendship networks in groups of people. Empirically gathered data is also filled with uncertainties and conflicting information, which occur as the result of e.g. measurement errors, or from un-truthful participants in surveys and interviews.

Earlier work in community detection has been concerned only with certain networks. Uncertain information has traditionally been treated by either removing or including all uncertain edges and nodes in the network. This has consequences as the uncertain information is completely removed or wrongly included. We therefore propose new methods to analyze community structures in networks containing uncertain and incomplete network information. A framework is introduced to merge evidence from multiple sources to find probabilities for the existence of nodes, edges and higher-order networks structure. Furthermore, methods to analyze the community

structure of these uncertain and imperfect networks are proposed. This enable full use of the uncertain information which should yield better estimates of the community structure than the two traditional solutions.

Background *Social Network Analysis* (SNA) is used to identify social roles, important groups, and hidden organization structures from gathered social data. SNA originated in the social sciences during the first part of the 20th century, when sociologist investigated human social behavior by observing groups and individuals. SNA can be said to have been founded in the 1930's when Moreno first used the *sociogram*, thus founding the field of sociometry. The methods later spread into other social sciences, like anthropology and psychology, but remained unknown to the natural sciences. [1]

An interesting and important problem in SNA is to identify groups and communities, i.e. modules in or groups of the network. The first applications of community detection was to analyze the community structure of workgroups in the U.S. government [7] and in voting patterns [8]. These communities were detected using manual methods, often with graphical means. The first use of mathematics and graph theory for analyzing social network was performed in a block-diagonal form or *sociomatrix*. Using statistical methods, known as clustering analysis, these matrices are traditionally used as similarity measures between nodes in the network. [3]

In 2002, physicists were introduced to community detection by Refs. [9, 10] that discussed some problems faced by network analysis and how methods from physics could help solve them. This sparked a large interest in analyzing social networks using sophisticated tools from the natural sciences. The increasing interest in SNA was also due to the emergence of social community websites, web-crawlers, and sequencing of DNA, which produced an unprecedented large amount of network data for scientists to analyze. Much effort was given to the analysis of the structure of social networks. [1, 3]

Previous and related work This thesis builds on and refines work in community detection in (weighted) networks and ensemble clustering methods from data clustering problems. Much of the background and previous work concerning these methods are introduced in *Chapter 2* of this thesis. Therefore, we only discuss these topics in brief in this introduction. Community detection has been an important research area in network analysis for some time, as mentioned earlier the first methods were manual without computers.

In recent years a large number of methods for detecting community structures have been developed, drawing on knowledge from many different fields, e.g. statistical mechanics, discrete mathematics, computer science, statistics, and sociology. These methods have also been improved to handle weighted, directed, and multi graphs. To our knowledge, no previous work has been performed with uncertain graphs but some published works study the effects

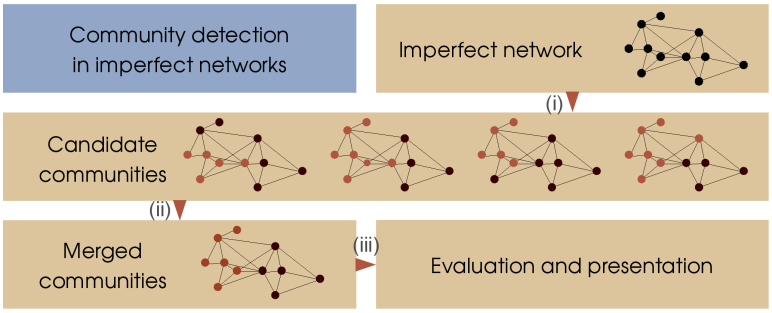

Figure 1.1: The proposed steps to detect community structures in imperfect networks. (i) sampling from the ensemble of consistent networks and find the community structure in each *candidate network*. (ii) merging the candidate communities using Fusion of Communities. (iii) evalutation and presentation of the obtained community structure.

of missing links and robustness of clustering. A thorough review of the current state in community detection is given in Ref. [3] and more background is given in *Chapter 2*.

Ensemble clustering is a technique used in e.g. bioinformational applications and was created to merge several clustering results into one. To our knowledge, no work has been devoted to applying these methods to community detection problems. However other methods have been used to merge several community structures, e.g. voting in Refs. [11, 12]. As data clustering and community detection are quite similar, it should be possible to merge communities in the same manner as ensembles of data with good results. Ensemble clustering methods were first introduced in Ref. [13] and further develop by e.g. Refs. [14, 15]. See *Chapter 2* for additional information about ensemble clustering methods.

Contribution and method In this thesis, we propose new methods for community detection in imperfect networks. This is accomplished using a three step procedure outlined in *Figure 1.1* with sampling, community detection, and merging. We present novel methods to sample from the ensemble of consistent networks, three different methods for merging several candidate community structures and for validating the merged results.

To validate the methods introduced in this thesis, we test the three merging methods on generated synthetic imperfect networks and scrambled real-world networks. These synthetic networks are designed to investigate the robustness of the methods related to the amount of uncertainty and the presence of missing/false edges. The resulting community structure is compared with the solution given by external information and the structure found by the same community detection method applied on the original network without uncertainty.

Results The main results include a demonstration that merging can be used to increase the effectiveness of fast stochastic community detection methods. One method, using *label propagation* [11], is shown to perform as well as the more advanced *spin glass* [16] method and this at a lower complexity. We also demonstrate how this merging method in combination with sampling to analyze uncertain networks. Most of the community structure is recovered in networks where only half of the edges are known with certainty.

Finally, we apply the same methods to imperfect networks, where edges have been randomly added and removed to simulate false/missing edges. The methods perform reasonably well at low level of noise but the community structure is lost in sparse networks even when adding only a few false edges. Therefore, this method is only applicable in denser networks where much network information is available.

The Label Propagation (LP) community detection method used in combination with NFC is the recommended method for detecting communities in imperfect network. This method generates good (or the best) results in the simulation studies conducted to evaluate the accuracy and performance of the proposed combinations of methods.

Disposition The outline of this thesis is as follows: *Chapter 2* presents and discusses previous work in cluster analysis, ensemble clustering, and community detection. *Chapter 3* provides a review some basic methods in uncertainty theory and propose a framework to combine evidence to form an imperfect network. *Chapter 4* combines the material in the two previous chapters into methods for analyzing uncertain/imperfect networks.

Chapters 5 and *6* discuss the design, implementation and results of the conducted simulation experiments. The thesis is concluded with *Chapter 7* containing a summary and some remarks. *Appendices A* and *B* contain elementary graph theory, spectral graph partitioning and centrality measures.

Chapter 2

Clustering analysis and community detection

Detecting communities in networks is comparable to partitioning sets of data into similar clusters. The main difference is that data clustering allows for grouping any pair of data points together, whereas community detection only allows for directly[1] grouping linked nodes together. We therefore begin this chapter by reviewing data clustering methods and discussing some methods for validation of clustering results. Ensemble clustering is introduced for combining ensembles of data clusterings; this method is later generalized in *Chapter 4* for community detection methods.

We continue by reviewing the generalized versions of data clustering methods for clustering nodes in networks. These techniques are generally referred to as *community detection* methods and are both similar and different compared to data clustering methods. We discuss the definition of a community as well as how to detect them manually as well as using computer algorithms. Common problems regarding community detection in networks are uniqueness, existence, and validation of the communities. These problems are discussed in this chapter, as they have implications for detection of community structures in uncertain and imperfect networks.

2.1 Data clustering

Cluster analysis is the umbrella term for statistical methods to divide a set of observed random data into subsets (clusters) without any external information. Clustering is therefore an unsupervised method different from *classification* methods that uses external information to train classifiers which group the data. Clustering has important applications in many different

[1] They can be grouped in the same communities even if they are not directly connected but indirectly through a number of short paths.

fields including biology, computer science, and the social sciences, to create some understanding of a data material by grouping data together into (perhaps) easily identifiable clusters. Another purpose of clustering is to simplify or reduce the data material before using other statistical tools, e.g. principal component analysis or classification. [17]

The main problem in data clustering is dividing a set of data, $\mathbf{x} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, into exactly k disjoint subsets, $\mathbf{x} = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(k)}\}$, in which all elements in some sense are similiar to each other. This problem is typically solved by using one of many possible combinations of clustering algorithms and similarity measures. Two different types of clustering algorithms are commonly used in practice: *hierarchical clustering* and *partitional clustering*. Three common families of *similarity measures* used to quantify the similarity between data points are: (i) *distance-based measures*, e.g. the usual norms (L_1, L_2, and supremum norms), (ii) *distribution-based measures*, e.g. Kullbeck-Leibler divergence for probability distributions, and (iii) *density-based measures* by finding peaks in the kernel density landscape. [18, 19]

The different algorithms and similarity measures yield different results and are used to cluster different kinds of data in different applications. The advantages of using hierarchical compared with partitional clustering, are the capability to find overlapping clusters and that it does not require the number of clusters to be specified beforehand. The main drawback of hierarchical clustering is that it has a higher computational complexity, thus only allowing for smaller data sets to be clustered. [20, 17, 18].

Example 1 (Clustering random data)**.** In *Figure 2.1*, the result of clustering random data is presented using k-means and agglomerative hierarchical clustering. The left-hand part of the figure shows the clustering as found by the k-means algorithm with the '+' indicating the centroids. The dendrogram shows the result from the hierarchical clustering, which is identical to the solution found by k-means.

2.1.1 Hierarchical clustering

The two most common hierarchical clustering methods are called *agglomerative hierarchical clustering* (merging data points from the bottom-up) and *divisive hierarchical clustering* (splitting the full data set into subsets top-down). The agglomerative version iteratively merges the most similar data points (and clusters) until only one cluster containing all data points remains, shown in *Algorithm 1*. Finding the similarity between the merged data points (clusters) require some *linkage rule* which is determined by the similarity measure used. The three most common rules are the *single*, *complete*, and *mean* linkage. The single (complete) linkage is equal to the maximum (minimum) similarity between two points in different clusters. The

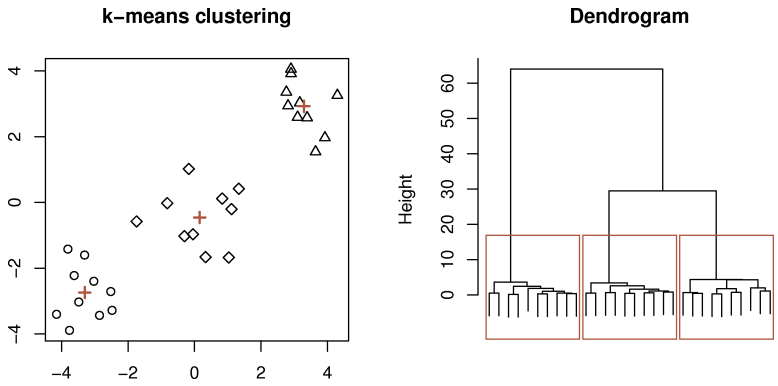

Figure 2.1: The clustering of random data by k-means and agglomerative hierarchical clustering using Ward's method. The random data is comprised of 10 sampled random variables from each of the following distributions: $\mathcal{N}(0,1)$, $\mathcal{N}(-3,1)$, $\mathcal{N}(3,1)$. The colored boxes in the right part of the figure indicate the three clusters found by the hierarchical clustering algorithm.

Algorithm 1 Agglomerative hierarchical clustering

Given a similarity matrix.

(i) merge the two most similar clusters,

(ii) update the similarity matrix to include the similarity between the new cluster and the existing clusters,

(iii) repeat (i)-(ii) until only one cluster remains.

mean linkage rule equals the mean similarity between all pairs of data points in different clusters. It is well-known that the choice of linkage method as well as similarity measure greatly influences the result of the clustering. Other problems with hierarchical clustering is that the method is sensitive to outliers and the high computational complexity, $\mathcal{O}(n^2)$, where n denotes the number of data points. [18, 21]

2.1.2 Partitional clustering

The most commonly used partitional clustering algorithm is *k-means* clustering. This algorithm does not merge individual points into clusters but splits the data sets into k clusters simultaneously. This is done by assigning each data point to the most similar centroid, which is calculated using the mean of the cluster of which it is the center. This method is iterative and outlined in *Algorithm 2*.

The simplest possible centroid is found by first randomly placing k centroids in the sample space and then using the sample average to update its position. The choice of centroid calculations and similarity measures have a large impact on the results, as the data points are placed into the cluster

Algorithm 2 k-means clustering

Given k arbitrary points as centroids.

(i) form clusters by assigning each point to the most similar centroid,

(ii) recompute the centroid for each cluster (the mean value of the points in the corresponding cluster),

(iii) repeat until the centroids do not change position (or with some tolerance).

with the most similar centroid. As the k-means method does not require recalculation of similarities it has a low complexity $\mathcal{O}(n)$, thus it is able to cluster larger data sets than hierarchical clustering. The k-means method is also sensitive to outliers and tends to place these in a cluster of leftover data points. [22, 21]

2.1.3 Validation and evaluation

A difficult problem related to clustering is that without any external information, it is very difficult to validate if the clustering is correct or not. This situation is further complicated since it is not often known how many clusters that exist in the data. The main questions as such are if the clustering exists (or is a result of the method used) and the quality of the clustering (if a clustering is assumed to exist).

To solve some of these problems two different classes of methods have been developed for cluster validation: *unsupervised validation* that evaluates clusterings without any additional information; and *supervised validation* which uses external information (labels) to evaluate clusterings.[20]

Some unsupervised methods are the two measures[2], *cohesion*, coh, and *separation*, sep, of a proposed clustering solution is defined using some similarity function, $\mathsf{sim}(\cdot)$

$$\mathsf{coh} = \sum_{\substack{y \in \mathbf{x}^{(i)}, \\ z \in \mathbf{x}^{(i)}}} \mathsf{sim}(y, z), \qquad \mathsf{sep} = \sum_{\substack{y \in \mathbf{x}^{(i)}, \\ z \in \mathbf{x}^{(j)}}} \mathsf{sim}(y, z) \text{ with } i \neq j, \tag{2.1}$$

where $\mathbf{x}^{(i)}$ is set of data points that constitutes cluster i. The separation is the total similarity between two clusters and the cohesion is the total similarity of elements within a cluster. These measure are used to compare two different solutions, typically a good solution has high separation and

[2]It is possible to show that the measures are equivalent to the SSE, Sum of Squared Errors, when the Euclidean distance is used as the similarity measure, $\mathsf{sim}(x, y) = x - y$. This measure is later used as the *Mean Square Error* (MSE) when comparing two community structures. [18]

cohesion. This would indicate dense clusters that are well separated from each other. [18]

A supervised validation method is the *correlation* between the frequency matrix and the proposed solution. We use labels to construct an *ideal frequency matrix*, $\mathbf{F}^* = [F^*_{ij}]$, where $F_{ij} = 1$ if points i and j are in the same cluster and $F_{ij} = 0$ otherwise. The correlation is calculated between the rows of the ideal frequency matrix and the matrix constructed by the obtained solution. High correlation indicates similiar clustering solutions. [20]

2.2 Ensemble clustering

Recently, advances have been made in improving the performance of clustering using *ensemble clustering*. The ensemble clustering problem is concerned with how to combine a given ensemble[3] of clusterings to produce a solution which is a mean of the ensemble. Two solutions to the problem are introduced in Ref. [13]: *Instance-based Ensemble Clustering* (IBEC) and *Cluster-based Ensemble Clustering* (CBEC). [15, 24]

2.2.1 Instance-Based Ensemble Clustering

The first ensemble clustering method uses the frequency of with which two point, i and j are placed in the same cluster as a similarity measure. This similarity is used in a hierarchical clustering method to cluster similar nodes together, i.e. nodes that are often found in the same cluster. [14, 15]

Definition 1 (IBEC)**.** Given an ensemble of clusters, $\mathbf{x} = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(r)}\}$, IBEC constructs a fully connected (complete) graph, $G = (V, \mathbf{F})$, where V is a set of n nodes and $\mathbf{F} = [F_{ij}]$ is a *frequency matrix* with F_{ij} as the frequency of instances that nodes i and j are placed in the same cluster.

Using this complete graph and the frequency matrix, the ensemble cluster is found by using e.g. agglomerative hierarchical clustering (with the frequency matrix as the similarity and one of the linkage methods described above) or some graph partitioning method. [25, 26, 14, 15]

2.2.2 Cluster-Based Ensemble Clustering

The second ensemble clustering method combines clusters from different ensembles that have a large overlap and are similar to each other. CBEC constructs a graph where the nodes represent generated candidate clusters, which are merged into meta-clusters. [15, 27, 28, 29]

[3] Which is often generated by re-sampling methods [23] or random projections [15].

Definition 2 (CBEC)**.** Let $\mathbf{x} = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(R)}\}$, be an ensemble of clusters and write $\mathbf{x} = \{\mathbf{x}^{(11)}, \ldots, \mathbf{x}^{(1K_1)}, \ldots, \mathbf{x}^{(R1)}, \ldots, \mathbf{x}^{(RK_R)}\}$, where $\mathbf{x}^{(ij)}$ represents cluster j formed during run i in the ensemble $\mathbf{x}$. Denote the total number of clusters in $\mathbf{x}$ by $t = \sum_r K_r$, and construct a graph, $G = (V, \mathbf{S})$, where V is a set of t nodes, each representing a cluster. $\mathbf{S} = [S_{ij}]$ is a *similarity matrix* with $S_{ij} = \mathsf{sim}(i, j)$ defined by some suitable measure.

Algorithm 3 CBEC

(i) construct a diagonal matrix, $\mathbf{D} = [D_{ij}]$, where $D_{ii} = \sum_j S_{ij}$ with S_{ij} as the similarity between nodes i and j, and all off-diagonal elements are zero,

(ii) find the normalized similarity matrix, $\mathbf{L}$, by multiplying the similarity matrix, $\mathbf{S}$, with the inverted diagonal matrix, i.e. $\mathbf{L} = \mathbf{D}^{-1}\mathbf{S}$,

(iii) calculate the k largest eigenvectors of $\mathbf{L}$ and form a matrix, $\mathbf{U} = [U_{ij}]$, where U_{ij} is the jth eigenvector for $j = 1, 2, \ldots, k$,

(iv) normalize each row in the matrix eigenvector matrix,

$$\bar{U}_{ij} = \left[\sum_i U_{ij}\right]^{-1} U_{ij}, \tag{2.2}$$

(v) find the meta-clustering by using the k-means method on $\bar{\mathbf{U}} = [\bar{U}_{ij}]$ treating the rows as low-dimensional representation of the network.

The similarity measure in this method is not the same as we used before in the data clustering methods. This measure should instead quantify the similarity between the elements of two sets. Many different measures exist for determining the similarity between sets, e.g. the Jaccard measure (2.3), the cosine measure (2.4), and the symmetric difference (2.5). These different methods are represented in *Figure 2.2* and are quite similar in appearance but do have some different properties which make them suitable for different applications. [15, 30, 31]

$$\mathsf{sim}_{jac}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \frac{\left|\mathbf{x}^{(i)} \cap \mathbf{x}^{(j)}\right|}{\left|\mathbf{x}^{(i)} \cup \mathbf{x}^{(j)}\right|}. \tag{2.3}$$

$$\mathsf{sim}_{cos}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \frac{\left|\mathbf{x}^{(i)} \cap \mathbf{x}^{(j)}\right|}{\sqrt{\left|\mathbf{x}^{(i)}\right| \left|\mathbf{x}^{(j)}\right|}}. \tag{2.4}$$

$$\mathsf{sim}_{sym}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \left|\mathbf{x}^{(i)} \cup \mathbf{x}^{(j)}\right| \setminus \left|\mathbf{x}^{(i)} \cap \mathbf{x}^{(j)}\right|. \tag{2.5}$$

Using the similarity matrix, $\mathbf{S}$, a clustering of the different clusters is found by using *spectral graph partitioning* to find meta-clusters, as outlined in *Algorithm 3*. Each individual node is assigned to the meta-cluster (community) it most often belongs to, breaking ties randomly. [15, 15]

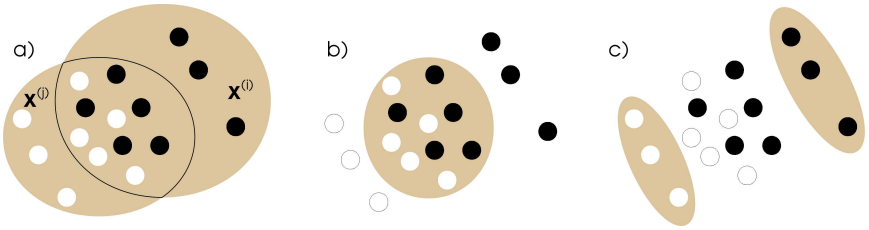

Figure 2.2: a) The two overlapping sets $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$. b) The intersection, $\mathbf{x}^{(i)} \cap \mathbf{x}^{(j)}$, between the sets. c) The symmetric difference (2.5) between the sets.

2.3 Community detection

Networks in general and social networks in particular often contain some form of group structure known as *communities* (other common terms used are *partitions*, *modules*, and *clusters*). Recall that in the context of data clustering each node inside a community is in some sense similar to its neighbors.

For example, we can often find friends, family, and colleagues in the social network of a typical person. These groups can be quite isolated with not many friendships existing between these different groups. In this case, these three groups are the communities of the network. They are also similar in regard with their position and social roles in the network. Therefore, it is in general possible to use the obtained community structure for identifying social roles, hierarchies and hidden groups within the network data material.

Example 2 (Community structure)**.** In *Figure 2.3*, the community structure of a small network with 16 nodes is shown. Note the difference in the inside community degree and the between community degree.

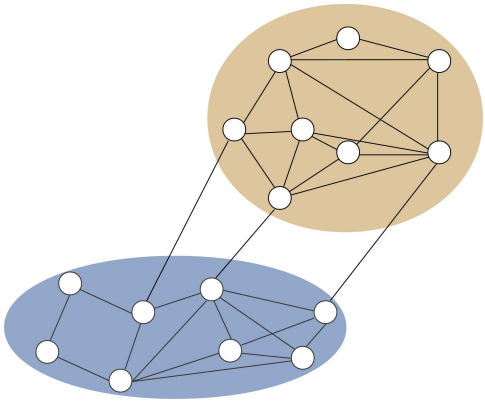

Figure 2.3: Two communities in a simple network. The number of edges in each community is much higher than between the two communities.

This section continues with discussions of the properties of community structures and methods to find and evaluate them. The next section contains a review of some different standard community detection methods.

2.3.1 Existence and uniqueness

Finding communities requires a formal definition of a community that depends on quantifiable network properties. Many different quantitative and qualitative definitions of communities have been proposed on local and global scales in networks. Despite this, no single satisfactory definition has been presented and it is unlikely that it is possible to define a community in general terms. As a result, nearly all network scientists have their own definition of a community and large variations between disciplines are common.

In this thesis, we are satisfied with the qualitative definition presented in *Definition 3*. The main problem with the adopted definition of a community is the lack of a quantified explanation of what denser means.

Definition 3 (Community)**.** A community (in qualitative terms) is a subset of nodes within a network such that connections between nodes are denser than connections with the rest of the network. [32]

A difficult problem in network analysis is proving the existence and uniqueness of community structures. This complication arises primarily as a result of the lack of a universal definition of a community. A pragmatic approach is used to evade that complication in this thesis. It is assumed that a community is simply the structure found by a community detection algorithm.

No concern is given to the problem of proving that the community exists and is not an artifact from the collected data and methods used. The focus is instead on how accurate these communities can be found using the methods proposed. A problem with the adopted pragmatic approach is that different community detection algorithms produce different community structures, making it impossible to tell which is the most correct with absolute certainty.

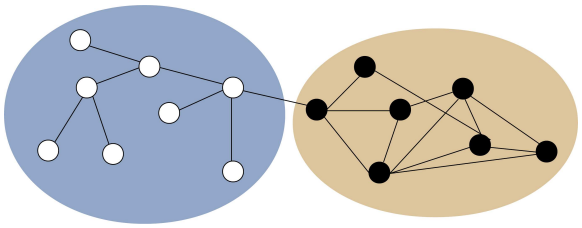

Figure 2.4: The communities of a small network with a tree like structure.

Example 3 (Communities or not?)**.** Even if no formal general definition exists for a community, some structures are often more expected as communities than others. A counter-intuitive examples which illustrates some problems with structures similar to trees and chains is shown in *Figure 2.4*. These structures occur frequently in applications, as most community detection algorithms require all nodes to be a member of a community. This is the result of the low degree of the nodes contained in the chain or tree. The problem is that these structures are not often seen as communities in real life, so the question is if they are communities or not.

Networks often do not contain perfect communities where each node has only one possible clustering; therefore clusters may not be entirely disjoint and unique. As previously discussed hierarchical clustering methods are able to find overlapping clusterings by the use of dendrograms. It is possible to use the same methods in community detection, where methods utilize hierarchical clustering. We briefly discuss[4] the problem with overlapping communities in *Chapter 4.4* where data from the merging methods are used to find confidence levels for the communities.

Some facts concerning community detection in general are important to discuss before continuing with discussions on properties of communities. Communities are a vague and fragile concept found in some real-world networks but seldom in randomly generated networks. Verification and evaluation of detected communities are as difficult as in data clustering, although some methods exist using e.g. resampling methods such as the bootstrap.

Finally, it is important to realize that in the end, the communities detected in the networks are the result of the data which is the only input given. Therefore as in all statistical methods, if the data is flawed the corresponding community structure could be misleading and uncertain. Robustness analysis and similar methods can be used to analyze the significance and stability of the structure found, thereby mimicking hypothesis testing in statistics.

2.3.2 Validation and evaluation

In previous sections, validation of data clusterings were discussed and it was concluded that it is a difficult problem to solve without external information. It is in the nature of unsupervised methods as data clustering that it is difficult to validate the results. The same problems as in data clustering occur in community detection as this method also is unsupervised.

It is however still important to validate the structures found in some manner and also to be able to quantify some goodness-of-fit measure. The latter is useful in comparing different methods which is needed later on in this thesis. Beginning with the validation problem, some help is found

[4]These problems are not discussed at length in this thesis, we refer interested readers to Ref. [3] for methods available to detect overlapping communities in networks.

from the extensive work in data clustering methods. Although seldom used in community detection, the previously introduced measures cohesion and separation are useful when no external information is available.

Another common method in data clustering that has been generalized to community detection problem is re-sampling methods. The idea behind this technique is to perturb the network structure by randomly removing some edges and investigate how the community structure changes. If the structure is relatively robust it is safe to conclude that a significant structure has been found. The re-sampling methods use are often non-parametric and parametric bootstrap. An example of this is given in Ref. [12] where the authors investigate changes in community structures over time, using bootstrap to find the significance level of communities.

We now continue by discussing the problems of evaluating different community structures for both comparison and choosing the optimal number of communities. In community detection problems a quality measure called *modularity* is often used to solve these problems. Modularity measures the degree to which the distribution of edges significantly differ from a random network with the same degree distribution. As such, it is similar to test statistics used in statistical hypothesis testing. In both areas, the statistic is used to find how significantly different the solution is from randomness. [33]

Modularity quantifies how significant the proposed community structure is by comparing the solution to a *null model*, i.e. a random network satisfying some property of the original network. This definition gives the modularity an important drawback, that the measure only can be compared to community structures found in the same network. Despite this, the measure is the most commonly used method to solve problems with ranking and finding the optimal number of communities. The modularity, $Q(\mathbf{c})$, of a community structure, $\mathbf{c} = (c_i)$, is calculated as

$$\begin{aligned} Q(\mathbf{c}) &= \frac{1}{2m} \sum_{i,j=1}^{n} (A_{ij} - N_{ij})\, \delta(c_i, c_j) \\ &= \frac{1}{2m} \sum_{i,j=1}^{n} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \delta(c_i, c_j), \end{aligned} \tag{2.6}$$

where c_i is the community of which node i is a member, k_i is the degree of node i, $\mathbf{A} = [A_{ij}]$ is the adjacency matrix, m is the number of edges, n is the number of nodes, and δ is the Kronecker delta function. [33]

The null model in the original definition of modularity is a randomly rewired network[5], such that the expected degree in the rewired network is

[5] By using such a null model, it is assumed that random networks should not have any community structure. Therefore this modularity measure compares a random solution with no communities to the communities found in the original network.

the same as the degree of the node in the original network. From this, we get the expected number (a result of the configuration model) of edges, $N_{ij} = k_i k_j / 2m$, between nodes i and j in the randomly rewired network. [5, 3]

As the modularity describes the quality of a partitioning of a network into communities, it is often assumed that a higher modularity corresponds to a better partitioning. Maximizing the modularity in community detection result is the same as maximizing the quality of the communities found. This optimization problem has been shown to be NP-complete and is therefore difficult to solve [34]. The modularity landscape is filled of many local maximum points and therefore it is difficult to find the global maximum [35]. Relaxed approximative methods are therefore needed to find an sub-optimal community structure.

Although widely used, modularity has problems with the so called *resolution limit*, i.e. the tendency to detect only larger features of the community structure in a graph. Often modularity optimization favors larger clusters over smaller clusters. One proposed solution is to introduce a *scaling factor* to enable some tuning for detection of larger or smaller features. [36, 4, 3]

Another drawback is the tendency of higher modularity in random networks than in real-world networks. This result in problems with evaluating different algorithms, as some algorithms which are good on random networks will perform worse on real-world networks. Some other measures for evaluating the quality of the community structure found in simulations are therefore needed. These community detection methods are discussed in *Chapters 2.1.3* and *5.4*. Other methods that do not rely on modularity maximization have also appeared, see Refs. [37, 38].

2.4 Algorithmic community detection

Historically, manual methods have been used to find community structures in collected data. Humans are often good at finding structures in small and sparse networks but manual methods are not practical for larger and denser network. To solve this problem, many methods used on computers have been proposed using different forms of iterative algorithms.

Despite this large effort, no completely satisfactory solution to community detection problem has yet to been devised. The main explanation for this is that community detection (maximization of modularity) is a NP-complete optimization problems[6], i.e. very time consuming and difficult to solve in an exact form. Instead some form of approximations, stochastic

[6] As such, the modularity optimization problem can be rewritten as other well-known NP-complete problems, i.e. *the satisfiability problem*, *the travelling salesman problem* and *the graph coloring problem*.

simulations, or heuristic methods are often used to find sub-optimal solutions.

Most community detection algorithms use at least one of these methods to find communities in networks and each simplification leads to different properties of the obtained solutions. For example, algorithms often have built-in tendencies to find communities of different sizes but also often find different community structures when applied to the same network. In general, more complex community detection methods are more accurate and robust in comparison with simpler and faster methods but are limited to small sparse networks. Since more complex methods have fewer and better motivated simplifications. As each method has different properties it is common to apply several different methods to the same problem and compare the results.

The oldest community detection methods proposed are based on the related problem of graph partitioning[7]. This problem is common in computer science and mathematics and has many applications. An important everyday application is to determine the correct division of computational effort on parallel computers[8]. Most graph partitioning methods are only able to divide a network into two parts and often find solutions with a very small cut set[9]. [5]

In this thesis, six different community detection methods are used to detect communities in so called candidate networks. Some of the more basic methods are based on well-known algorithms from computer science and results from linear algebra using centrality measures from the analysis of social networks. The more advanced methods are based on concepts from statistical mechanics, statistical processes, and the study of Bayesian networks. The discussion for each method could easily span many pages but here only a brief explanation is given of the ideas behind the algorithms.

The community detection methods discussed are only a small sample of the many methods that exist. Some of these methods show great promise in applications, e.g *clique percolation* and *synchronization*. Other methods have been developed for detecting communities in more complex graphs that are weighted, directed, or dynamic. For thorough discussions and comparisons of all the proposed community detection methods, see Refs. [4, 3].

2.4.1 Divisive algorithm based on betweenness

The first community detection algorithm proposed is based on the *betweenness centrality measure*, see *Appendix B*. A common interpretation of this

[7] See *Appendix A* for a discussion on graph partitioning problems.

[8] Often called the load balancing problem in practice, see e.g. some standard work on parallel computation or Ref. [5] for more information.

[9] A *cut set* is the set of edges that need to be removed from a graph to generate two disjoint components

centrality measure is the flow of information through a node or an edge. The idea is to eliminate the edge with the highest betweenness centrality in the network thereby separating the communities of the network. The algorithm for finding the community structure using the betweenness centrality method, is based on three steps as outlined in *Algorithm 4*.

Algorithm 4 Divisive edge betweenness

(i) find the edge with the highest betweenness centrality and remove it,

(ii) recalculate the edge betweenness for the edges in the network,

(iii) repeat from (i) until no edges remain.

The modularity is calculated after removing each edge, to find the modularity maximizing network clustering. This algorithm has a rather high complexity $\mathcal{O}(n^3)$. [39]

Some improvements have been proposed by various authors to decrease the high complexity. Perhaps the simplest improvement is to approximate the betweenness (introduced in Ref. [40]) by using a Monte Carlo estimation procedure thereby limiting the number of shortest paths to calculate. Another method to decrease the complexity of the algorithm is by changing the betweenness centrality measure to a simpler one. For example, in Ref. [32] it is suggested to use the number of short loops of edges that the particular edge is a part of.

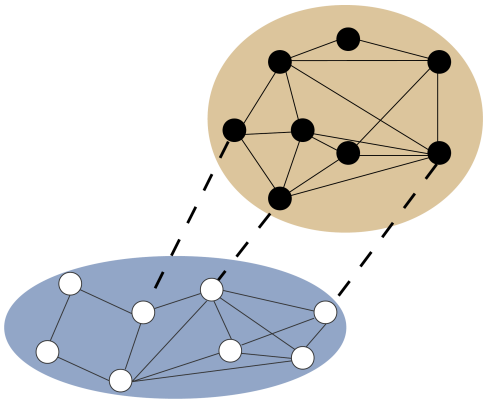

Figure 2.5: The application of the betweenness algorithm on a simple network with 16 nodes. Dashed lines indicate three edges with the largest betweenness centrality.

Example 4 (Edge betweenness in action)**.** To illustrate this idea, a simple situation is depicted in *Figure 2.5*, where two communities are identified in a simple network. Three dashed lines indicate edges with large betweenness centrality. We remove these edges in three repetitions, recalculating[10] the

[10]It is important to recalculate the betweenness measure, as it depends on the config-

betweenness for all edges after each removal. The resulting network contains two disjoint partitions which are taken as the two communities in the network.

2.4.2 Greedy agglomerative method

Another well-known community detection algorithm is based on a common technique for finding optimal solutions to graph problems. This method is called *greedy optimization* and surprisingly often finds the optimal solution e.g. in finding the shortest path or maximum flow through a network. The greedy method is a form of agglomerative hierarchical clustering which is outlined in *Algorithm 5*.

Algorithm 5 Greedy agglomerative method

(i) start with n clusters (each containing one node),

(ii) compute the difference in modularity for each possible merge of clusters,

(iii) merge the two clusters that yield the largest increase (or no difference) in modularity into one node,

(iv) repeating steps (ii)-(iii) until maximum modularity is reached.

This method bears a striking resemblance to the standard hierarchical clustering method, using modularity as the similarity measure. The method does not guarantee that the global maximum is found, but it has a low complexity $\mathcal{O}(nlog^2n)$. Thus this algorithm is more suitable for large graphs than the previously discussed divisive algorithm. [41]

Some improvements have been discussed by various authors, e.g. in Ref. [42] a factor is introduced to reduce the bias of the algorithm to favor large clusters. In Ref. [43] the authors suggest moving a single vertex after the original algorithm to another cluster, which is shown to find higher modularity closer to the global maximum.

2.4.3 Spectral method

The third method is based on the singular decomposition of matrices and the corresponding spectral graph partitioning problem, see *Appendix A*. Begin by defining the modularity matrix, $\mathbf{B} = [B_{ij}]$, as

$$B_{ij} = A_{ij} - \frac{k_i k_j}{2m}, \tag{2.7}$$

uration of edges, if one removes one edge the information will perhaps flow through a longer path. This will result in that the edges composing the alternative route will have an increased betweenness centrality.

where $\mathbf{A} = [A_{ij}]$ is the adjacency matrix, k_i is the degree of node i, and m is the number of edges in the network. Let $\mathbf{s} = (s_i)$, where $s_i \pm 1$, denote the cluster that node i belongs to with only two clusters allowed (-1 for cluster 1 and 1 for cluster 2). The modularity (2.6) is written using a singular matrix decomposition as

$$Q = \frac{1}{4m}\sum_{i,j=1}^{n} B_{ij}s_i s_j = \frac{1}{4m}\mathbf{s}^\top \mathbf{B}\mathbf{s} = \frac{1}{4m}\sum_{i=1}^{n}(\mathbf{u}_i^\top \cdot \mathbf{s})^2\beta_i, \tag{2.8}$$

where $\mathbf{u}_i = (u_{ij})$ are the eigenvectors for the modularity matrix and β_i is the eigenvalue of $\mathbf{B}$ corresponding to modularity matrix eigenvector $\mathbf{u}_i$. Using the leading eigenvector and optimizing the modularity with respect to s_i, one can show that the optimum value of the modularity is given by

$$\mathbf{s}^\top = \sum_{j=1}^{n} s_j u_{1j}, \tag{2.9}$$

this allows for maximizing the modularity by solving $s_j u_{1j} > 0$ for each $j = 1, 2, \ldots, n$. The resulting vector, s_j, indicates the community to which node j belongs. To cluster each meta-cluster into smaller clusters the steps are redone again for ΔQ, the corresponding change in modularity due to a division of the current clusters, instead of Q in (2.8), until the modularity begins to decrease. [44, 5]

This algorithm has a complexity of order $\mathcal{O}(n^2)$ and is often complemented with moving a single node between communities, if this increases the modularity, compare the *Kernighan-Lin algorithm* in *Appendix A*. [3]

2.4.4 Spin glass algorithm

The two most well-known models of magnets in statistical mechanics are the Ising and Potts models. The *Ising model* allows for two different spin states (up and down) whereas the more general *q-Potts model* uses q different spin states. [45, 46, 47]

These states are placed in either a random or uniform lattice structure, in which each spin variable interacts with other randomly selected or neighboring states. These interactions (couplings) between different spin states can be either ferromagnetic, anti-ferromagnetic, or a mix of both types. In ferromagnetic (antiferromagnetic) couplings, the spins tend to align (disalign) with their neighbors to minimize the total energy.

A *spin glass* is a generalization of the Ising and Potts models, i.e. a model of magnets with disorder and frustration. The disorder is created by setting the interactions between spins randomly. A spin state could therefore interact with some spins as a ferromagnet and others as an antiferromagnetic. In this model frustrations occur when these interactions do not match, i.e. where spins are aligned such that the coupling is not fulfilled.

Example 5 (Frustration in a lattice)**.** In *Figure 2.6*, a simple case is depicted using the Ising model with nearest-neighbor interaction. The spin states are indicated by the texture of the nodes (open for upward spins and closed for downward spins), the plus and minus signs indicate the interaction between spin states (+ for ferromagnetic). Frustration occurs when it is impossible to arrange the spin states to satisfy the interactions, as is seen in the area marked by the dashed circle.

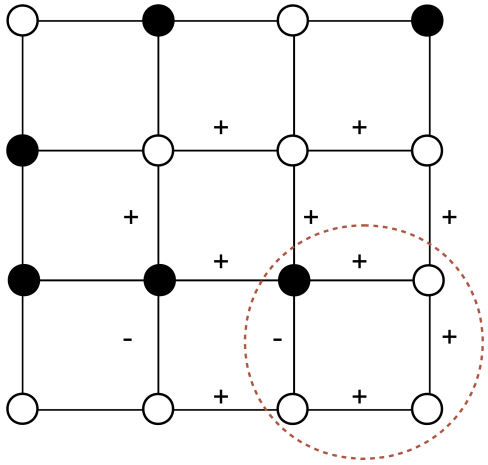

Figure 2.6: An Ising spin glas model with frustration. The interactions on the lattice structure do not allow for a perfect spin distribution. Not all the interactions can be fulfilled at the same time, as seen in the red circle. '+' indicate ferromagnetic interaction (align) and '-' anti-ferromagnetic (disalign).

A common problem in spin glass models is to find the ground-state energy, the minimum total energy which for this system is described by

$$\mathcal{H} = -\sum_{(i,j)} J_{ij}\delta(s_i, s_j), \tag{2.10}$$

where J_{ij} is a (random) coupling constant for the spins at positions (i, j), $\delta(\cdot)$ is the Kronecker delta function, and $s_i \in \{1, 2, \ldots, q\}$ is the value of node i. It is possible to show that finding the ground-state energy for some spin glass models is an NP-complete problem. Using the spin glass formulation these problem can be solved in relaxed form by *stimulated annealing*, which is a heuristic optimization method. [48, 49]

In the community detection problem, Ref. [16] proposes a q-Potts spin glass model to identify community structures in networks. The proposed *Hamiltonian* for the system is

$$\mathcal{H} = -J\sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij}\delta(\sigma_i, \sigma_j) + \gamma\sum_{k=1}^{q}\binom{s_k}{2}, \tag{2.11}$$

where J and γ are *coupling parameters*, $\delta(\cdot)$ is the Kronecker delta function, and s_k is the number of spins in state k. The first term favors many edges inside communities (i.e. few edges between communities) and the second term favors an equal distribution of nodes in communities.

The energy minimum is found by using *simulated annealing*[11], where all the initial spins are assigned randomly and spins are changed with some probability depending on the change of total energy. [48, 49]

The ratio γ/J is a *scale factor* that allows for tuning of the community detection for larger or smaller communities. In the following simulation experiments the coupling parameters are selected as $J = \gamma = 1$, thus making existing and non-existing edges equally important[12]. Simulated annealing optimizing is complex and therefore the algorithm is of rather high complexity, $\mathcal{O}(n^{2+\theta})$ with $\theta = 1.2$ on sparse network. [16]

2.4.5 Random walk on networks

This method for detecting communities is based on the idea that a random walker[13] should spend more time within communities than between them, since a community should have a higher intra-connectivity within the community than inter-connectivity between communities.

The algorithm uses agglomerative hierarchical clustering with Ward's method to merge the different nodes into clusters. The merge of existing nodes/clusters into some cluster C_k, is selected to be i and j which minimizes the following expression

$$\Delta R(C_i, C_j) = \frac{1}{n} \frac{|C_i||C_j|}{|C_i| + |C_j|} r^2_{C_i C_j}, \tag{2.12}$$

with the distance, $r_{C_i C_j}$, between two nodes/clusters i and j defined as

$$r_{C_i C_j} = \left[\sum_{l=1}^{n} \frac{(P^t_{C_i l} - P^t_{C_j l})^2}{k_l} \right]^{1/2}, \tag{2.13}$$

where $P^t_{C_{il}}$ is the probability to travel from community C_i to node l in t steps and k_l is the degree of node l. In the simulation study, the probabilities are estimated using a random walk with four steps. After a merge, the different quantities are recalculated and the algorithm is repeated until all nodes/clusters have been merged. The appropriate number of final clusters is determined by the maximum in the modularity. The expected complexity of this algorithm is $\mathcal{O}(n^2 \log n)$. [50, 3]

[11] With initial and final temperatures $T_0 = 1$ and $T_t = 0.1$, and cooling factor 0.99.

[12] These parameter settings limit the method to find only communities larger than $\sqrt{m}$.

[13] As discussed in *Appendix A*, random walks can be used to explore some structures of social networks.

2.4.6 Label propagation

The last method is called *label propagation* and operates by assigning labels to each node in the network and letting the labels propagate. Each node is initially given an unique label which is changed depending on the most common label of the neighboring nodes. This is equivalent to the requirement that a node should be connected more densely to nodes in the same community than to other nodes. Therefore the most common label of neighboring nodes should be the community in which the node is a member. The labels propagate through the network stochastically using an iterative algorithm outlined in *Algorithm 6*.

Algorithm 6 Label propagation

(i) initialize the labels at all nodes in the network by setting the label, $\ell_i(t = 0) = i$, for each node i and set $t = 1$,

(ii) sample a node without replacement $i \in V(G)$, let

$$\ell_i(t) = \mathsf{cm}\left[\ell_{i1}(t), \ldots, \ell_{im}(t), \ell_{i(m+1)}(t-1), \ldots, \ell_{ik}(t-1)\right], \tag{2.14}$$

where $\mathsf{cm}(\cdot)$ is a function returning the most frequent label, breaking ties randomly,

(iii) if every node has a label which is the same as the label of the maximal number of their neighbors, then stop. Otherwise set $t = t + 1$ and repeat from (ii).

This algorithm produces different community structures for each run, due to its stochastic nature in both selecting the updating order for nodes and the random breaking of ties. Therefore, the authors originally proposing this method in Ref. [11] merge five runs of the algorithm and present this merged structure as the communities found in the network. The merging method used is quite dissimilar to the methods proposed in this thesis, but shows nevertheless that the method generates good results when aggregating a few runs. Merging runs of this algorithm is further discussed in *Chapter 6.1* as a possible application for the proposed merging methods.

It is worth mentioning this algorithm is a zero-temperature version of the spin glass method using q-Potts model. As a result of the modularity (energy) landscape, it is difficult to find the global maximum using this method although the method is very fast and has almost linear complexity, $\mathcal{O}(m)$. Note that the number of edges in the network, m, often is larger than the number of nodes, n, but $m \leq \frac{n}{2}(n-1) < n^2$. Resulting in a rather low complexity compared to other algorithms of complexity $\mathcal{O}(n^2)$, but often slower than algorithms with complexity $\mathcal{O}(n \log n)$. [11, 3]

Chapter 3

Uncertain and imperfect networks

The first step in detecting communities in networks is to formulate a graph structure consistent with the available information. Assume for the remainder of this thesis that network data is gathered using some methods that allows for estimating the uncertainty in the observed data. It is further assumed that a large portion of the data is uncertain and try to utilize the data in the best possible manner.

In previous work, data sets are often considered certain and the uncertainty is removed by using one of three alternatives: (i) include all edges and nodes found, thereby possibly adding false nodes and edges into the network, (ii) remove all uncertain edges and nodes, thereby risking problems with missing edges and sparse networks, (iii) include all edges with an existence probability higher than some limit value, $\hat{p}$, thereby removing a number of edges keeping the remaining as certain edges. The third alternative could be successful in applications if $\hat{p}$ is known but it is probable that it depends on the underlying network structure. Thereby it is not easily estimated prior to simulation runs.

It is not difficult to realize that the three alternatives given generate different network structures and hence also different detected communities. In all the possibilities given, some useful network information is lost therefore risking a suboptimal detection of the community structure. An additional alternative is to use the probabilities as weights together with a community detection method supporting weighted graphs. The problem with this is that high-order network structures are neglected as a community is determined by more than the neighbors of each node independently. This approach is later analyzed in *Chapters 4.3.1* and *6.2*.

In this thesis, an alternative method is proposed that uses an observation model together with a sampling method. This method does not approximate the uncertain network as a certain network, but rather keeps all information

found to detect the best community structure possible.

We continue this chapter by discussing an observation model of networks and discuss some possible future generalizations of the model to include more interesting difficulties encountered in practical applications. Some frameworks to quantify and combine several different sources of information are introduced to estimate imperfect networks: (i) *probability theory*, (ii) *Dempster-Shafer theory*, and (iii) *Fuzzy-set theory*. The outputs of these methods are added to the network structure, from which an ensemble of consistent networks is created. In *Chapters 4* and *5*, we discuss how to use this information to detect the community structure in the uncertain network.

3.1 Observation model

The *observation model* is a formalization of the problem with observation of an underlying network by the use of other related networks. Often it is not possible to observe the network of interest directly and therefore a *proxy network* has to be used as an approximation. A classical example of this is using the communication network between people as a proxy of different kinds of relations. People with stronger connections and deeper relationships are assumed to communicate more often or in some detectable characteristic pattern.

Formalizing this, we assume that the *real network*, f, is not directly observable, but similar to a proxy network, g, to estimate the underlying network. By using this other network to describe the network of interest several different problems are encountered, e.g. finding edges in the observed network which does not exist in the real network etc. Assume that an *edge existence probability*, $\mathbb{P}(g_{ij})$, i.e. the probability that an edge exist (or does not exist) in the observed network, g, between nodes i and j can be found as

$$\mathbb{P}(g_{ij}) = FP + TP = \mathbb{P}(g_{ij}|\neg f_{ij}) + \mathbb{P}(g_{ij}|f_{ij}), \tag{3.1}$$

$$\mathbb{P}(\neg g_{ij}) = TN + FN = \mathbb{P}(\neg g_{ij}|\neg f_{ij}) + \mathbb{P}(\neg g_{ij}|f_{ij}), \tag{3.2}$$

where FP(N) denotes *False Positive* (*Negative*), TP(N) denotes *True Positive* (*Negative*), and $\mathbb{P}(\cdot)$ is the observation probability. The probability, $\mathbb{P}(g_{ij})$, should in some aspect indicate the uncertainty of the information regarding the edge, g_{ij}. High probabilities indicate strong evidence for the hypothesis that the edge exist in the real network. Smaller probabilities indicate vague or contradicting evidence. This is an observation model which links the observed with the real network and formalizes the uncertainty in using this approximation.

Example 6 (Uncertain network)**.** The simplest example of how to find a quantitative measure of the edge uncertainty is to analyze information

flowing over some network. Assume that e-mails are gathered and analyzed, resulting in labeling regarding content. A measure of the certainty of an edge in the network is found by determining the fraction of e-mails with relevant subjects of the total amount of exchanged e-mail. This proxy network could be visualized in the same manner as a small example shown in *Figure 3.1* with some edge existence probabilities, $E_{ij} = \mathbb{P}(g_{ij})$.

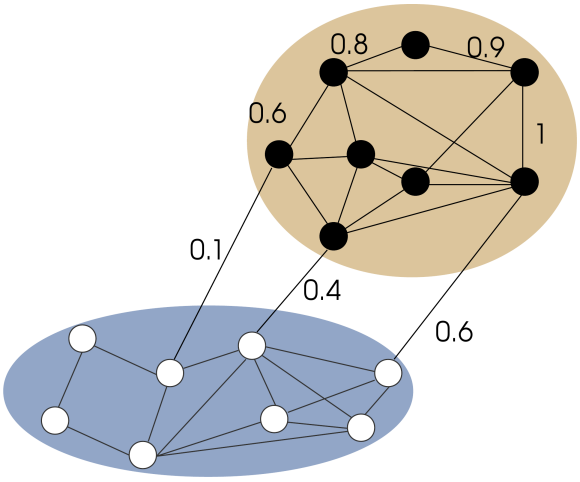

Figure 3.1: A small uncertain network with edge existence probabilities.

False positives and negatives have been given much attention in previous works. False negatives are called *missing edges* in network science and the impact of them is investigated in e.g. Ref. [51]. It is well-known that missing edges may cause severe problems leading to radically altered community structures, especially in sparse networks. False positives are called *false edges* and are interpreted as some noise introduced in the network. These can alter the network structure and the detected communities in much the same manner as missing edges. In practical applications these problems are common as *proxy networks* often are used to estimate the underlying real network structure. Uncertain networks in combination with missing/false edges form *imperfect networks* defined in *Definition 4*.

Definition 4 (Imperfect networks)**.** An *uncertain network* is a graph, $G(V, \mathbf{E})$, where V is a set of nodes and $\mathbf{E} = [E_{ij}]$ is some *edge existence probability matrix* with $E_{ij} = \mathbb{P}(g_{ij})$ as the probability that an edge exists between nodes i and j. An *imperfect network* is an uncertain network with missing and false edges, i.e. edges that do (not) exist in the real (observed) but not in the observed (real) network.

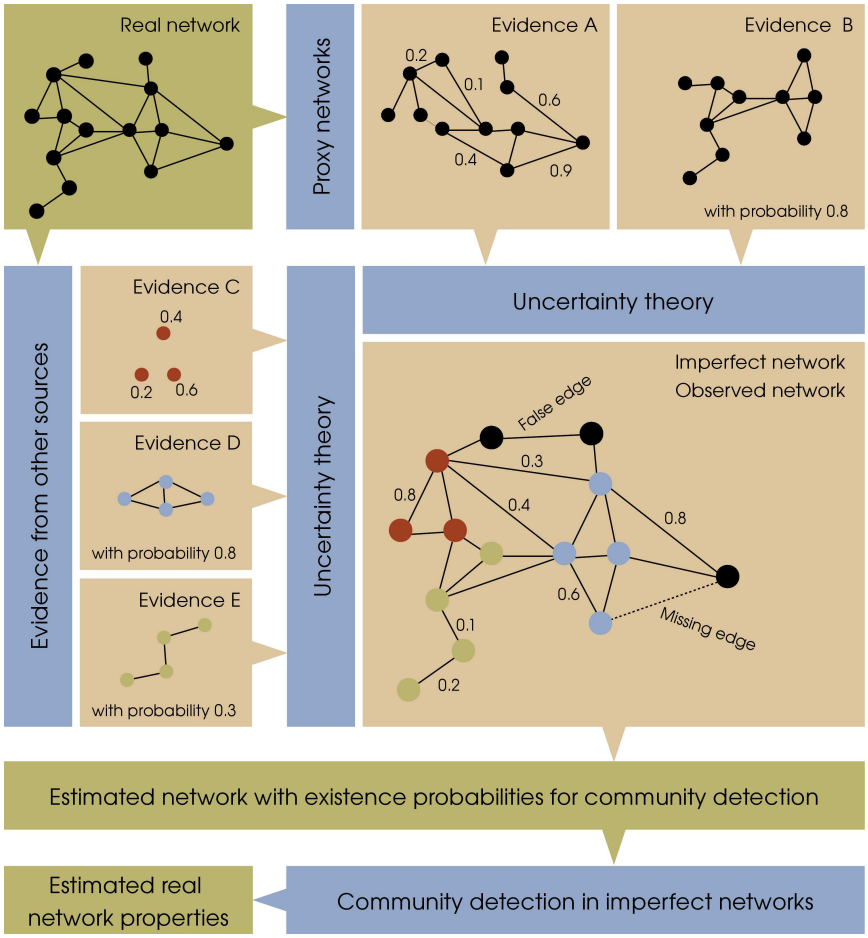

Figure 3.2: Schematic representation of the imperfect network and the relation between the observed and real networks. Given an unobservable real network, we can estimate an imperfect network by combining a number of evidences using uncertainty theory. The result is an estimated network structure with existence probabilities which can be analyzed by the methods developed in this thesis.

3.2 Generalizing the observation model

Edges are not the only uncertain and imperfect objects found in imperfect networks, more complex structures and objects may also be uncertain, missing, or falsely included. Nodes can be modeled in the same manner as edges, i.e. the network may contain uncertain, missing and false nodes. In this case, false nodes could mean that two nodes in the observed network are really only one in the real network. The probabilities can also describe different network structures, e.g. triangles, n-cliques, paths, and trees. Adding evidence regarding these structures can improve the estimated network structure found using observed network data.

We can further use additional sources of information and not only observed networks. As shown in *Figure 3.2*, it is possible to combine many different kinds of observations to form an imperfect network. Evidence A and B are some kind of observed network structures that are assumed to be similar to the real network. These *proxy networks* can be found from e.g. communication networks, observational data, or information gathered from witnesses and other sources. These observed networks can be modeled either by estimating the probability that edges exist or the probability that the entire structure exist.

Evidence C contains information about some nodes in the network and the probability that they exist and are unique. As they form a triangle (3-clique) it could be possible that those three observed nodes are only one node in the real network. Evidence D and E are examples of probabilities that certain structures exist in the network, e.g. subgraphs of a certain type or paths. This is an advantage as social networks often contain e.g. many triangles between nodes, as friends of a person often met and form friendships. This is called *triadic closure* and is well documented in sociological experiments as well as in analysis of real-world social network information. [52]

Using the framework built in the remaining part of this chapter, it is possible to combine different sources of information to estimate an imperfect network. The resulting structure from a combination of evidences is an imperfect network with existence probabilities for edges, nodes, and structures as well as missing and false edges.

3.3 Probability theory

The classical method to quantify uncertainty is *Probability Theory* (PT) using Bayesian inference. The probability, p, that some event X has occurred is determined by a probability function, $p = \mathbb{P}(X)$. The main drawback of probability theory is that it is binary and can therefore *only* be used to calculate probabilities that an event has occurred or not. This also means that if an event has not happened, its complement must have. The axiomatic definition of the probability function, $\mathbb{P}(\cdot)$, is given in *Definition 5*. [53]

Definition 5 (Probability function)**.** A *probability function*, $\mathbb{P}(X)$, for a discrete random variable with an *event space*, Ω, satisfies the following axioms

$$\mathbb{P}(\emptyset) = 0, \tag{3.3}$$
$$\mathbb{P}(\Omega) = 1, \tag{3.4}$$
$$\mathbb{P}(A \cup B) = \mathbb{P}(A) + \mathbb{P}(B), \tag{3.5}$$

when A and B are some events, $A, B \subseteq \Omega$, which are *independent*, $A \cap B = \emptyset$.

The last axiom (3.5) states that probabilities are additive for independent events. This is a core feature of probability theory which allows for the simplification of probability structures. By conditioning variables large simplifications are often obtained. This is also the corner stone of Bayesian interference which is based on Bayes' theorem, presented in *Theorem 1*. [54]

Theorem 1 (Bayes' theorem)**.** *Let $\Omega_{X\times Y} = \Omega_X \times \Omega_Y$ be a joint event space with $X \in \Omega_X$ and $Y \in \Omega_Y$ some random variables then*

$$\mathbb{P}(X|y) = \frac{\mathbb{P}(y|X)\mathbb{P}(X)}{\sum_{x\in\Omega_X} \mathbb{P}(y|x)\mathbb{P}(x)}, \tag{3.6}$$

where $\mathbb{P}(X|y)$ is the posterior probability function, $\mathbb{P}(y|X)$ the likelihood function, and $\mathbb{P}(X)$ the prior probability function.

Bayes's theorem states that given some prior probability (function) it is possible to incorporate new evidence using a likelihood function. This result in an updated posterior probability which include the additional information. The application for this is often to combine evidence from different sources. Let y and z be some evidence observed, then the posterior probability of X is found using

$$\mathbb{P}(X|y,z) = \frac{\mathbb{P}(y,z|X)\mathbb{P}(X)}{\sum_{x\in\Omega_X} \mathbb{P}(y,z|x)\mathbb{P}(x)}. \tag{3.7}$$

This allows for merging evidences regarding the uncertainty of the network structure to generate an edge existence probability. [54]

3.4 Dempster-Shafer theory

The theory of evidence, or *Dempster-Shafer Theory* (DST), is a generalization of probability theory which relaxes the axiom of additivity and introduces a different method for merging evidence from multiple sources. The theory also allows for the construction of intervals with upper and lower probabilities to include the uncertainty in merged conflicting evidence. DST is popular in some areas of artificial intelligence, decision support, and data fusion. Instead of probability functions, the theory of evidence uses belief and plausibility functions, defined in *Definition 6*. [55, 53]

Definition 6 (Belief and plausibility functions)**.** Assume that the *frame of discernment*, Θ, is a finite set and let 2^Θ denote the set of all subsets of Θ.

Suppose that the *belief function* $\mathsf{Bel} : 2^{\Theta} \to [0,1]$ satisfies the following[1]

$$\begin{aligned} \mathsf{Bel}(\emptyset) &= 0, && (3.9)\\ \mathsf{Bel}(\Theta) &= 1, && (3.10)\\ \mathsf{Bel}\left(A_1 \cup \ldots \cup A_n\right) &\geq \sum_i \mathsf{Bel}\left(A_i\right) - \sum_{i<j} \mathsf{Bel}\left(A_i \cap A_j\right) \\ &+ \ldots + (-1)^{n-1} \mathsf{Bel}\left(A_1 \cap \ldots \cap A_n\right)), && (3.11) \end{aligned}$$

where n is some positive integer and every collection, $A_1, \ldots, A_n$, is a subset of 2^{Θ}. The *plausibility function* $\mathsf{Pl} : 2^{\Theta} \to [0,1]$ is the dual of the belief function

$$\mathsf{Pl}(A) = 1 - \mathsf{Bel}(A^c), \tag{3.12}$$

where A^c denotes the complement of the subset $A \subset 2^{\Theta}$.

The *Belief function*, $\mathsf{Bel}(A)$, is interpreted as the belief that the truth lies in some subset of the set A. The *Plausibility function*, $\mathsf{Pl}(\cdot)$, measures the failure to doubt the truth and note[2] that $\mathsf{Bel}(A) \leq \mathsf{Pl}(A)$ for each $A \subseteq 2^{\Theta}$. Therefore the probability that the truth lie in A is given by the interval $[\mathsf{Bel}(A), \mathsf{Pl}(A)]$. [55]

Combining probabilities from different sources is the essence of information fusion. Dempster-Shafer theory allows for combining evidence in a more general manner than in probability theory, presented in *Theorem 2*. Although no prior probabilities and likelihood functions are needed, we need to have a *mass function*, $m(\cdot)$, that assigns probability masses to the frame of discernment, Θ. [53]

Theorem 2 (Dempster's rule of combination)**.** *Let $m_i : 2^{\Theta} \to [0,1]$, for $i = 1, 2$, be two (different) basic probability assignment functions on some frame of discernment, Θ, satisfying*

$$m(\emptyset) = 0, \quad \textit{and}, \quad \sum_{A \in 2^{\Theta}} m(A) = 1. \tag{3.13}$$

The combined belief of a subset $A \subseteq 2^{\Theta}$ is

$$m_{1,2}(A) = (m_1 \oplus m_2)(A) = [1 - K]^{-1} \sum_{B \cap C = A \neq \emptyset} m_1(B) m_2(C), \tag{3.14}$$

[1] The third condition (3.11) is related to the *inclusion-exclusion principle* in probability theory

$$\left| \bigcup_{i=1}^{n} a_i \right| = \sum_{i=1}^{n} |A_i| - \sum_{i<j} |A_i \cap A_j| + \ldots + (-1)^{n-1} |A_1 \cap \cdots A_n| , \tag{3.8}$$

which is similar to the condition if $\mathsf{Bel}(A_i) = |A_i|$ but has an equality where (3.11) has an inequality.

[2] This is due to that the belief is *sub-additive*, $\mathsf{Bel}(A) + \mathsf{Bel}(A^c) \leq 1$ and plausibility is *super-additive*, $\mathsf{Pl}(A) + \mathsf{Pl}(A^c) \geq 1$. Probability is additive and does in some situations coincide with the belief and plausibility, when $\mathsf{Bel}(A) + \mathsf{Bel}(A^c) = \mathsf{Pl}(A) + \mathsf{Pl}(A^c) = 1$.

where $m_{1,2}(\emptyset) = 0$ *and* K *is the amount of conflict between the two beliefs defined by*

$$K = \sum_{B \cap C = \emptyset} m_1(B) m_2(C). \tag{3.15}$$

By using the basic probability assignment function, $m(\cdot)$, the belief and plausibility is found by the following expression

$$\mathsf{Bel}(A) = \sum_{B \subset A} m(B), \qquad \mathsf{Pl}(A) = \sum_{B \cap A \neq \emptyset} m(B), \tag{3.16}$$

thus allowing for combination of evidence to construct probability intervals. By repeatedly adding evidence using this rule of combination, e.g. four evidences are combined using

$$m_{1,2,3,4}(A) = (((m_1 \oplus m_2) \oplus m_3) \oplus m_4)(A), \tag{3.17}$$

any number of evidence can be combined. DST can thus be used to quantify the uncertainty in a network by combining evidence from several different sources. [55]

Example 7 (Combination of evidence in PT and DST)**.** Assume that we are interested to determine whether two persons, i and j, are friends. Let a graph, $G(V, E)$, be a social network where each node, $v_i \in V$, is a person and an edge, $e_{ij} \in E$, indicates if they are friends. Further assume that we can not observe the real network, G, directly but instead can observe some other networks and evidence.

The first evidence, denoted A, indicates that the two persons have attended the same school and taken the same classes. Statistical surveys indicate that an average student is friends with about 70% of the other students attending the same classes. Summarizing the evidence A in terms of the Dempster-Shafer theory, we obtain

$$m_A(E_{ij}) = 0.70, \quad m_A(\neg E_{ij}) = 0, \quad m_A(\Theta) = 0.30, \tag{3.18}$$

where $m(\cdot)$ denotes some mass function and Θ is the frame of discernment[3]. In terms of probability theory, we determine that there is a 70% probability that the two persons are friends and a 30% probability that they do not. In the Dempster-Shafer view, we have some evidence supporting the hypothesis that they are friends, but no evidence to contradict it.

Another observed evidence, called B, is a friend of person i and j saying that they are not friends. We have somehow estimated the probability that this friend is lying as 20%. From this the mass function can be found for, $m_2(E_{ij})$, i.e. the probability that i and j are close friends

$$m_B(E_{ij}) = 0, \quad m_B(\neg E_{ij}) = 0.80, \quad m_B(\Theta) = 0.20. \tag{3.19}$$

[3]That is any combination of that they attend the same classes, are friends etc.

Finally these evidences can be combined using *Theorem 2* as

$$m_{A,B}(E_{ij}) = (m_A \oplus m_B)(E_{ij}). \tag{3.20}$$

The different values for the two mass functions m_A and m_B are summarized in *Table 3.1* and is used in the following calculations. We begin by cal-

m_A / m_B	$E_{ij} : 0$	$\neg E_{ij} : 0.80$	$\Theta : 0.20$
$E_{ij} : 0.70$	0	0.56	0.14
$\neg E_{ij} : 0$	0	0	0
$\Theta : 0.30$	0	0.24	0.06

Table 3.1: The combined mass function for any subset of Θ.

culating the amount of conflict, K, between the two mass functions. This is accomplished by summation of the mass functions such that $M \cap N = \emptyset$, in our problem this is all combinations of E_{ij} and $\neg E_{ij}$. Which are $m_A(E_{ij})m_B(\neg E_{ij})$ and $m_A(\neg E_{ij})m_B(E_{ij})$, because $E_{ij} \cap \neg E_{ij} = \emptyset$. The sum of conflict is therefore

$$K = m_A(E_{ij})m_B(\neg E_{ij}) + m_A(\neg E_{ij})m_B(E_{ij}) = 0.70 \cdot 0.80 = 0.56. \tag{3.21}$$

Proceeding with combining the evidence for the different possible cases using (3.14)

$$m(E_{ij}) = 0.32, \quad m(\neg E_{ij}) = 0.545, \quad m(\neg E_{ij}, E_{ij}) = 0.135. \tag{3.22}$$

We conclude that it is uncertain that whether two persons i and j are friends. Also there is a conflict, $K = 0.56$, between the two evidences. Using these mass functions, we find the belief, plausibility, and the corresponding probability interval

$$[\mathsf{Bel}(E_{ij}), \mathsf{Pl}(E_{ij})] = [0.320, 0.320 + 0.135]\,. \tag{3.23}$$

A probability interval for the hypothesis that the two persons are friends is thus found as $[0.320, 0.455]$. To improve upon this interval, we could use another evidence and the same rule of combination once more.

For probability theory, the corresponding result can be found using (3.7) and assuming equal prior probabilities $\mathbb{P}(x) = 0.5$, we have

$$\mathbb{P}(E_{ij}|A,B) = \frac{\mathbb{P}(A,B|E_{ij})}{\mathbb{P}(A,B|E_{ij}) + \mathbb{P}(A,B|\neg E_{ij})} \tag{3.24}$$

$$= \frac{0.70 \cdot 0.20}{0.70 \cdot 0.20 + 0.30 \cdot 0.80} = 0.368. \tag{3.25}$$

Assuming independent probabilities, the calculation yields the conditional probability as 0.40 for the hypothesis that they are friends is true.

3.5 Fuzzy-Set theory

Another generalization of probability theory is known as *Fuzzy-Set theory*, which was developed to counter the drawback in probability theory that the statement that some event has occurred must either be absolutely true or false. In comparison with classical sets, fuzzy sets have two main differences: (i) an element is allowed to exist in several sets simultaneously, (ii) fuzzy sets have fuzzy boundaries (which cannot be precisely defined) in comparison with *crisp* boundaries. [53]

Definition 7 (Fuzzy set)**.** Let Θ denote some finite set, then a *fuzzy set* A of Θ is defined by a *membership function*, $\mu_A : \Theta \to L$, where L is an ordered set of membership values and μ_A is the grade of membership of the element $\theta \in \Theta$ in A.

If the membership grade, $\mu_A(\theta) = 1$, it is said that θ certainly belong to A and if $\mu_A(\theta) = 0$ then θ certainly does not belong to A. For all other values of the membership grade, the element θ is a fuzzy member of A. This corresponds to the notion of probability and belief, which can be used to quantify the uncertainty of the network structure. In crisp set theory the membership grade, $\mu_A(\theta) \to \{0, 1\}$, i.e. only assumes the values 0 and 1. It is possible to merge different fuzzy sets by an aggregation operation, where the aggregation operator, $h : [0, 1]^n \to [0, 1]$, is used to combine n fuzzy sets as

$$\mu_A(\theta) = h(\mu_{A1}(\theta), \ldots, \mu_{AN}(\theta)), \tag{3.26}$$

where h is a bounded, monotonically increasing, continuous, symmetric function. A common aggregation function of some degrees of membership, $x_i = \mu_{Ai}(\theta)$, is the *generalized means*, $h^\beta(\cdot)$ and the *ordered weighted average*, $h_w(\cdot)$, defined as

$$h^\beta(x_1, x_2, \ldots, x_n) = \left[\frac{1}{n}\sum_{i=1}^{n} x_i^\beta\right]^{1/\beta}, \tag{3.27}$$

$$h_w(x_1, x_2, \ldots, x_n, \theta) = \sum_{i=1}^{n} w_i x_{(i)}, \tag{3.28}$$

where β is some real number, $\sum_i w_i = 1$ and $y_{(i)}$ denotes element i in the (decreasingly) sorted values of x_i. Thus the degree of membership can be used to quantify uncertainty in the network structure, if a suitable membership function can be found. [53]

Example 8. Continuing on *Example 7* using fuzzy-set theory, we can combine the evidence using the two aggregation functions introduced above. Using the notation of fuzzy sets, $x_1 = 0.70$ and $x_2 = 0$, as the evidence A

was 70% that they are friends and evidence B 0%. Using these two aggregations we find

$$h^{\frac{1}{2}}(x_1, x_2) = \sqrt{\frac{0.70 + 0}{2}} = 0.591, \tag{3.29}$$

$$h_{\frac{1}{n}}(x_1, x_2) = \frac{0.70 + 0}{2} = 0.350, \tag{3.30}$$

using $\beta = 1/2$ and equal weights for all sets, $w = 1/n$. Note that these two probabilities are somewhat different then those found by PT and DST. It is therefore not straight-forward to select the best method and aggregation methods.

Chapter 4

Detecting communities in imperfect networks

This chapter contains the proposed method that in combination with the framework introduced in *Chapter 3* enables for community detection in imperfect networks.

The complete method is outlined in *Figure 4.1*, with network data at the top and ending with an estimated community structure. In practical applications of this method, we would use the results presented in previous chapters to combine evidence from proxy networks and other forms of collected information to find an estimated network structure with existence probabilities. The result is called an imperfect network, which is a simplified schematic representation of *Figure 3.2* and the starting point of this chapter.

The steps in *Figure 4.1*, outlined in detail in this chapter are: (i) sampling candidate networks from the ensemble of consistent networks using (Markov Chain) Monte Carlo-methods. (ii) detecting candidate communities using standard methods discussed in *Chapter 2.4*. (iii) merging candidate communities, using methods based on ensemble clustering discussed in *Chapter 2.2*, into the most probable community structure of the uncertain/imperfect network.

In the last section of this chapter, it is demonstrated how the confidence of the resulting community structure can be estimated using data from the merging methods.

4.1 Sampling candidate networks

The introduced methods for quantifying and combining uncertainty in network information all result in a probability or a probability interval. These measures correspond to the degree of uncertainty that an edge, node, or structure exists in the network.

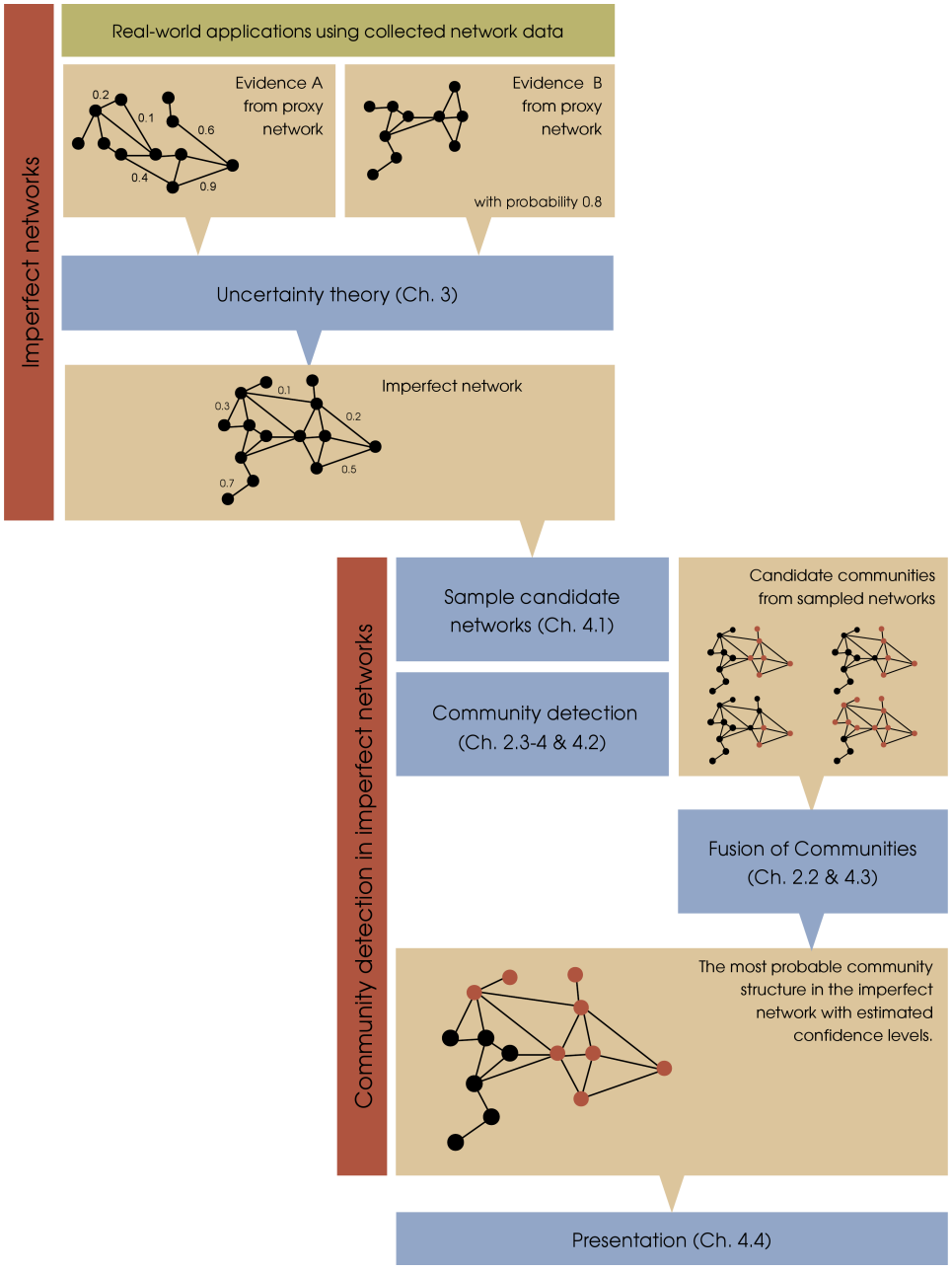

Figure 4.1: The proposed method to detect communities in imperfect networks.

An uncertain network, $G(V, \mathbf{E})$, is constructed from these existence probabilities, $\mathbf{E} = [E_{ij}]$, where e.g. E_{ij} is the probability that an edge exist between nodes i and j. We limit ourselves to probabilities describing uncertain edges in this thesis, however there are generalizations for other uncertain objects[1] and also using probability intervals[2] to quantify uncertainty.

The first step in detecting communities in uncertain networks is to generate an ensemble of networks that are consistent with the network information. Realizations are found using Monte Carlo-sampling with the existence probabilities, E_{ij}. The sampling[3] is performed using uniformly distributed random numbers to generate a symmetric matrix, $\mathbf{R} = [R_{ij}]$, where $R_{ij} \sim \mathcal{U}[0, 1]$ with $R_{ij} = R_{ji}$.

An edge is included in the graph if the random element in the corresponding matrix is less than or equal to the product of the corresponding elements of the edge probability matrix and the adjacency matrix, i.e. $R_{ij} \leq E_{ij}A_{ij}$. By simulating many random matrices, $\mathbf{R}$, a large ensemble of certain networks are generated, each consistent with the information in the uncertain network. The sampled networks are called *candidate networks* or *realizations* of the ensemble of consistent networks.

A simple situation using this sampling method in shown in *Figure 4.2*, where a small network is sampled. The ensemble of consistent networks consist of the four different possible permutations of the network structure. Each permutation exists with a proportion p_i and therefore also has the probability p_i of being sampled.

This example can be used as a basis for constructing candidate networks instead of sampling them from a larger ensemble. A perfect set of candidate networks can be constructed by including an edge in a fraction of the set which corresponds to the given probability. This is however impractical for larger networks which have a large number of possible permutations. In these cases, sampling is useful to estimate the properties of the ensemble without knowing all communities in the possible permutations.

[1] In a more general setting, the first step would be to sample the nodes to include in the network. The second step would be to sample edges between the included nodes and finally sampling higher-order structures to include in the network. Another method is by sampling structures first then nodes and edges. Despite the chosen order an imperfect network, $G(\mathbf{V}, \mathbf{E}, \mathbf{S})$, have existence probabilities for nodes, edges, and structures.

[2] Sampling from intervals is not covered in this section but perhaps the simplest method is to used the center point of the interval as a probability. Another viable method is Markov Chain Monte Carlo using the Metropolis-Hastings algorithm. The aim of the sampling method is to create realizations of the imperfect network thar as consistent as possible. This is accomplished by accepting the inclusion of edges that contribute to generating a network in which the e.g. the degree distribution is similar to the imperfect information.

[3] No variance reduction method has been applied in the sampling step. The inclusion of such a method may reduce the number of required samplings, n_s, and is an important subject for further study. The simplest method to apply is probably stratified sampling. By estimating the distribution of the edge probabilities one could adapt the sampling method to sample more or less certain edges etc.

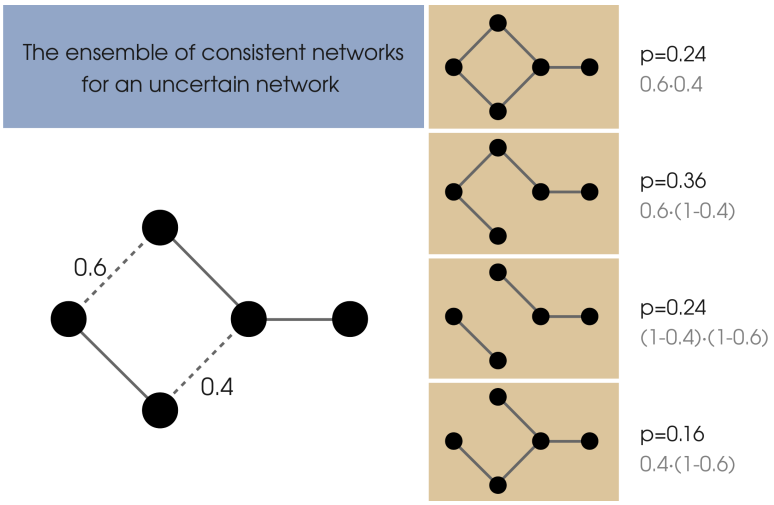

Figure 4.2: A small uncertain network and its corresponding ensemble of networks. The proportion of networks of a certain form in the ensemble is determined by the edge existence probabilities.

The idea is therefore to mimic the main features of the distribution of permutations in the ensemble by sampling many networks from the ensemble. So that if 50 networks are sampled from the ensemble, then approximately 12 should be of the first type of permutation in which no edge is removed and so on.

To simplify simulations, all nodes of degree one are removed during the community detection and merging steps. After those steps, the nodes with degree one are reinserted into to the network again. The removed nodes (all of degree one) are assigned to the same community as their neighbor. As a further simplification, the community detection is only performed on the largest component of the graph.

4.2 Detecting candidate communities

For each realization of the imperfect network some community detection algorithms[4] are applied. All these detection methods are based on the assumption that maximizing modularity is equivalent to maximizing the quality of the community clustering.

The community structures of each candidate network are referred to as *candidate communities*. The community structures for the n_s candidate networks of the uncertain network are summarized in a *membership matrix*,

[4]In the following simulation experiments, the implementations in the R-package igraph [56, 57] are used to detect communities and calculate network properties . The standard parameters for each algorithm, as presented in connection with the review of each method in *Chapter 2.4* are used.

$\mathbf{M} = [M_{ik}]$, where M_{ik} is the community in which node i is a member in candidate network $k = \{1, 2, \ldots, n_s\}$ and n_s is the number of samplings.

4.3 Merging candidate communities

The final problem is to merge the different community structures found in each candidate network into one community structure. Earlier works used some voting method or graph matching technique to accomplish this. In this thesis, we propose to instead construct a mean of the candidate communities using other types of merging methods. This is accomplished by merging nodes often found in the same cluster or by merging similar candidate communities.

Three different methods are proposed to accomplish this: (i) using a *two-step fusion of communities*, (ii) *node-based fusion of communities*, and (iii) *community-based fusion of communities*. The two latter methods are based on the ensemble clustering method for data sets, introduced in *Section 2.2*. An example of the resulting community structure is shown in *Figure 4.3*, as the merged community structure from 10 candidate communities using community-based fusion of communities.

4.3.1 Two-step Fusion of Communities

The first method, *Two-step Fusion of Communities* (TFC) can be used by community detection algorithms that handle weighted graphs[5]. The steps used in the method are outlined in *Figure 4.4* and begin with creating a (complete) graph, $G(V, \mathbf{F})$, with the same nodes as before and the *frequency matrix* element F_{ij} as the weight of the edge between nodes i and j. Each edge in the network indicates that two nodes have been placed in the same candidate community. The weight is the number of times the two nodes i and j are found in the same candidate community.

The next step is to detect communities in the weighted network, G, by applying a community detection algorithm. The method used for detecting the communities can be different in the two steps. The communities detected in the last step are taken as the communities of the merged candidate communities.

Another approach to this problem is to use the existence probability matrix, $\mathbf{E}$, instead of the frequency matrix, $\mathbf{F}$. This would save a lot of computational effort to sample candidate networks and to determine the community structure of each. This approximation however neglect all higher order network structures that contribute to community structures. Not only

[5]The following previoulsy introduced community detection methods are able to handle weighted networks: greedy agglomerative method, spin glass, random walk on the network, and label propagation.

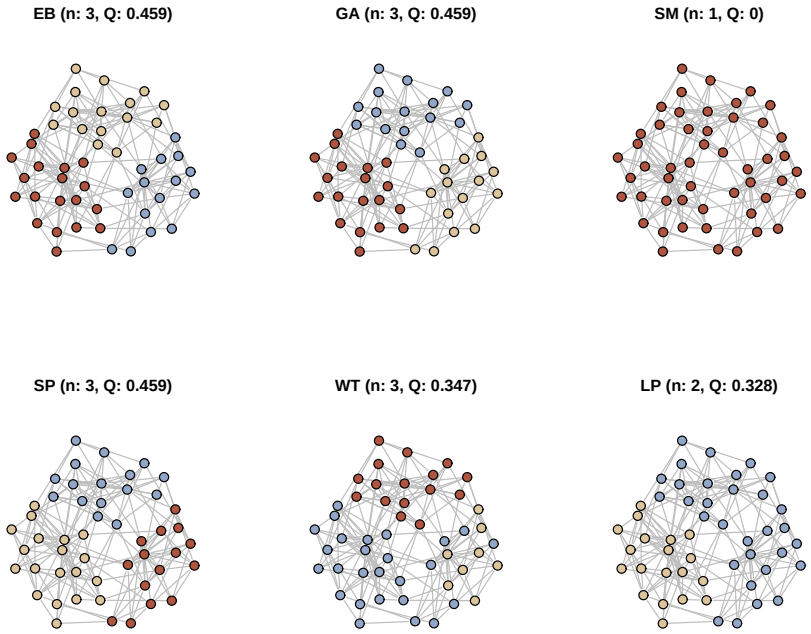

Figure 4.3: The communities found in the network Synthetic 1 (see *Chapter 5.1*) using CFC. Q denotes the modularity, n the number of detected communities, and the colors indicate the community to which the node belongs. The correct community structure is shown in *Figure 5.2*.

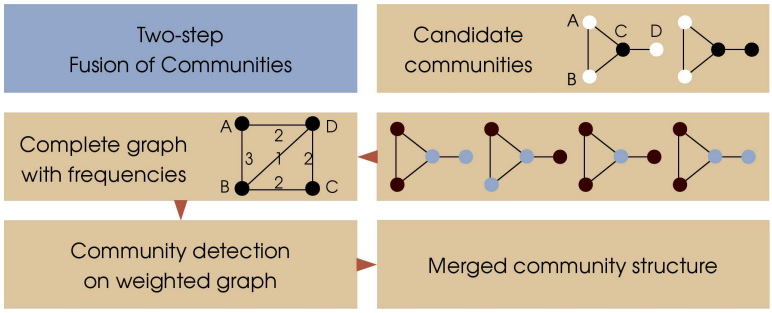

Figure 4.4: Two-step Fusion using a weighted community detection method.

simple structures such as edges are important and this approximation is thereby probably not valid. This is investigated further in the following simulation experiments.

This method depends on the performance of the community detection algorithms used. If slower but more accurate methods are used, the corresponding merge result is more accurate but TFC is slower. As in general, faster algorithms for community detection find the merged communities faster but are more inaccurate. The complexity of TFC is the same as the complexity of the community detection method used.

4.3.2 Node-based Fusion of Communities

The second method, *Node-based Fusion of Communities* (NFC) is an extension of Instance-based Ensemble Clustering presented in Refs. [14, 15] and previously discussed in *Section 2.2*. NFC is similar to TFC but uses agglomerative hierarchical clustering to group the nodes of the weighted graph into clusters instead of applying a community detection algorithm to the graph.

The NFC-method is outlined in *Figure 4.5* which begins with the construction of a complete graph $G = (V, \mathbf{F})$ from the candidate communities. In the new graph, nodes are the same as in the original network but the edges have a different interpretation. As in TFC, the new weighted edges correspond to the frequency of instances when two nodes have been grouped together in the same candidate community. This differs from the other interpretation that the edge weight correspond to the probability that an edge exist between two nodes.

The nodes are clustered using agglomerative hierarchical clustering with the edge weight as the similarity between two nodes. Thus nodes often found in the same candidate cluster are grouped together by the hierarchical clustering method. Note that the difference compared to TFC is that the NFC can handle the problem with higher-order structures mentioned above.

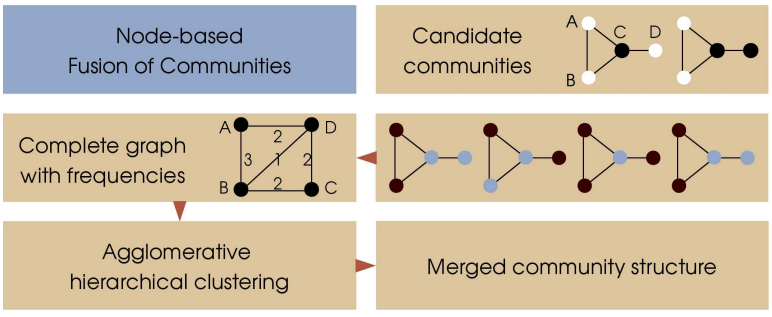

Figure 4.5: Node-based Fusion using agglomerative hierarchical clustering.

The main difference with this method compared to IBEC is the process in which the frequency, F_{ij}, is recalculated after each merge. The fre-

quency between the merged nodes (cluster) l and the other nodes or clusters, $v_1, v_2, \ldots, v_{n_l}$, is found by

$$F_{k,l} = \left| \bigcap_k M_{kl} \right|, \tag{4.1}$$

where the *membership matrix*, $\mathbf{M} = [M_{ik}]$, where M_{ik} is the community in which node i is a member in candidate network $k = \{j, i_1, \ldots, i_{n_l}\}$. That is, $F_{k,l}$ is the number of occurrences where all nodes (in both clusters) are in the same candidate cluster.

This linkage rule incurs some loss of information about individual nodes as they are clustered together and information about the similarity of individual nodes are lost. The result of the hierarchical clustering algorithm is a dendrogram and a list of merges. The clustering corresponding to the maximum modularity is taken as the communities found in the merged candidate networks. The complexity of NFC is determined by the hierarchical clustering algorithm and therefore is $\mathcal{O}(n^2)$.

4.3.3 Community-based Fusion of Communities

The third method is based on Community-based Ensemble Clustering and singular value decomposition of the similarity matrix, $\mathbf{S} = [S_{ij}] = [\mathsf{sim}(i, j)]$. This method is quite dissimilar to TFC and NFC because it is based on merging similar clusters and not similar nodes.

The full method for merging candidate communities using *Community-based Fusion of Communities* (CFC) is outlined in *Figure 4.6*. The first step is to construct a complete graph $G = (V, \mathbf{S})$, with a set of nodes, $V(G)$, consisting of each candidate cluster, $\mathcal{C}_i$. The similarity matrix, $\mathbf{S} = [S_{ij}]$, is calculated using the cosine measure

$$S_{ij} = \frac{|\mathcal{C}_i \cap \mathcal{C}_j|}{\sqrt{|\mathcal{C}_i|\,|\mathcal{C}_j|}}, \tag{4.2}$$

where $\mathcal{C}_i$ and $\mathcal{C}_j$ are candidate clusters for some number of clusters R such that $i, j = 1, \ldots, R$ and $i \neq j$. This measure is chosen due to its linearity and that it is scaled to unity. The similarity matrix is expanded using singular value decomposition, in the same manner as discussed in *Algorithm 2.2.2*. This is done by first normalizing and calculating the k largest eigenvectors of the similarity matrix, $\mathbf{L}$. These are placed as rows in a new matrix which is normalized and partitioned using k-means clustering.

Each node, from the original network, is assigned to the community of which it is most often a member of by voting and breaking ties randomly. This is repeated for all possible values of $k \leq R$, to find the final community structure of the network, i.e. the one that maximizes the modularity.

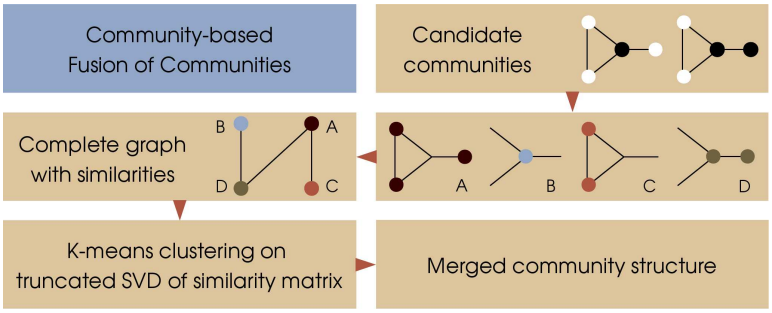

Figure 4.6: Community-based Fusion using k-means clustering on the truncated singular value decomposition of the adjacency matrix.

The complexity of CFC is determined by the eigenvector calculation and therefore is approximately $\mathcal{O}(n^3)$.

An advantage of the applied approximation (using only $k < R$ eigenvectors) is the removal of noise in the data, as only the most distinctive features are used in the merging process. In theory, this make the CFC method more robust than the other two proposed methods which use all or at least most of the network data.

Heuristic approximation

The CFC-algorithm needs to compute the meta-clustering for all $k < R$, which is quite time consuming. Therefore some techniques from the related statistical method *Principal Component Analysis*[6] (PCA) are applied to this *Singular Value Decomposition* (SVD) problem. The method applied in this approximation builds on an automatic procedure to find the *elbow point* of the scree plot (a plot of the decreasingly sorted eigenvalues, see *Figure 4.7*), which is the point after which all eigenvalues are of approximately the same size. In PCA, the elbow point correspond to the optimal number of principal components to use.

To automate this process, we transform the detection of the knee point into a Maximum Likelihood (ML) problem by following Ref. [58]. The estimated optimal number, $\hat{q}$, of eigenvectors (or principal components in PCA) to include is given by

$$\hat{q} = \max_k l_q(k), \tag{4.3}$$

where k is the number of clusters and $l_q(k)$ is a profile log-likelihood function

$$l_q(q) = \sum_{i=1}^{q} \log f\left(d_i; \mu_1, \hat{\sigma}(q)\right) + \sum_{j=q+1}^{n} \log f\left(d_j; \mu_2, \hat{\sigma}(q)\right), \tag{4.4}$$

[6] In PCA, we construct a new basis using uncorrelated linear combinations of the given sample from some random variables. This maximizes the explained variance of the sample and where each linear combination is orthogonal to the others. [21]

where d_i is the ith largest eigenvalue for a network of n nodes where $f(\cdot)$ is the density function of the normal distribution $\mathcal{N}(d; \mu_j, \sigma^2)$. The interpretation of (4.4) is that maximizing the log-likelihood the eigenvalues are divided into two normal distributions with different means with $\mu_1 > \mu_2$. The first distribution are the larger eigenvalues and the second distribution are all the remaining eigenvalues which are of approximately equal size. The optimal allocation of eigenvalues are given by maximizing (4.4) assuming that there are enough eigenvalues to satisfy the assumption of the *central limit theorem* (which is the case in this thesis as $R \geq 30$, i.e. the common rule-of-thumb).

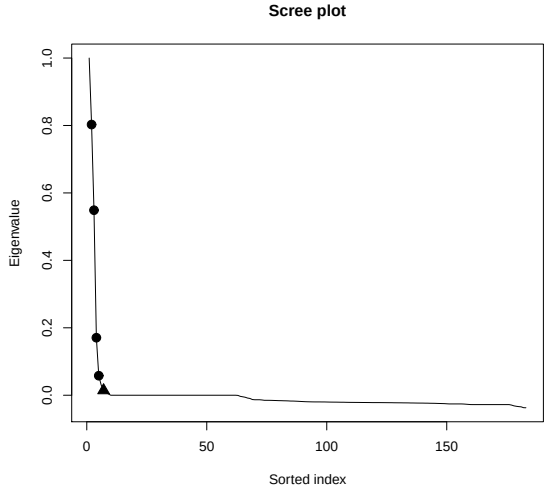

Figure 4.7: The scree plot of 150 eigenvalues from a cosine similarity matrix constructed using 50 candidate clusters on the Karate network with $p_t = 0.5$, see *Chapter 5* for details on the L-distribution. The filled circle indicate the estimated set of possible number of eigenvectors as found by the approximation. This filled triangle indicate the traditional elbow point in the scree plot. Note that this is only done to limit the search as modularity maximization is used to determine the optimal number of communities in the set of possible number of eigenvectors.

The mean value, μ_j is estimated by the sample mean and the variance, σ^2, by the pooled sample standard deviation

$$\mu_1 = \frac{1}{q}\sum_{i=1}^{q} d_i, \quad \mu_2 = \frac{1}{n-q}\sum_{i=q+1}^{n} d_i, \tag{4.5}$$

$$\hat{\sigma}^2 = \frac{1}{n-2}\left[(q-1)s_1^2 + (n-q-1)s_2^2\right]. \tag{4.6}$$

This computation is simpler than determining the similarity and computing the k-means partitioning for all $k < R$. The ML-method is used to narrow the search, by limiting the set of possible number of eigenvectors to $k \in$

$\{\lceil(1/2)\hat{q}\rceil, \ldots, \lceil(3/2)\hat{q}\rceil\}$. This is smaller than R, thus decreasing the number of possible clustering solutions to calculate. [58]

4.4 Confidence level of communities

Using the data from the merging of candidate communities, we can find a qualitative measure of the community structure. This is especially important for the nodes that lie in the borderland between communities. Perhaps it is equally likely that the node belongs to another neighboring community.

It is also important in structures similar to chains and tree in the network, as discussed in *Chapter 2.3.1*. These nodes are naturally quite sensitive to uncertain edges because they have few neighbors. Nodes having an uncertain community membership can be found using the candidate communities. If a node is found quite often in two different communities, the confidence that it has been classified correctly is low as it is very sensitive to the network structure.

4.4.1 Node-based Fusion of Communities

For the *node-based method*, the nodes are merged in a hierarchical manner, the two first nodes are the most similar and for each merge the nodes get more dissimilar. If a node was merged early into a community, it is less likely that it would belong to another community.

Therefore a qualitative measure of the certainty that a node i belongs to a community is found as $b_i = t_i^{-1}$ where t_i is the number of merges needed before the node i is added to the community. A larger value of this score indicates an early merge and therefore a more certain merge.

4.4.2 Community-based Fusion of Communities

For the *community-based algorithm*, the nodes are not merged in a hierarchical manner but a voting procedure is utilized after the k-means clustering.

Let $t_i = (t_{ij})$ indicate the number of communities in the meta-community j to which the node i belongs. Let $t_i^* = \max_j(t_{ij})$ be the maximum number of times the node has been found in some meta-community, j^*. The belonging score, b_i, of node i is found as the fraction of the frequency of that the node has belonged to meta-community, j^*, and the number of samplings, n_s. A correctly assigned node should be found in the same community with a large frequency, i.e. often in the same meta-cluster. The belonging score, b_i, is found by

$$b_i = \frac{t_i}{n_s}, \tag{4.7}$$

which is normalized with all the belonging score of all other nodes, as $\bar{b}_i = b_i \left[\max_i(b_i)\right]^{-1}$, for $i = 1, 2, \ldots, n$.

Chapter 5

Simulation experiments

In this chapter, we discuss methods used in the simulation experiments to evaluate and test the proposed methods. The primary aim is to determine how much certain information that is necessary in order to recover a satisfactory community structure compared with the external labels. This is accomplished by simulating edge existence probabilities and adding them to a certain network structure, thus creating an uncertain network. The certain networks can either be *real-world networks* or *synthetic networks*.

Recall that in a certain network all edges are known to exist with certainty, i.e. *the existence probability* $E_{ij} = 1$ for edges between nodes i and j in the network. In an *uncertain network*, the existence probability $E_{ij} \neq 1$ for at least some edges between nodes i and j. An *imperfect network* has in addition to uncertain edges, also missing or false edges in the network.

The generated uncertain and imperfect networks are used with the method outlined in the previous chapter. The obtained solution is compared with the external labels using validation methods. These labels indicate the correct community structure of the network. Note that some community detection methods are not able to obtain the same communities as specified by the labels even in the certain network.

This chapter contain discussions on three different topics. (i) the test networks used (both synthetic and real-world networks). (ii) the generation of uncertainty with missing and false edges. (iii) evaluation and comparison of the obtained solutions, based on discussions in *Chapters 2.1.3* and *2.3.2*. *Figure 5.1* contain a schematic outline of the proposed method and its connection with the material in this chapter.

5.1 Test networks

To compare methods and evaluate them for use in practical applications, both synthetic and real-world networks are used in the simulation experiments. Synthetic networks allow for careful adjustment of network param-

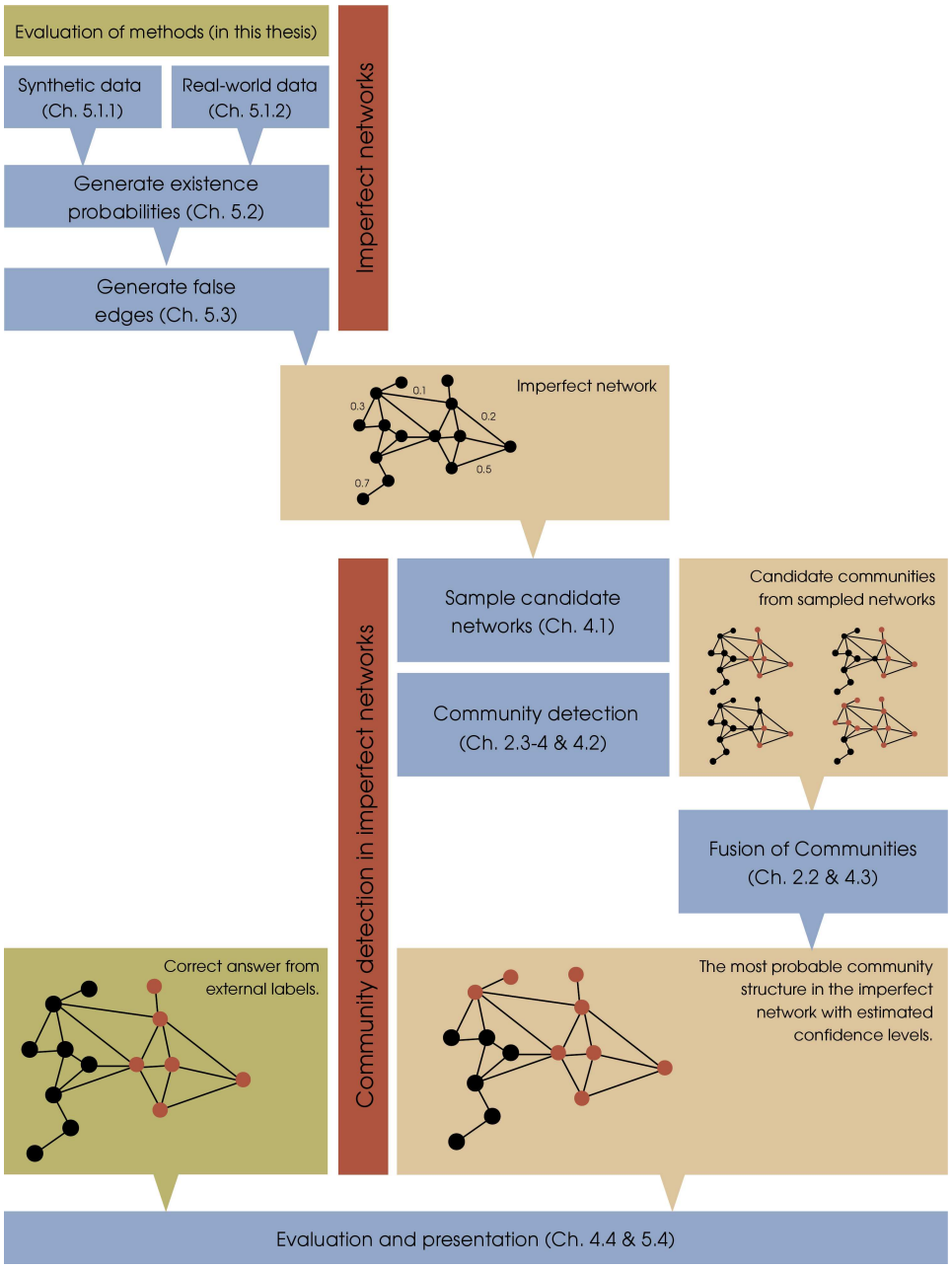

Figure 5.1: The method used in the simulation experiments to evaluate and compare the proposed methods for detecting communities in uncertain/imperfect networks.

eters. This enables the construction of networks with e.g. a certain degree distribution or a specific mixing parameter (see definition below). This is an advantage because it is difficult to find real-world networks with a certain number of nodes or degree distribution.

The drawback with using synthetic networks is that these networks are quite different from real-world networks, e.g. synthetic networks often have a heterogeneous degree distribution and often have less triangles than their real-world counter-parts. A mix of both network types is therefore required to enable careful evaluation of the proposed methods.

5.1.1 Synthetic networks

The *synthetic network* model used in this thesis is adopted from Refs. [59, 60]. The authors have constructed algorithms to generate artificial networks with community structures, which has become a standard benchmark for community detection using synthetic networks. The networks are generated using six different input parameters, shown in *Table 5.1*, together with the values used in this thesis. These parameters allow for the generation of families of networks with desired properties.

Variable	Value	Description
n	50	number of nodes in the network
$\bar{k}$	8	mean degree of each node
$k_{\max}$	16	maximum degree
μ	-	mixing parameter
$c_{\min}$	4	minimum size of a community
$c_{\max}$	20	maximum size of a community
β	1	exponent of community size distribution (typically $1 \leq \beta \leq 2$ in real-world networks)
γ	2	exponent of degree distribution (typically $2 \leq \gamma \leq 3$ in real-world networks)

Table 5.1: The parameters used for generating synthetic networks in the simulation studies using the algorithm from Ref. [59]. These parameters create networks similar to Newman-Girvan benchmarks. The mixing parameter, μ, varies for each synthetic network, see *Table 5.2* for details. [39]

The mixing parameter, μ, is the fraction of edges between the different communities and $1 - \mu$ is the fraction of intra-community edges. A small mixing parameter corresponds to well-separated communities, the extreme is when $\mu = 0$ and only disjoint communities exist. As μ increases, the communities become more difficult to detect, until $\mu = 0.5$ when no communities exist in the network according to the adopted definition of a community in *Definition 3*. The algorithm to generate the synthetic networks consists of five different steps. A simplified version, see Ref. [59] for the full version, is presented in *Algorithm 7*.

Algorithm 7 Generating synthetic networks with community structure

Firstly, generate the degree of each node by sampling from a power-law distribution with parameter, γ, satisfying, $\bar{k}$ and $k_{\max}$. Secondly, generate the size of each community from a power-law distribution with parameter, β, such that all nodes are members of a community and the community sizes are consistent with the parameters, $c_{\min}$ and $c_{\max}$.

(i) using the configuration model, assign edges between all nodes such that the degree of all nodes are satisfied,

(ii) randomly distribute the nodes to the communities in the network,

(iii) rewire the edges between nodes until the mixing parameter, μ is satisfied.

The drawback of this algorithm is the lack of triangles observed in real-world social networks, which result in a sparser network than in empirically found networks. The advantage is that synthetic networks enable the study of how the mixing parameter is correlated with the effectiveness in finding communities in uncertain networks. The algorithm has a linear complexity, $\mathcal{O}(n)$, and can therefore be used to simulate large networks with community structures that are consistent with real-world social networks. [59]

5.1.2 Real-world networks

The three real-world networks used are classics in network science and describe different types of networks: human friendships, football matches and animal association networks. These networks all have a known community structure which is supplied by external labels. The *karate* network describes friendships between members of a karate club at a U.S. university in 1977. The club fractured into two parts during the study and the resulting two groups are the labels used for the external evaluation. The community structure (the factions) is assumed possible to be recovered using a good community detection algorithm. [61]

The *football* network contains all the Division IA college football teams and the edges indicate games during the fall of 2000. The labels are the conferences to which each team belongs and matches are most often played between teams from the same conference. Therefore communities detected in this network should indicate the different conferences. [39]

The last real-world network is the *dolphin network* which describes a number of dolphins with frequent association. The labels are assigned using observations of dolphin behavior, as members of a school signal the start and finish of some travel by using *side flopping* and *upside-down lobtailing*. Because only dolphins that are members of the school does this signaling, it is possible to recover the real communities in the network. [62]

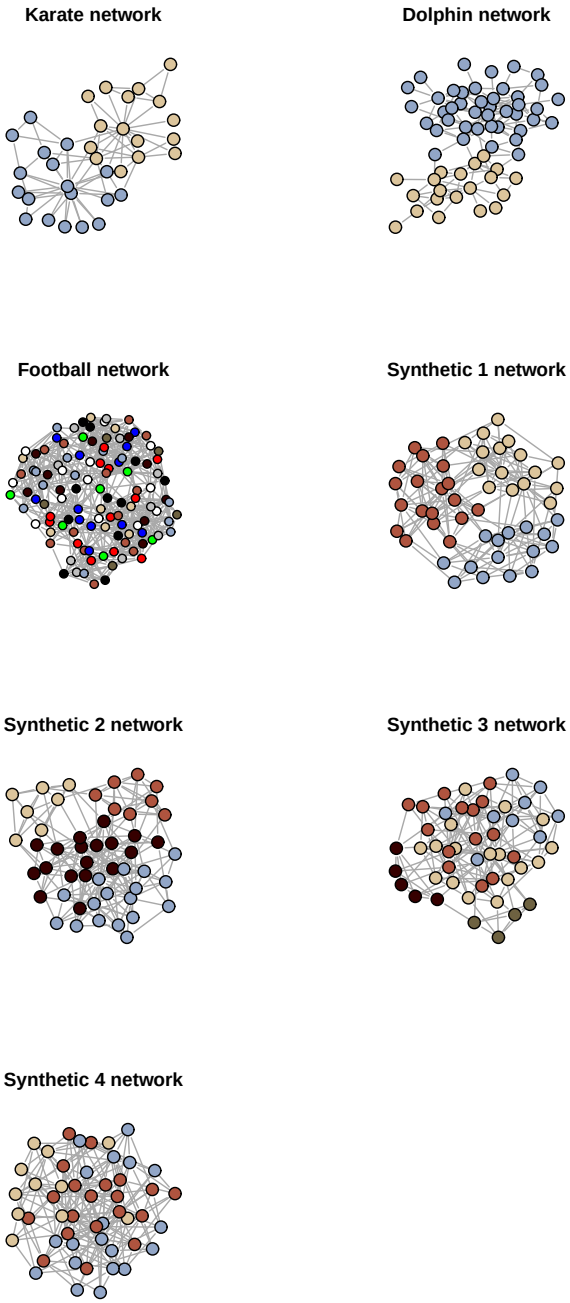

Figure 5.2: The networks shown in *Table 5.2* with the community structure as given by external labels. The four synthetic networks are quite similar in appearance but with varying diffuseness of the community structure.

5.1.3 Descriptive network statistics

We test the methods on seven networks with different characteristics in hope to find some general rule-of-thumb when to use which method. These networks are presented by their statistical properties in *Table 5.2* and graphically (with their community structure given by external labels) in *Figure 5.2*.

Name	μ	n	m	C	Ref.
Karate	0.14	34	78	0.256	[61].
Dolphin	0.04	62	159	0.309	[62].
Football	0.36	115	613	0.407	[39].
Synthetic 1	0.20	50	193	0.266	[59].
Synthetic 2	0.30	50	200	0.280	[59].
Synthetic 3	0.40	50	180	0.224	[59].
Synthetic 4	0.50	50	207	0.173	[59].

Table 5.2: Benchmark networks to test the performance of the proposed methods with n nodes, m edges, transitivity C, and mixing parameter μ. The transitivity is calculated as fraction of the number of closed paths of length two (triangles) and the number of paths of length two. Thus representing the number of paths with length two that are triangles, this measure is often high for social networks. The mixing parameter is the fraction of inter-community edges and the total number of edges.

5.2 Generating uncertain networks

To generate uncertain networks, probabilities are simulated and assigned to each edge in the synthetic and real-world networks. The probabilities are simulated from two different distributions[1]: a discrete *M-distribution*, $\mathcal{M}(x, p_c)$, and the semi-discrete *L-distribution* $\mathcal{L}(x, p_t)$. The distributions are defined by the following probability distribution functions

$$f_{\mathcal{M}}(x) = \begin{cases} \frac{p_c}{2}, & x = 0 \\ 1 - p_c, & x = \frac{1}{2} \\ \frac{p_c}{2}, & x = 1 \end{cases} \qquad f_{\mathcal{L}}(x) = \begin{cases} 1 - p_t, & x \in [0, 1) \\ p_t, & x = 1 \\ 0, & x \notin [0, 1] \end{cases} . \tag{5.1}$$

The generated probabilities are arranged in an *edge probability matrix*, $\mathbf{E} = [E_{ij}]$, where $E_{ij} \sim \mathcal{M}(x, p_c)$ or $E_{ij} \sim \mathcal{L}(x, p_t)$, for $i, j = 1, 2, \ldots, n$.

The two distributions, shown in *Figure 5.3*, help us study two different aspects of the efficiency in the proposed methods: (i) the case of missing

[1]Named after its rotated M-like shape when $p_c < 1/2$ (actually the distribution is W-shaped but named M-shaped to distinguish it from the Weibull-distribution) and the rotated L-formed density function on the interval $[a, b]$, where the total probability mass in the tail is denoted P_t.

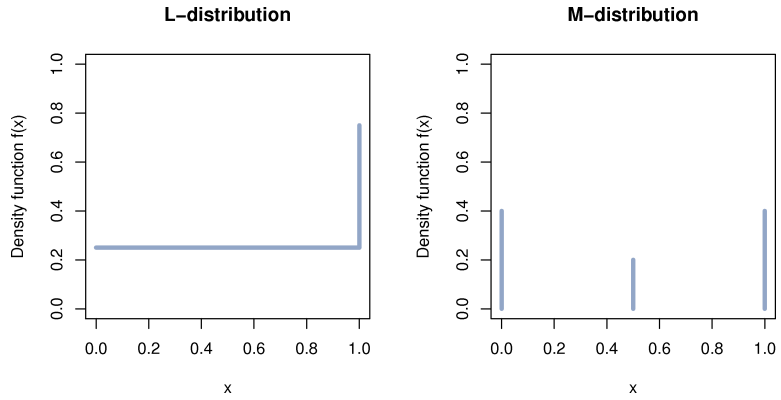

Figure 5.3: The density functions of the L-distribution with $p_t = 0.75$ and M-distribution using $p_c = 0.8$.

links, and (ii) the case of the edge existence probability distribution. The optimal algorithm should be quite insensitive to changes in p_c, i.e. the modularity and diagnostic values (see below) should be independent of p_c. An optimal algorithm should also detect a good community structure at low values of p_t, i.e. networks with many uncertain edges.

The M-distribution is used to observe how well the communities are detected when edges are missing. When $p_c = 0$, all edges are included in the network but they are uncertain with the probability, $E_{ij} = 1/2$. Increasing the probability of certainty, p_c, increase the number of randomly removed edges with the remaining edges treated as certain.

The L-distribution does not remove any edges from the network but instead varies the number of edges that are certain versus uncertain. When $p_t = 0$, all edges are uncertain with some uniformly distributed existence probability and increasing p_t will generate more certain edges with $E_{ij} = 1$. The extreme is found at the point at $p_t = 1$ where the original network is recovered.

5.3 Generating imperfect networks

To investigate how the algorithms work on imperfect networks which also have false edges (that do not exist in the *real* network), a noise term is added into the adjacency matrix. A symmetric *noise matrix*, $\mathbf{F} = [F_{ij}]$, is generated with each element as a Bernoulli randomly distributed variable $F_{ij} \sim \mathcal{B}(p_n)$ with $F_{ij} = F_{ji}$ and p_n the *noise factor* i.e. the fraction of edges added.

The noise matrix is added to the adjacency matrix element-wise, $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{F}$, adding elements which only add new edges. All non-zero elements are treated equally as the network is not weighted, i.e. if the element is non-

zero indicate the existence of an edge. The network is sampled as in the case for uncertain networks, described in *Chapter 4.1*, using $\tilde{\mathbf{A}}$ instead of $\mathbf{A}$. The M-distribution is used to generate the edge existence probabilities as this randomly remove edges, i.e. introduce missing edges into the uncertain network with false edges.

5.4 Evaluating community structures

The last and most important step is to evaluate the obtained solution, by calculating a measure of the similarity between the solution found and the optimal (correct) solution. This makes it possible to qualitatively rank the different combinations of methods for different types of networks. Using this approach with external information (e.g. labels) is often called *supervised evaluation* and is used in this thesis to evaluate how the methods perform with respect to the correct solution.

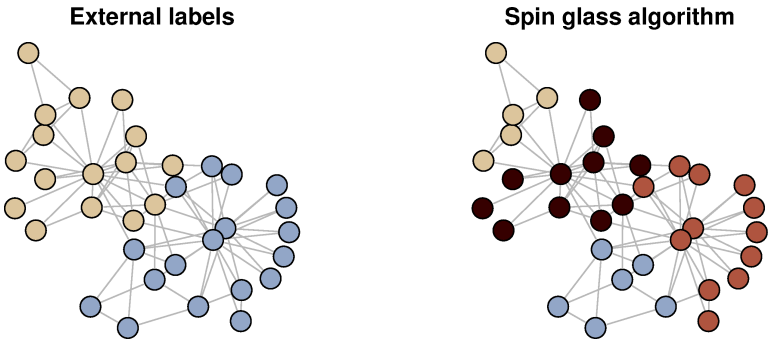

Figure 5.4: The community structure of the karate network [61] using external information (found in the study) and the community structure detected by the spin glass method [16].

An important remark is that most community detection algorithms do not find the same solutions as indicated by labels, as demonstrated in *Figure 5.4*. Therefore, we apply another method called *unsupervised evaluation* to compare the solution found on the imperfect network with the solution found by the same community detection method applied on the certain network. That is, we assume that the solution found by the community detection method is the community structure in that specific network. Recall the discussion in *Chapter 2.3.2*, that it is difficult to validate community structures and different methods generate different solutions.

5.4.1 Unsupervised evaluation

The first unsupervised method is comparison of modularity values with the same method used on the certain version of the network. This measure has

the main drawback that modularity is ill-defined and it is not obvious what the difference in modularity scales to. The problems include answers for questions like: what is a large and small difference in modularity, and what does this mean in terms of number of misplaced nodes.

Instead, we use two statistically based methods: the *Mean Square Error* (MSE) and the *correlation* to compare two solutions. The MSE is often used to compare estimated values to the true value and is also similar to the measures of separation and cohesion, defined in *Chapter 2.1.3*.

Let the estimated community structure be represented by a neighbor matrix, $\hat{\mathbf{N}} = [\hat{N}_{ij}]$, where $\hat{N}_{ij} = 1$ if nodes i and j are found in the same community and 0 otherwise. Construct the corresponding matrix for the assumed correct solution (from the community detection on the certain networks), $\mathbf{N} = [N_{ij}]$. The matrix version of $\mathsf{MSE}(N, \hat{N})$ is the mean of the square root of MSE for each row

$$\mathsf{MSE}(N, \hat{N}) = \frac{1}{n} \sum_{i=1}^{n} \sqrt{\frac{1}{n} \sum_{j=1}^{n} \left(\hat{N}_{ij} - N_{ij} \right)^2}. \tag{5.2}$$

Note that the mean should be taken over $n - 1$ degrees of freedom but when n is large the difference is minor but the resulting estimator is biased.

The matrix MSE estimates the differences of the matrices, thereby the difference between the ideal (correct) and the obtained community structure. The correlation is used to calculate a slightly different value, which indicate how the rows in the matrices tend to be similar [20]. The mean correlation $\bar{\rho}(N, \hat{N})$, between the two matrices, N and $\hat{N}$, is found as the mean of the *Pearson correlations*, ρ_i, for each row

$$\bar{\rho}(N, \hat{N}) = \frac{1}{n} \sum_{i=1}^{n} \rho_i(N_i, \hat{N}_i) = \frac{1}{n} \sum_{i=1}^{n} \frac{\mathsf{Cov}(N_i, \hat{N}_i)}{\mathbb{V}[N_i]\mathbb{V}[\hat{N}_i]}, \tag{5.3}$$

where $\mathbb{V}(\cdot)$ denotes the variance. Using the covariance between each element in each row and the expected value (mean) of the that specific row

$$\mathsf{Cov}(N_i, \hat{N}_i) = \frac{1}{n} \sum_{j=1}^{n} (N_{ij} - \mathbb{E}[N_i])(\hat{N}_{ij} - \mathbb{E}[\hat{N}_i]), \tag{5.4}$$

where $\mathbb{E}(\cdot)$ denotes the expected value.

5.4.2 Supervised evaluation

Another way to compare clustering results is supervised evaluation, in which some external correct solution is used. One method is based on the MSE and correlation method introduced for unsupervised validation. The difference is that the labels are used to create the *ideal neighborhood matrix*, $\mathbf{N}^* = [N^*_{ij}]$,

with $N^*_{ij} = 1$ when $l_i = l_j$ but zero otherwise and $N^*_{ii} = 0$, thus creating an powerful supervised method.

Another method is to use the *Normalized Mutual Information* (NMI) measure from information theory. The NMI-measure, $\hat{\mathbb{I}}(\mathcal{C}, \mathcal{L})$, can be interpreted as how much we know about the external labeling, $\mathcal{L}$, given the obtained solution, $\mathcal{C}$, and vice versa.

Assume that $\mathcal{L} = \{l_i\}$ where l_i is the label of node i and the same for $\mathcal{C}$ with the obtained community membership of node i. Further assume that l and c are the realizations of some random variables, L and C, with some (joint) probability distributions as

$$\mathbb{P}(l, c) = \mathbb{P}(L = l, C = c) = \frac{|\mathcal{L} \cap \mathcal{C}|}{n}, \tag{5.5}$$

$$\mathbb{P}(l) = \mathbb{P}(L = l) = \frac{|\{l_i \in \mathcal{L} : l_i = l\}|}{n}, \tag{5.6}$$

$$\mathbb{P}(c) = \mathbb{P}(C = c) = \frac{|\{c_i \in \mathcal{C} : c_i = c\}|}{n}, \tag{5.7}$$

where $|\{l_i \in \mathcal{L} : l_i = l\}|$ is the number of elements in $\mathcal{L}$ which equals the label l with the corresponding for c, and n is the total number of nodes, $n = |\mathcal{L}| = |\mathcal{C}|$. The *mutual information*, $\mathbb{I}(C, L)$, is defined by

$$\mathbb{I}(C, L) = \sum_l \sum_c \mathbb{P}(c, l) \log \left(\frac{\mathbb{P}(c, l)}{\mathbb{P}(c)\mathbb{P}(l)} \right), \tag{5.8}$$

where the sums are taken over all assumed values of l and c, and $\log(\cdot)$ is the logarithm (with base 2). To normalize the mutual information, we need to introduce the *entropy*, $\mathbb{H}(X)$, of a random variable X defined as

$$\mathbb{H}(X) = -\sum_x \mathbb{P}(x) \log \left(\mathbb{P}(x) \right). \tag{5.9}$$

The expression for mutual information in (5.8) can be rewritten using the expression for the entropy, $\mathbb{H}(X)$, in (5.9) above as

$$\mathbb{I}(C, L) = \mathbb{H}(\mathcal{C}) - \mathbb{H}(\mathcal{C}|\mathcal{L}). \tag{5.10}$$

This can be used to show the relationship between entropy and mutual information in a schematic form as in *Figure 5.5*.

The *normalized mutual information*, $\hat{\mathbb{I}}(\mathcal{C}, \mathcal{L})$, between the obtained community structure, $\mathcal{C}$, and the externally given labels, $\mathcal{L}$, is

$$\hat{\mathbb{I}}(\mathcal{C}, \mathcal{L}) = 2\frac{\mathbb{I}(\mathcal{C}, \mathcal{L})}{\mathbb{H}(\mathcal{C}) + \mathbb{H}(\mathcal{L})}, \tag{5.11}$$

which equals zero if the community structures are independent and unity if they are equivalent. This measure accounts for the possibility that a

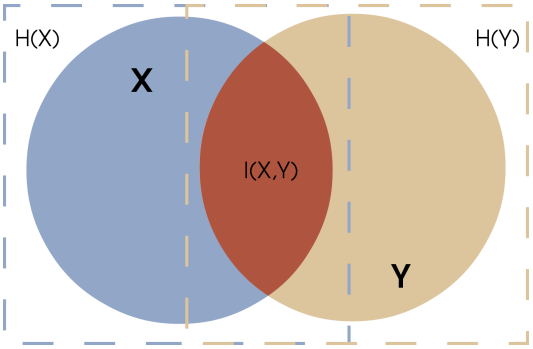

Figure 5.5: Schematic representation of the mutual information, $\mathbb{I}(X, Y)$, between two random variables X and Y, entropy of the variable X, $\mathbb{H}(X)$, and, entropy of the variable Y, $\mathbb{H}(Y)$. Adapted from Ref. [63].

combination of communities in e.g. the obtained solution may correspond to one community in the information obtained by external labels.

This measure is superior to conventional methods like recall and precision and has therefore become the general method for benchmarking community detection algorithms. In this thesis, we therefore adopt this approach for most evaluation, despite the existence of other simpler evaluation measures. [64, 63]

Chapter 6

Results and discussion

In this chapter, we use simulation experiments to evaluate the methods introduced in *Chapter 4*. This is performed using the methods discussed in *Chapter 5* for generation of imperfect networks and evaluation of merged community structures.

The first part of this chapter is concerned with demonstrating how to improve accuracy by merging multiple runs of the same community detection algorithm. This is a cornerstone in the analysis of the imperfect networks, as we need to merge the detected candidate communities. It is therefore fruitful to analyze merging methods isolated from the statistical errors introduced by sampling and by scrambling network data.

The merging method itself is also useful for other applications than analyzing uncertain networks. Merging communities found by different community detection methods in the same network could result in a better estimated community structure. As previously discussed, many community detection methods and the modularity measure itself depend on parameters which determine the size of communities found. Therefore, merging several runs of a certain method using different parameters can be used to analyze overlapping communities, robustness of the obtained structure and other similar properties.

The second part of this chapter is concerned with testing the methods to detect community structures in uncertain/imperfect networks. We begin this part by a short recapitulation of the dependencies between the introduced methods for sampling, community detection[1] and merging[2]. The main interest is demonstrating the possibility of detecting communities in net-

[1] In this chapter, we adopt the following abbreviations for the different community detection methods used to detect the candidate communities: (EB) Divisive algorithm based on betweenness, (GA) Greedy agglomerative method, (SM) Spectral method, (SP) Spin glass algorithm, (WT) Random walk on networks, (LP) Label propagation.

[2] To merge the candidate communities, we apply the three proposed methods in this thesis: *Two-step Fusion of Communities* (TFC), *Node-based Fusion of Communities* (NFC) and *Community-based Fusion of Communities* (CFC).

works estimated using incomplete information. An additional aim is finding some guidelines regarding when to use a specific combination of community detection and merging methods.

To evaluate the efficiency of the proposed methods, we use supervised evaluation with Normalized Mutual Information (NMI) to compare the obtained solution with the correct community structure specified by labels in the data material. Two additional measure are also used: the matrix mean square error (MSE) and the correlation between the neighborhood matrix constructed from the obtained solution and the external label information. These methods are all discussed in *Chapter 5*.

In the last part of this chapter, we offer a comparison between supervised and unsupervised evaluation, as we often are forced to use the unsupervised form when no external labels are available. We also show how it is possible to obtain confidence levels for that a node belong to a specific community.

All graphs and statistical analysis are carried out using non-parametric statistical methods. These include *non-parametric regression analysis* in which we use the R-package `np` [56, 65] with the second-order Gaussian kernel. Hypothesis testing and confidence bands are estimated using *non-parametric bootstrap methods*. As the simulations are quite computationally intensive, these methods offer a possibility to do statistical analysis when the assumptions underlying the central limit theorem are not fulfilled. The non-parametric regression method also allows for detecting non-linear relationships in the data.

Excellent introductions to these methods are given in Ref. [66], with more advanced presentations of non-parametric regression in Refs. [67, 68] and bootstrapping in Ref. [69].

6.1 Improving community detection by merging

One possible application for the proposed merging methods is to combine the results of several different community detection methods or different results from the same method. As previously discussed, we often have to choose between fast or accurate methods and different methods favor different sizes of detected communities. By combining these solutions, we can hopefully take advantage of the differences in the methods to find a better solution. We have also discussed the problem with the resolution limit and how different parameters in each community detection method influence it. By varying these parameters and merging the results, we can detect community structures of different sizes to find communities within communities.

In this section, we present results from merging a number of runs by the LP algorithm to demonstrate how the combination can be used to improve performance. As discussed in *Chapter 2.3*, the LP algorithm is stochastic and generates different results for each run on the same network. The result-

ing community structure from each run is regarded as a candidate network and is merged using the three methods discussed in *Chapter 5*. The LP algorithm is used to detect communities in the weighted constructed graph in the TFC method.

We have applied the LP-algorithm to the three real-world networks found in *Table 5.2*. In *Figure 6.1*, the normalized mutual information and correlation (supervised) is shown for an increasing number of combined results from the LP algorithm. Note that, these networks are certain and no weight (edge existence probability) has been added onto the structure. We have merely used the mean of 30 runs to eliminate any stochastic effects of the combination step, especially in the TFC method using the LP algorithm.

The results in *Figure 6.1* are summarized in *Table 6.1* with some diagnostics of the obtained regression model. The regression is performed using a non-parametric method with the second order Gaussian kernel and Bootstrap methods to estimate the confidence bands and the p-value of the regression significance test. The comparison is performed by examining the estimated confidence band from the merged LP runs with the reference NMI value from the SP algorithm. If the confidence band does not include the reference value, found by the SP algorithm, we conclude that there is a significant difference between the two methods.

We note that there are significant improvements for two of the networks: karate and dolphin. The merged LP runs does not perform better on denser networks as the football networks. As the karate and dolphin networks share some characteristics (see below) it is expected that similar results should be obtained on these networks.

Network	Method	bw	p-value	Compared with SP
karate	NFC	2.57	0.01	Better
	CFC	0.93	< 0.01	Better (with $n > 8$)
	TFC	2.77	< 0.01	Better (with $n > 8$)
dolphin	NFC	4.76	< 0.01	Better
	CFC	0.46	< 0.01	Inconclusive
	TFC	2.83	< 0.01	Better
football	NFC	5.00	0.09	Worse
	CFC	0.31	< 0.01	Worse
	TFC	0.94	< 0.01	Worse

Table 6.1: The bandwidth, *bw*, p-value of the hypothesis test of significant regression, and comparison with the Spin glass algorithm for the non-parametric regression of the NMI values shown in *Figure 6.1*. The hypothesis test is performed by bootstrap methods from Ref. [70]. The comparison is made by examination of the bootstrapped 95% confidence intervals for the estimated NMI function and the reference solution found by the SP algorithm, see text for details.

We can also conclude that the CFC method yields less varying results

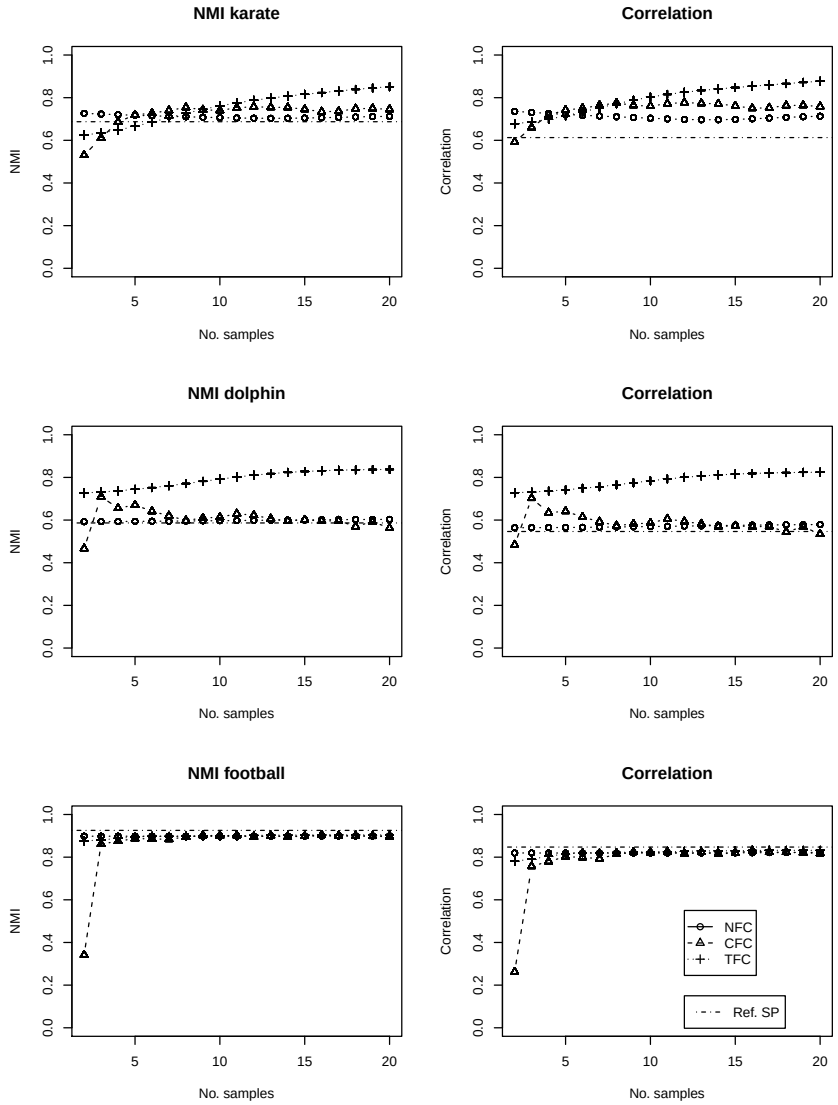

Figure 6.1: The normalized mutual information and correlation of merged results from the LP algorithm using TFC, NFC and CFC. The values are the mean of 30 runs in the karate, dolphin, and, football networks. The horizontal line indicates the NMI and correlation of the community structure found by the Spin glass algorithm. The curves are estimated using non-parametric regression with the Gaussian kernel.

as the bandwidth is smaller for this method, larger bandwidths indicate larger smoothing (averaging) of the underlying data material. This is as expected from the discussion in *Chapter 4* that CFC uses a low-dimensional approximation that effectively remove noise.

The results from the three different networks indicate that the TFC method is best in general. However the TFC method is stochastic in this experiment as it uses a second application of the LP algorithm on the weighted network constructed from the frequency matrix. Individual runs of LP-TFC perform worse or better as the indicated mean in *Figure 6.1*, which is calculated from 30 runs of the merging method. In practical applications, we should therefore refrain from using TFC as a deterministic outcome is preferred.

NFC is a good choice that yields results comparable with the Spin glass algorithm and this at a lower complexity[3], $\mathcal{O}(n^2)$. This complexity is lower than the complexity of the Spin glass algorithm, which is $\mathcal{O}(n^{3.2})$ for sparse networks (see *Chapter 2.4.4*).

We conclude that it is possible to combine several results from the LP algorithm to find a good estimate of the community structure in the three real-world networks observed. The performance is equivalent to the Spin glass algorithm but at a significant lower complexity. We also note that this method could be used to merge[4] the results from different community detection methods.

6.2 Simple method for uncertain networks

A simple possible solution to the problem with uncertain edges is using a community detection method that supports weighted graphs[5]. The edge existence probability, E_{ij}, is used as the weight for each edge and therefore mimics the TFC and NFC method as it constructs a similar graph using an estimated frequency matrix.

In this simpler case, we instead use the probabilities directly with the interpretation that the probabilities indicate how probable it is that two nodes are found in the same community. This is not entirely correct as the probability in the uncertain graph in conjunction with the network structure determines the communities. The probabilities given in the edge existence

[3]The complexity of the LP algorithm is $\mathcal{O}(m)$ and the hierarchical clustering in the NFC method has complexity $\mathcal{O}(n^2)$. Combining k results from the LP algorithm yields a complexity of $\mathcal{O}(km+n^2)$, which is simplified by knowing $k < n$ and $m < \frac{1}{2}n(n-1) < n^2$ to $\mathcal{O}(n^2)$.

[4]A weighted version of this technique is possible, thus giving more importance to some methods than others. One possibility is using the normalized modularity as a weight, to give higher importance to community structures with higher modularity. As higher modularity indicate a better community structure, this should be an natural extension.

[5]Four of the six methods presented in this thesis support weighted networks: SP, GA, WT, and LP.

probability matrix only describe how probable it is that two nodes are linked together, not that they are clustered together, compare the discussion in *Chapter 4.3.1*. We begin by introducing a simulation study of this solution and then discuss its implications and why the method is unsound.

To study the accuracy of this approximation, we apply the method on the three different real-world networks presented in *Table 5.2*. In *Figure 6.2*, we present the detailed results regarding four different algorithms as a mean of 50 different sets of probabilities. The results presented indicate that this method is capable to recover some of the community structure in the uncertain network. Denser networks with more edges and more clear structure (as the football network) are more robust to uncertainty. However useful results are also found when applying this method on more sparse networks (karate and dolphin networks). The solutions found by the LP algorithm in the dolphin and karate networks have the highest NMI and correlation. In the football network, the SP algorithm finds community structures with the highest values.

We note that *none* of the four algorithms managed to find the correct network structure. This was expected as the probabilities used in conjunction with the weighted community detection are not the same as estimated by the frequency matrix. Community structures are determined by more information than the edge existence probabilities. There are probably higher-order structures within the network (triangles, paths, etc.) which create a robustness for uncertainty. These structures are neglected in this approach and would not work in practical applications. This was previously discussed in *Chapters 3* and *4.3.1*.

The method only works in this case because the edge existence probabilities have all been generated from the same distribution. Therefore the expected edge existence probability, $\mathbb{E}[E_{ij}] = \mathbb{E}[E]$, i.e. is the same for all edges. In practical applications, this would not hold and therefore this method cannot find a good estimated community structure.

Finally, the approximation of using the edge probability matrix is limiting our efforts in analyzing imperfect networks, as this method cannot analyze uncertain nodes and randomly added false edges. Although a fast method, it is not straight-forward to show that the approximation is valid and when it does perform well.

6.3 General method for uncertain networks

In this section, we are interested in the performance of NFC and CFC on different kinds of networks when using existence probabilities generated by the L-distribution. This tells us how the amount of certain versus uncertain edges are related to the performance and accuracy of the proposed methods. By using this distribution, we can investigate how the performance depends

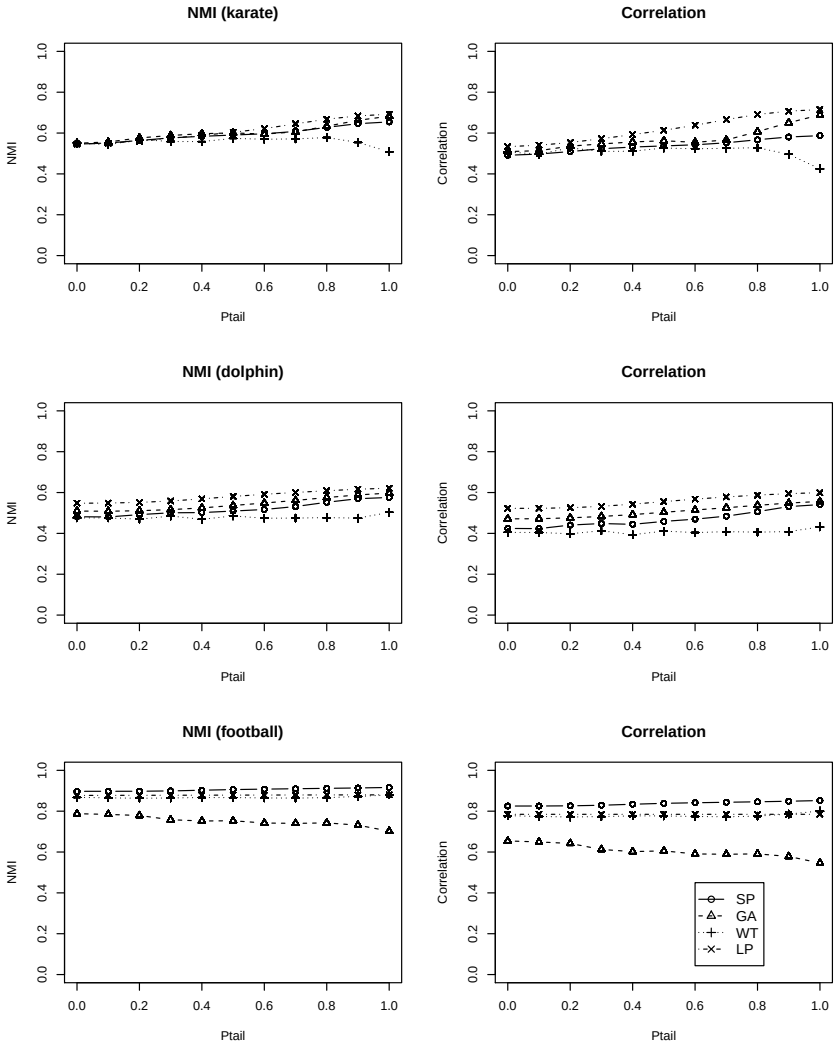

Figure 6.2: The normalized mutual information and correlation from using the SP, GA, WT, and LP algorithms using the existence edge probabilities as weights. The values are the mean of 50 uncertain networks generated using the L-distribution to generate uncertain edges in real-world networks. The curves are estimated using non-parametric regression with the Gaussian kernel.

on the mixing parameter, μ, in synthetic networks and the number of certain versus uncertain edges, p_t, in the simulated edge existence probabilities. Each network is sampled $n_s = 50$ times, generating as many candidate networks and the results are based on the mean of 30 simulation runs.

We begin our investigation by using synthetic network structures with different mixing parameters, ranging from 0.20 in Synthetic 1 (distinct community structure) to 0.50 (no community structure) in Synthetic 4. In *Figures 6.3* and *6.4*, we show the NMI and (supervised) correlation using CFC and NFC to merge the candidate communities found in the synthetic networks.

We begin by concluding that the supervised correlation and NMI in many cases yield equivalent results. This indicates that the simpler correlation measure can be used instead of the more complex NMI, resulting in the same conclusions. Continuing, we note that using NFC to merge candidate networks perform as good as or better than using CFC. This is a good result as NFC demands less computational effort than CFC, the complexity is $\mathcal{O}(n^2)$ (the complexity of a typical agglomerative hierarchical clustering method) for the first and at least $\mathcal{O}(n^3)$ for the latter (as the eigenvector calculation is quite complex).

This result holds for both NMI and correlation and for all the synthetic networks used. Further, we note that the LP algorithm often yields the best results when used with the NFC method and the EB algorithm is the best choice using the CFC method. As the EB algorithm has a higher complexity than the LP algorithm, $\mathcal{O}(n^3)$ versus $\mathcal{O}(m)$, we conclude that LP-NFC is the better choice on synthetic networks[6].

We also note that the performance of the community detection methods decreases with increasing mixing parameter, μ. This effect has been observed before in many papers benchmarking different community detection methods on generated synthetic networks with varying mixing parameters. In Ref. [64], this effect is documented in a wide range of different community algorithms, the authors also use the same method to generate synthetic networks as in this paper.

As the mixing parameter tends to 0.5, the community structure becomes more vague and when $\mu = 0.5$, there are as many edges between communities as within them. This is approximately the same as saying that the network does not have a community structure, however in Ref. [64] the authors claim to find community structures in networks with $\mu = 0.75$, but this violates the definition of a community used in this thesis.

This is also the explanation for the poor performance of the LP algorithm in Synthetic 4. As p_t increases the NMI decreases for the LP-NFC method. This does not change the fact that LP-NFC is the most preferable method.

[6] As the number of edges m is typically larger than the number of nodes n, but n^2 is often larger than m for sparse networks.

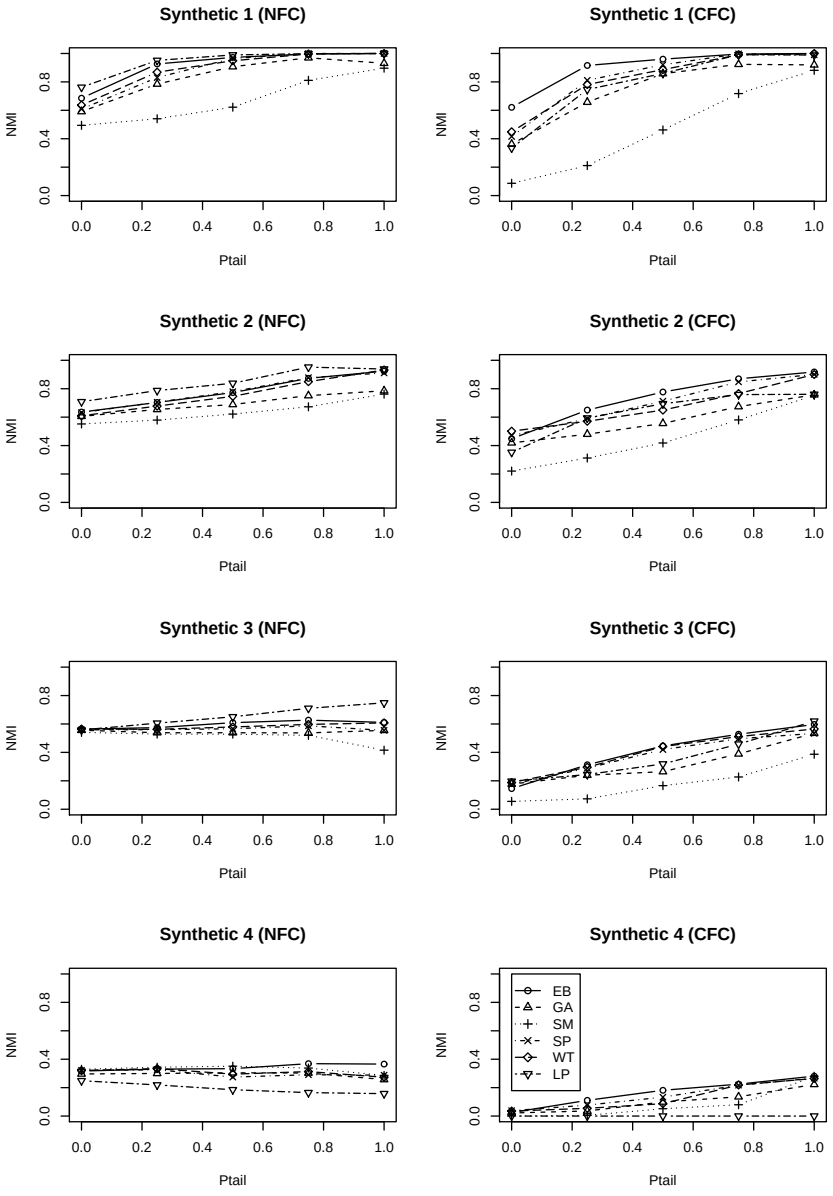

Figure 6.3: The normalized mutual information for the six different community detection methods used in combination with NFC or CFC on four different synthetic networks with the L-distribution. The curves are estimated using non-parametric regression with the Gaussian kernel on the result of 30 runs each using 50 candidate networks.

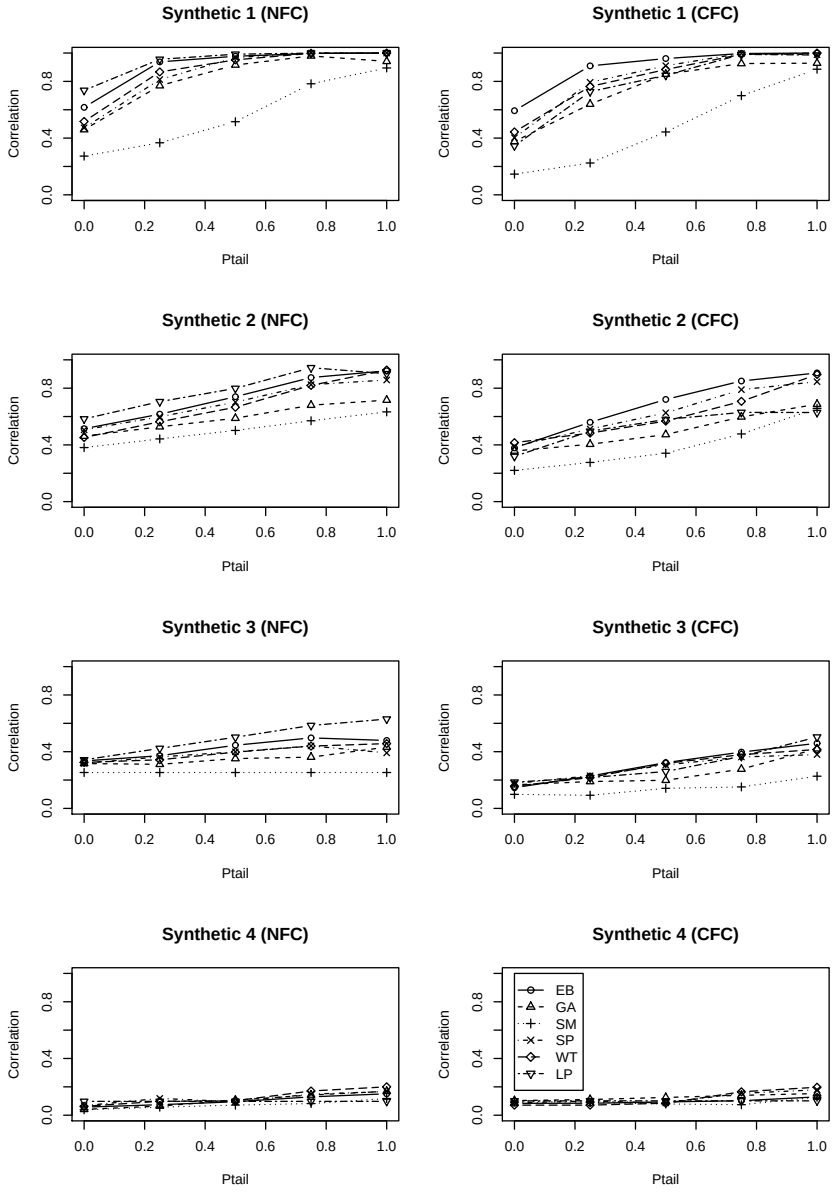

Figure 6.4: The correlation for the six different community detection methods used in combination with NFC or CFC on four different synthetic networks with the L-distribution. The curves are estimated using non-parametric regression with the Gaussian kernel on the result of 30 runs each using 50 candidate networks.

As noted above, Synthetic 4 does not contain a community structure and it is therefore desirable that the NMI score indicates this as well.

The shape of NMI and correlation curves are explained by that increasing p_t correspond to an increasing number of certain edges. As we approach $p_t = 1$, we recover the results found by applying the community detection methods on the certain network. This is as expected and validates the accuracy of the merging and sampling methods. The number of samplings, n_s, has a positive effect on the NMI and correlation values at all values of p_t. An increased number of samplings result in that the maximal NMI and correlation is found at lower values of p_t. With this we mean that the curve attains its maximum plateau at a lower value of p_t. To some degree, we can therefore compensate the uncertainty in the network with more candidate networks. Note that this works only to some extent and we can never compensate for information loss by sampling more candidate networks.

We now continue with some real-world networks: the karate, football, and dolphin networks, in which we have added some edge existence probabilities. In *Figures 6.5* and *6.6*, we show the NMI and (supervised) correlation estimated by 30 simulation runs using 50 samplings in each run.

In these three networks, we observe the same trend as in the generated synthetic network structures. The merging method NFC perform as good as or better than CFC when using most community detection algorithms. When using NFC to merge candidate communities, LP is the best choice when observing both NMI and correlation. Using CFC, we get the best results using the SP algorithm (for karate and football networks) and the GA algorithm (for the dolphin network). As previously discussed SP and GA have a higher complexity than the LP algorithm. Therefore, we conclude that LP-NFC is the best choice in general with a low complexity as well as the highest accuracy for the methods tested.

6.3.1 Why is LP-NFC the best method?

The reason for why LP algorithm performs well could be the stochastic nature of the method which perturbs the community structure in each run. It is probable that this creates more varied candidate communities, as each perturbation induces small changes in the found communities. This leads to better exploration of the space of possible communities structures. When used in combination with NFC the small information loss counteracts the perturbations, thus resulting in better modularity and quality of the detected communities. This is similar to adding noise in e.g. bootstrap methods to generate smoother estimates of probability density functions (see Ref. [69]). Thus the perturbations can be seen as noise which results in *smoother* detected community structures.

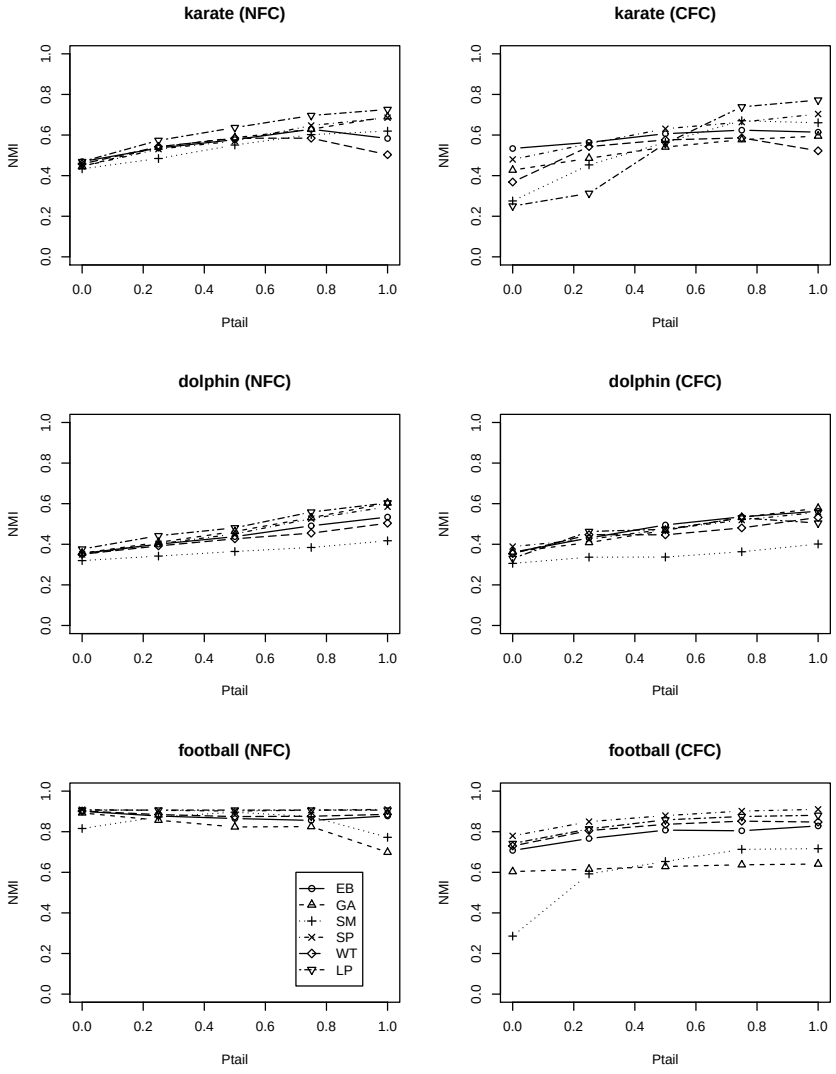

Figure 6.5: The normalized mutual information for the six different community detection methods used in combination with NFC or CFC on three different real-world networks with the L-distribution. The curves are estimated using non-parametric regression with the Gaussian kernel on the result of 30 runs each using 50 candidate networks.

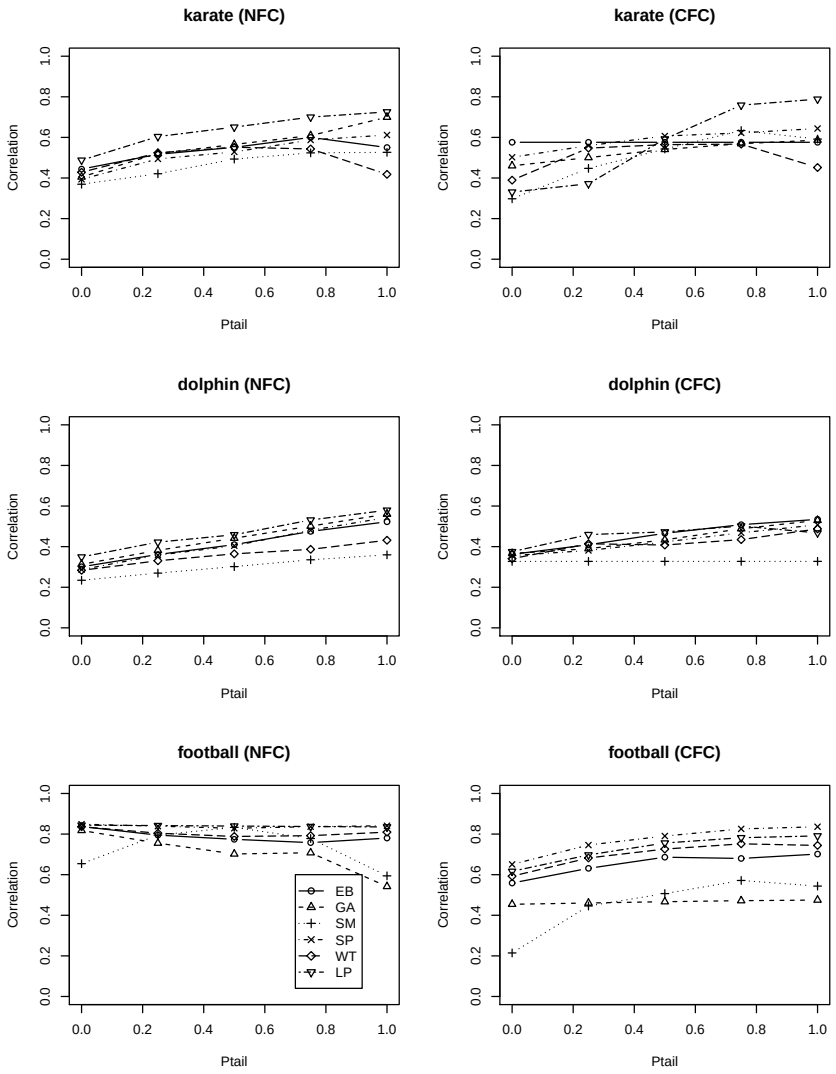

Figure 6.6: The correlation for the six different community detection methods used in combination with NFC or CFC on three different real-world networks with the L-distribution. The curves are estimated using non-parametric regression with the Gaussian kernel on the result of 30 runs each using 50 candidate networks.

6.3.2 Comparing real-world and synthetic networks

Finally, we compare the synthetic networks with the real-world networks to find similarities in the NMI and correlation curves. Our hypothesis is that there is a connection between some unknown network measure and the resulting shape of the NMI and correlation functions. Firstly, we note that the karate and dolphin networks have similarly shaped NMI and correlation functions. The community detection methods for uncertain networks applied on the karate network often find community structures with higher accuracy than the same method find in the dolphin networks. The accuracy functions for two real-world networks have quite similar appearance as for the network Synthetic 3.

Secondly, the corresponding functions for the football network are quite dissimilar to the karate and dolphin networks. However the network Synthetic 2 has similar NMI and correlation functions as the football network. We have found two distinct shapes of the accuracy functions and a correspondence between synthetic and real-world networks regarding this shape. There exists however no simple explanation for this, as the networks have quite different mixing parameters and transitivity[7], see *Table 5.2*. We have therefore been unable to find an explanation for why these networks tend to have the same accuracy. The reason probably lies in higher-order network structures not captured by the network statistics.

6.4 General method for imperfect networks

This section is concerned with how the proposed methods perform in imperfect networks, found by adding missing and false edges in an uncertain network. A good method for detecting communities in imperfect networks should require only a small amount of certain information but also be able to handle false and missing edges. In the previous part, we investigated the relationship between performance and the amount of certain and uncertain edges. We concluded that LP-NFC and SP-NFC are good choices which generate the most accurate communities (of the tested methods) at a low complexity. The complexity aspect is important as many social networks are large and therefore we require fast methods that could be applied on large networks with at least hundreds of nodes.

[7]Further simulation runs on more synthetic networks were conducted to determine if there exists a relation between the transitivity and the effectiveness of the proposed methods. This revealed nothing conclusive but indicated that a certain transitivity is required to find the community structure with good accuracy. This was expected as well as with the mixing parameter because a higher transitivity and a high fraction of in-community edges create more distinct communities.

6.4.1 Uncertain networks with missing edges

We generalize the uncertain networks by allowing for missing edges using edge existence probabilities generated using the M-distribution. In this case, we are interested in how many edges that can be removed and still find an accurate result. A good method should be insensitive to missing edges and therefore have a flat NMI and correlation curve as functions of p_c, the probability that an edge is certainly missing or included in the network. In *Figure 6.7*, we present the resulting NMI curves from a simulation run on three real-world networks using NFC and CFC to merge candidate communities.

NFC merging results in that most community detection methods give the same NMI value, in comparison with CFC where there is more spread between the different curves. Studying particular community detection methods, we see that SP and LP still are good choices in combination with the NFC method. These methods also perform well using CFC to merge candidate clusters but still produce NMI scores lower than when using the NFC method. Using the same arguments as before, we therefore conclude that LP-NFC still is the best choice for detecting communities in uncertain networks.

We conclude by noting that the NMI scores are all quite low indicating that the communities found are not the same as indicated by the external labels. However, the problem with missing edges is a difficult problem in sparse networks as there are not many edges to start with in this type of networks. Removing these few edges clearly has a large impact on the community structures found by the community detection methods. Denser networks, as the football network, are more robust as there are many more edges between nodes and therefore many more edges can be removed while preserving the community structure. This is the explanation for the lower/unchanged NMI scores compared with the previous results using the L-distribution.

6.4.2 Adding false edges

The last model of imperfect networks analyzed in this thesis is a model with uncertain, false and missing edges. In this section, we study the accuracy of community detection in imperfect networks using SP-NFC and LP-NFC on the synthetic 1 and synthetic 2 networks. The methods have been chosen because of their good performance in uncertain networks as well as in networks with missing edges. It is therefore more probable that these methods also perform well in imperfect networks. The aim of this simulation experiment is to study how robust the community detection is in relation to the number of edges added and removed from the network.

The uncertain and missing edges are generated using the M-distribution and the false edges are added using the noisy adjacency matrix discussed in *Chapter 5.3*. The expected number of added false edges, N_f, by the noisy

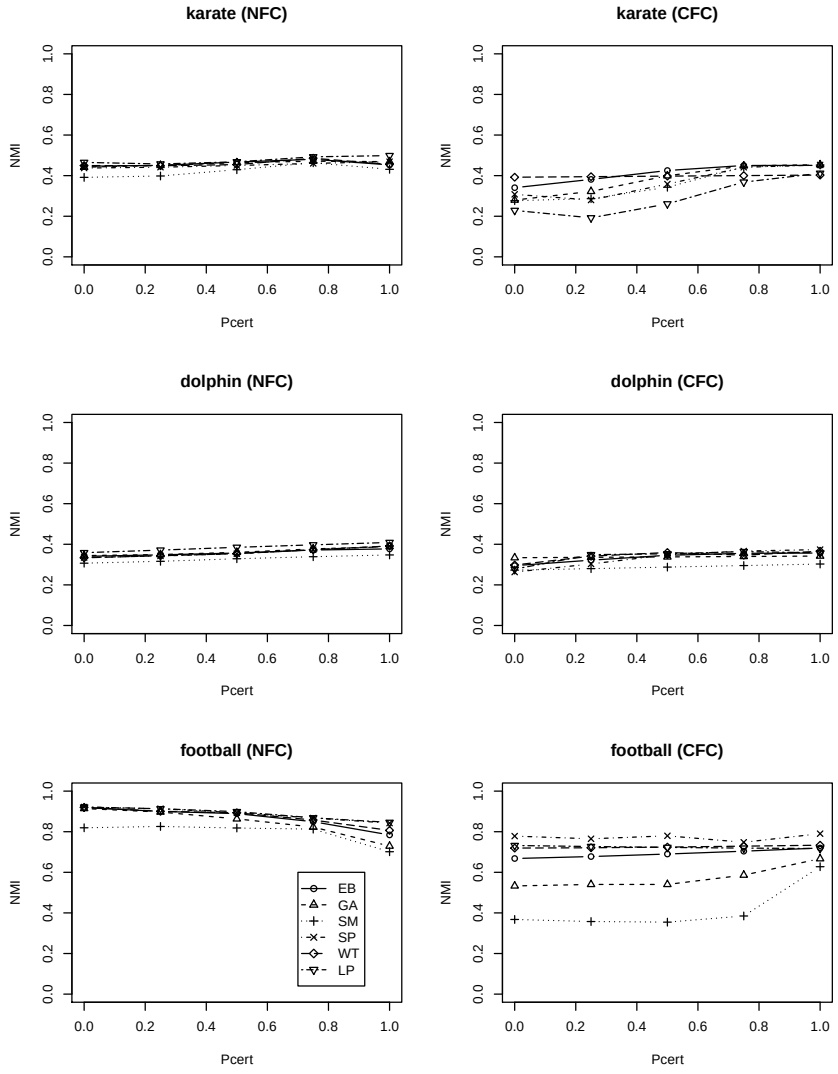

Figure 6.7: The normalized mutual information for the six different community detection methods used in combination with NFC or CFC on three different real-world networks with the M-distribution. The curves are estimated using non-parametric regression with the Gaussian kernel on the result of 30 runs each using 50 candidate networks.

adjacency matrix is given by

$$\mathbb{E}[N_f] = \frac{1}{2} n\,(n-1)\,p_n, \tag{6.1}$$

where p_n is the noise factor or the probability to add an additional false edge into the network. The noise factor assume values, $p_n \in \{0, 0.01, 0.05, 0.10, 0.15, 0.20, 0.25\}$ and the synthetic networks and the karate network have approximately 50 nodes.

Therefore the expected number of noisy edges added in the uncertain networks is $\mathbb{E}[N_f] \in \{0, 12, 60, 120, 180, 240, 300\}$. Comparing with *Table 5.2*, a third of the edges are false in the synthetic networks at $p_n = 0.17$ which is a considerable amount. As the M-distribution is used to remove edges (real and falsely added), it is important to recall that the number of edges removed varies between none ($p_c = 0$) to half ($p_c = 1$). This means that at all values of p_c and $p_n = 0.17$, a third of the edges are falsely added to the network.

In *Figure 6.8*, results are presented from 30 simulation runs on two different networks using 30 candidate communities with the SP-NFC and LP-NFC methods. The NMI are used for comparison between the different noise levels and the probability of certainty, p_c, which together control the number of false and missing edges.

The LP-NFC method is better at low $p_n \leq 0.05$ for all different p_c, at higher p_n the NMI values rapidly decrease to a very low level. SP-NFC method is clearly better and more robust in the region with high p_n. The method generates NMI values almost constant over all values of p_n, which shows that it is not sensitive to false edges.

An interesting solution is to combine SP and LP in some manner to get the good properties of LP in uncertain networks and of SP in imperfect networks. We still recommend the LP-NFC method for most detections of communities in uncertain and imperfect networks when not combining LP and SP, due to the large difference in computational complexity.

A similar experiment using CFC instead of NFC generated worse results than LP-NFC and has therefore not been presented in this chapter.

6.5 Unsupervised evaluation and confidence levels

In this final part, we present some minor results and discuss their implications in practical applications. Some natural complications when applying the proposed methods are validation of the obtained solution and confidence levels for the communities detected. Therefore some effort is needed to solve or at least propose possible solutions to these problems in this thesis.

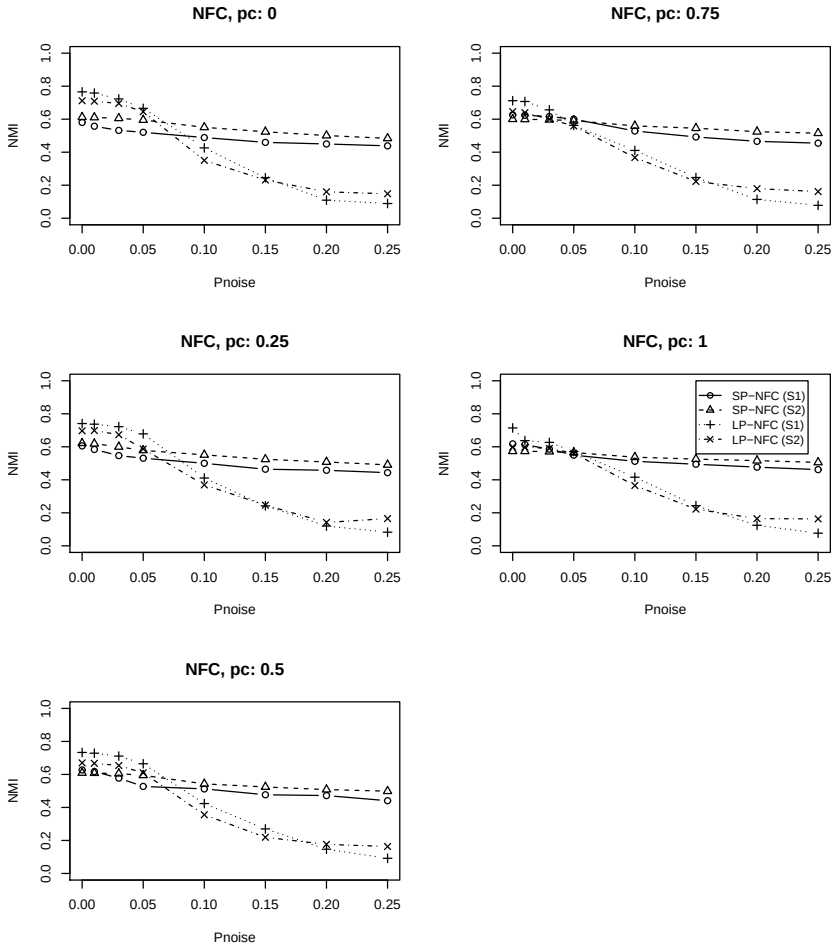

Figure 6.8: The normalized mutual information for two different community detection methods used in combination with NFC on two different synthetic networks with the M-distribution. The curves are estimated using non-parametric regression with the Gaussian kernel on the result of 30 runs each using 30 candidate networks.

6.5.1 Unsupervised evaluation

In this chapter, we have only used supervised evaluation since external labels are available in the networks used. In real-world applications however it is often difficult to find the correct solution to compare with, when constructing and testing new methods. An alternative solution to supervised evaluation is to use an unsupervised method by assigning node labels using something other than external labels. One possible solution is to use the communities detected by the same community detection algorithm on the certain network. For example, if we are interested in evaluating the SP algorithm on some network structure by generating existence probabilities, we could apply the

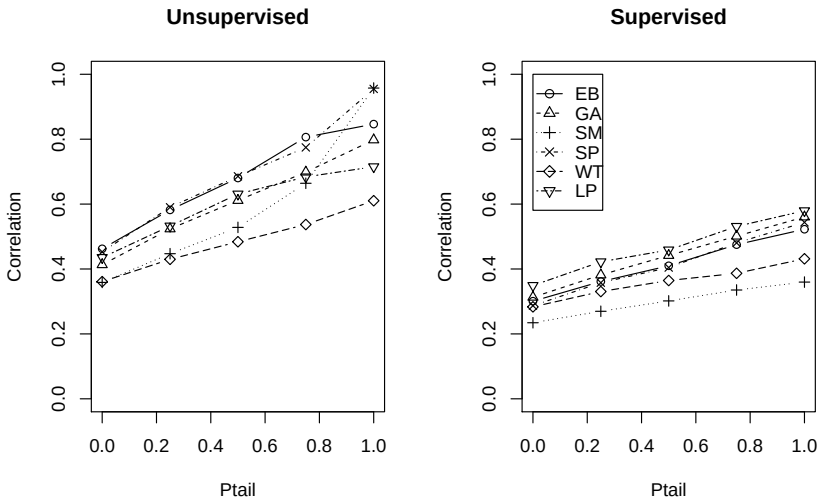

Figure 6.9: A comparison between the unsupervised and supervised correlation for evaluating community detection methods. The unsupervised correlation is calculated by estimating the external labels with the result obtained when applying the community detection algorithm on the certain version of the network. The results are found using the NFC merging method on 30 runs each using 50 candidate communities detected in the dolphin network with probabilities generated by the L-distribution.

SP algorithm on the certain network before probabilities have been added, then using the obtained solution as labels. A good method should be able to detect a similar structure in uncertain/imperfect networks as when applied on the certain network.

In Figure *6.9*, we present a comparison between supervised and unsupervised evaluation using the correlation measure. The underlying data is produced by 30 simulation runs on the dolphin network with edge existence probabilities generated by the L-distribution. NFC merging is used to combine 50 candidate communities detected in the uncertain network. The resulting correlations are quite dissimilar as they answer different questions. The supervised correlation measure was analyzed in *Figure 6.6* where we concluded that the LP method is the most promising on this type of networks. The LP method however does not perform well in the unsupervised version as the method is stochastic and the combined communities from 30 runs are compared with only one run on the certain network.

However, the SP and SM algorithms converge to the same solution when $p_t = 1$ as found on the certain version of the network. The remaining methods do not converge to the same solution and this is probably the result of the information loss incurred by the NFC method. When using the CFC method all methods, except the LP algorithm, converge to unity

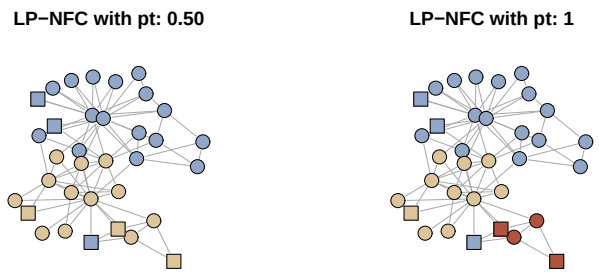

Figure 6.10: An example of the six most uncertain nodes in the estimated community structure in uncertain network. The LP-NFC method is used and combine 50 candidate communities. Some artifacts are visible in this example but neglecting these, we find reasonable results.

correlation as the tail probability approaches unity.

We conclude that this type of evaluation sometimes is necessary even if it is not as good as supervised validation and also answers a different kind of question. Supervised evaluation compares the obtained solution with the real solution, that may not be possible to find in the data given. Unsupervised evaluation however compares the obtained solution using community detection in uncertain networks with the result from using community detection on the certain version of the network. If the method is known to perform well on certain networks and yields high unsupervised correlation, it is possible to conclude that the method also perform well on uncertain networks.

6.5.2 Confidence level of communities

We end this chapter by demonstrating how to estimate confidence levels for merged candidate communities, as discussed in *Chapter 4.4*. We have merged 50 runs of the LP algorithm using the NFC method on the karate network. In *Figure 6.10*, we present graphically the six nodes with the lowest confidence using squares instead of circles to indicate the nodes.

Removing the obvious artifacts[8] and we are left with a good approximation of the uncertain nodes. Especially note that the lower right corner in each of the networks and that at $p_t = 0.5$ it is indicated that there could exist another partition inside the detected community.

[8]In the other simulations, we remove these artifacts but have not performed in this example to show some problems that are the results of the merging methods. For example, when calculating NMI, correlation, and MSE, we do not allow a node to be a member of one community and all its neighbors another.

This method, however qualitative in nature, does help to detect uncertain nodes that are placed in incorrect community. Thereby telling the person doing the analysis that more evidence is needed regarding this node to properly place it in a community. Finally note that when using the CFC method, we can find these confidence levels in a more quantitative manner as voting is used to place the nodes in the correct meta-cluster.

Chapter 7

Concluding Remarks

In this thesis, we have proposed a framework to analyze community structures in imperfect network data. As a part of this, a sampling method has been adopted to sample candidate networks from the ensemble of networks consistent with the uncertain information. The major innovations are the three proposed methods to merge the community structures found in the candidate networks. The methods are based on previous work in ensemble clustering and community detection. The introduced methods to merge communities are believed to have many applications in network related areas and is an important future area of research.

Summary and implications We have demonstrated that the merging methods introduced are able to greatly improve the performance of the label propagation algorithm. Using node-based fusion of communities to merge a few runs of the algorithm, results in an accuracy comparable to the more advanced spin glass algorithm at a much lower complexity. Merging methods are therefore a possibility for developing faster, more robust, and/or more accurate methods for community detection by combining different structures.

Faster and more accurate methods can be developed by merging the results from several different community detection methods or using the same method with different parameters. By varying parameters in the community detection algorithm and merging the resulting community structures, it is possible to find overlapping communities on many different scales and allowing for a better understanding of network properties. Using merging in combination with bootstrap re-sampling, outlined in Ref. [12], is one possible method to develop methods to calculate the robustness of community structures.

Another application of merging community structures are in imperfect and uncertain networks. To study these networks, we sample from an ensemble of networks consistent with the provided imperfect network information. We have evaluated the proposed merging methods used in conjuncture with

six different community detection methods. This include the results from three larger simulation studies on both real-world and generated synthetic networks. Existence probabilities have been simulated using two different distributions to evaluate the accuracy in detecting community structures in uncertain and imperfect networks.

Results indicate that the framework is able to recover the community structures of the test networks when half of the edges are known with certainty. The network structure largely influences the performance of the methods to find communities in imperfect network. However, Label Propagation (LP) used in combination with Node-based Fusion of Communities (NFC) is the most promising method and often performs well on the test networks. The LP-NFC method has been shown to be quite robust to missing edges but sensitive to false edges. It is the best candidate for community detection in imperfect networks. We have demonstrated how the framework can be used to adequately detect community structures with uncertain, incomplete and conflicting network data. It is stressed that these methods should be used to test hypotheses about the community structure together with a human supervisor, as some artifacts are bound to emerge when merging many different candidate communities.

Finally, we have demonstrated the possibility to evaluate the framework using networks without external labels. A method for finding confidence levels of the hypothesis that a certain node belongs to a certain community has also been implemented and discussed. This completes the framework supporting the detection of communities in imperfect networks from gathered data to the most probable community structure with confidence levels.

Future work Possible further research into the subjects discussed in this thesis include examining the methods using more and larger networks with more samplings. This to verify the results and conclusions given in this thesis on a broader spectrum of network types. Other important further studies are how the limit value (at which LP-NFC crosses SP-NFC in the NMI-graph) of p_n varies given the network structure. Another further development is to include the possibility of uncertain nodes and graph structures, where a probability for their existence is given in the same manner as for edges. These generalizations have natural applications in social network analysis but also in other network formalisms. The framework can handle these generalizations with smaller modifications of the sampling step.

Moreover, other methods for merging candidate communities can be developed using *Hybrid Bipartite Graph Formulation* [15] and *Markov Clustering Algorithm* (MCL) [3]. Some newer community detection methods using *Infomaps* [71] and *clique percolation* [72] show much promise as accurate and fast methods. Therefore, some effort should be given to evaluate these methods using the framework introduced. Other methods for sampling from probability intervals, using Markov Chain Monte Carlo, are also possible future extensions of this thesis.

Bibliography

[1] Stanley Wasserman and Katherine Faust. *Social network analysis : methods and applications*. Structural analysis in the social sciences, 8. Cambridge University Press, 1 edition, November 1994. 1, 2, 89, 95, 96

[2] John P. Scott. *Social Network Analysis: A Handbook*. SAGE Publications, January 2000. 1

[3] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5):75–174, February 2010. 1, 2, 3, 13, 15, 16, 19, 21, 22, 82, 89

[4] Mason A. Porter, Jukka-Pekka Onnela, and Peter J. Mucha. Communities in Networks. September 2009. 1, 15, 16

[5] Mark Newman. *Networks: An Introduction*. Oxford University Press, USA, 1 edition, May 2010. 1, 15, 16, 19, 89, 91, 93, 95, 96

[6] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, June 2002. 1

[7] Robert S. Weiss and Eugene Jacobson. A method for the analysis of the structure of complex organizations. *American Sociological Review*, 20(6), December 1955. 2

[8] Stuart A. Rice. The Identification of Blocs in Small Politicial Bodies. *The American Political Science Review*, 21(3), August 1927. 2

[9] M. E. J. Newman. Scientific collaboration networks. I. Network construction and fundamental results. *Physical Review E*, 64(1):016131+, June 2001. 2

[10] M. E. J. Newman. Scientific collaboration networks. II. Shortest paths, weighted networks, and centrality. *Physical Review E*, 64(1):016132+, June 2001. 2

[11] Usha N. Raghavan, Réka Albert, and Soundar Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E*, 76(3):036106+, September 2007. 3, 4, 22

[12] Martin Rosvall and Carl T. Bergstrom. Mapping Change in Large Networks. *PLoS ONE*, 5(1):e8694+, January 2010. 3, 14, 81

[13] A. Strehl and J. Ghosh. Cluster ensembles — a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, 3:583–617, March 2003. 3, 9

[14] Stefano Monti, Pablo Tamayo, Jill Mesirov, and Todd Golub. Consensus Clustering: A Resampling-Based Method for Class Discovery and Visualization of Gene Expression Microarray Data. *Machine Learning*, 52(1):91–118, July 2003. 3, 9, 41

[15] Xiaoli Z. Fern and Carla E. Brodley. Solving cluster ensemble problems by bipartite graph partitioning. In *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, pages 36+, New York, NY, USA, 2004. ACM. 3, 9, 10, 41, 82, 91

[16] J. Reichardt and S. Bornholdt. Statistical mechanics of community detection. *Phys Rev E Stat Nonlin Soft Matter Phys*, 74(1 Pt 2), July 2006. 4, 20, 21, 54

[17] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2 edition, November 2001. 6

[18] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. Addison Wesley, us ed edition, May 2005. 6, 7, 8, 9

[19] David Skillicorn. *Understanding Complex Datasets: Data Mining with Matrix Decompositions (Chapman & Hall/Crc Data Mining and Knowledge Discovery Series)*. Chapman & Hall/CRC, May 2007. 6, 92, 94

[20] A. K. Jain and R. C. Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988. 6, 8, 9, 55

[21] Richard A. Johnson and Dean W. Wichern. *Applied Multivariate Statistical Analysis (6th Edition)*. Prentice Hall, April 2007. 7, 8, 43

[22] J. B. MacQueen. Some Methods for Classification and Analysis of MultiVariate Observations. In Le M. L. Cam and J. Neyman, editors, *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967. 8

[23] Sandrine Dudoit and Jane Fridlyand. Bagging to improve the accuracy of a clustering procedure. *Bioinformatics (Oxford, England)*, 19(9):1090–1099, June 2003. 9

[24] Vikas Singh, Lopamudra Mukherjee, Jiming Peng, and Jinhui Xu. Ensemble clustering using semidefinite programming with applications. *Machine Learning*, December 2009. 9

[25] Xiaoli Z. Fern and Carla E. Brodley. Random Projection for High Dimensional Data Clustering: A Cluster Ensemble Approach. In *Proc. 20th International Conference on Machine Learning (ICML'03)*, Washington, August 2003. 9

[26] Ana L. N. Fred and Anil K. Jain. Data Clustering Using Evidence Accumulation. *Pattern Recognition, International Conference on*, 4:40276+, 2002. 9

[27] Moses Charikar, Venkatesan Guruswami, and Anthony Wirth. Clustering with Qualitative Information. In *FOCS '03: Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, pages 524+, Washington, DC, USA, 2003. IEEE Computer Society. 9

[28] Nir Ailon, Moses Charikar, and Alantha Newman. Aggregating inconsistent information: Ranking and clustering. *J. ACM*, 55(5):1–27, November 2008. 9

[29] N. Bansal, A. Blum, and S. Chawla. Correlation clustering, 2002. 9

[30] Paul Jaccard. Étude comparative de la distribution florale dans une portion des Alpes et des Jura. *Bulletin del la Société Vaudoise des Sciences Naturelles*, 37:547–579, 1901. 10

[31] Gerard Salton. *Automatic Text Processing: The Transformation Analysis and Retrieval of Information by Computer (Addison-Wesley series in computer science)*. Addison-Wesley Pub (Sd), 1988. 10

[32] Filippo Radicchi, Claudio Castellano, Federico Cecconi, Vittorio Loreto, and Domenico Parisi. Defining and identifying communities in networks. February 2004. 12, 17

[33] M. E. J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, June 2006. 14

[34] U. Brandes, D. Delling, M. Gaertler, R. Goerke, M. Hoefer, Z. Nikoloski, and D. Wagner. Maximizing Modularity is hard. August 2006. 15

[35] Benjamin H. Good, Yves A. de Montjoye, and Aaron Clauset. Performance of modularity maximization in practical contexts. *Physical Review E*, 81(4):046106+, April 2010. 15

[36] Santo Fortunato and Marc Barthélemy. Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, 104(1):36–41, January 2007. 15

[37] Martin Rosvall and Carl T. Bergstrom. An information-theoretic framework for resolving community structure in complex networks. *Proceedings of the National Academy of Sciences*, 104(18):7327–7331, May 2007. 15

[38] Jake M. Hofman and Chris H. Wiggins. Bayesian Approach to Network Modularity. *Physical Review Letters*, 100(25):258701+, June 2008. 15

[39] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69(2):026113+, February 2004. 17, 49, 50, 52

[40] Joshua R. Tyler, Dennis M. Wilkinson, and Bernardo A. Huberman. *Email as spectroscopy: automated discovery of community structure within organizations*, pages 81–96. Kluwer, B.V., Deventer, The Netherlands, 2003. 17

[41] Aaron Clauset, M. E. J. Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical Review E*, 70(6):066111+, December 2004. 18

[42] Ken Wakita and Toshiyuki Tsurumi. Finding community structure in mega-scale social networks: [extended abstract]. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 1275–1276, New York, NY, USA, 2007. ACM. 18

[43] Philipp Schuetz and Amedeo Caflisch. Multistep greedy algorithm identifies community structure in real-world and computer-generated networks. September 2008. 18

[44] M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 74(3):036104+, 2006. 19, 91

[45] Ernst Ising. Beitrag zur Theorie des Ferromagnetismus. *Zeitschrift für Physik A Hadrons and Nuclei*, 31(1):253–258, February 1925. 19

[46] Renfrey B. Potts. Some Generalized Order-Disorder Transformation. In *Transformations, Proceedings of the Cambridge Philosophical Society*, volume 48, pages 106–109, 1952. 19

[47] F. Y. Wu. The Potts model. *Reviews of Modern Physics*, 54(1):235–268, January 1982. 19

[48] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by Simulated Annealing. *Science, Number 4598, 13 May 1983*, 220, 4598:671–680, 1983. 20, 21

[49] Christian P. Robert and George Casella. *Monte Carlo Statistical Methods*. Springer-Verlag, 1 edition, August 1999. 20, 21

[50] Pascal Pons and Matthieu Latapy. Computing Communities in Large Networks Using Random Walks. In pInar Yolum, Tunga Güngör, Fikret Gürgen, and Can Özturan, editors, *Computer and Information Sciences - ISCIS 2005*, volume 3733, chapter 31, pages 284–293. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005. 21

[51] Aaron Clauset, Cristopher Moore, and M. E. J. Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 453(7191):98–101, May 2008. 25

[52] David Easley and Jon Kleinberg. *Networks, crowds, and markets : reasoning about a highly connected world*. Cambridge University Press, July 2010. 27

[53] Éloi Bossé, Jean Roy, and Steve Wark. *Concepts, Models, and Tools for Information Fusion*. Artech House, Inc., first edition, 2007. 27, 28, 29, 32

[54] J. M. Bernardo and Adrian Smith. *Bayesian Theory (Wiley Series in Probability and Statistics)*. Wiley, 1 edition, March 2000. 28

[55] G. Shafer. *A mathematical theory of evidence*. Princeton university press Princeton, NJ, 1976. 28, 29, 30

[56] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2009. 38, 60

[57] Gabor Csardi and Tamas Nepusz. The igraph Software Package for Complex Network Research. *InterJournal*, Complex Systems:1695, 2006. 38

[58] Mu Zhu and Ali Ghodsi. Automatic dimensionality selection from the scree plot via the use of profile likelihood. *Computational Statistics & Data Analysis*, pages 918–930, 2006. 43, 45

[59] Andrea Lancichinetti and Santo Fortunato. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 80(1):016118+, 2009. 49, 50, 52

[60] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. Benchmark graphs for testing community detection algorithms. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 78(4), 2008. 49

[61] W. W. Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33:452–473, 1977. 50, 52, 54

[62] David Lusseau, Karsten Schneider, Oliver J. Boisseau, Patti Haase, Elisabeth Slooten, and Steve M. Dawson. The bottlenose dolphin community of Doubtful Sound features a large proportion of long-lasting associations. *Behavioral Ecology and Sociobiology*, 54(4):396–405, 2003. 50, 52, 95, 96

[63] David J. C. MacKay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 1st edition, October 2003. 57

[64] Andrea Lancichinetti and Santo Fortunato. Community detection algorithms: A comparative analysis. *Physical Review E*, 80(5):056117+, November 2009. 57, 66

[65] Tristen Hayfield and Jeffrey S. Racine. Nonparametric Econometrics: The np Package. *Journal of Statistical Software*, 27, 2008. 60

[66] Larry Wasserman. *All of Statistics: A Concise Course in Statistical Inference (Springer Texts in Statistics)*. Springer, December 2003. 60

[67] Larry Wasserman. *All of Nonparametric Statistics (Springer Texts in Statistics)*. Springer, May 2007. 60

[68] W. Härdle. Applied Nonparametric Regression. E-book: http://www.quantlet.com/mdstat/scripts/anr/pdf/anrpdf.pdf. 60

[69] A. C. Davison and D. V. Hinkley. *Bootstrap Methods and Their Application (Cambridge Series in Statistical and Probabilistic Mathematics , No 1)*. Cambridge University Press, 1 edition, October 1997. 60, 69

[70] Jeff Racine. Consistent Significance Testing for Nonparametric Regression. *Journal of Business and Economic Statistics*, 15:369–378, 1997. 61

[71] Martin Rosvall and Carl T. Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4):1118–1123, January 2008. 82, 94

[72] Gergely Palla, Imre Derényi, Illés Farkas, and Tamás Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, June 2005. 82

[73] Douglas B. West. *Introduction to Graph Theory (2nd Edition)*. Prentice Hall, 2 edition, September 2000. 89, 93

[74] U. Brandes and T. Erlebach. *Network Analysis : Methodological Foundations (Lecture Notes in Computer Science)*. Springer, March 2005. 89, 96, 97

[75] Eric D. Kolaczyk. *Statistical Analysis of Network Data: Methods and Models (Springer Series in Statistics)*. Springer, 1 edition, March 2009. 89, 94, 97

[76] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences)*. W. H. Freeman, first edition edition, January 1979. 91

[77] David A. Grossman and Ophir Frieder. *Information Retrieval: Algorithms and Heuristics (The Information Retrieval Series)(2nd Edition)*. Springer, 2nd edition, December 2004. 92

[78] Charles. Spectral Partitioning: The More Eigenvectors, The Better. In *Design Automation, 1995. DAC '95. 32nd Conference on*, pages 195–200, 1995. 92

[79] A. Ng, M. Jordan, and Y. Weiss. On Spectral Clustering: Analysis and an algorithm. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, pages 849–856. MIT Press, 2001. 92

[80] J. R. Ullmann. An Algorithm for Subgraph Isomorphism. *J. ACM*, 23(1):31–42, January 1976. 93

[81] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, STOC '71, pages 151–158, New York, NY, USA, 1971. ACM. 93

[82] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, December 1959. 93

[83] M. E. J. Newman. A measure of betweenness centrality based on random walks. *Social networks*, 27(1):39–54, January 2005. 94

[84] Geoffrey R. Grimmett and David R. Stirzaker. *Probability and Random Processes*. Oxford University Press, USA, 3 edition, August 2001. 94

[85] M. E. J. Newman and Juyong Park. Why social networks are different from other types of networks. May 2003. 95

[86] M. E. J. Newman. Analysis of weighted networks. July 2004. 95

[87] Tore Opsahl, Filip Agneessens, and John Skvoretz. Node centrality in weighted networks: Generalizing degree and shortest paths. *Social Networks*, 32(3):245–251, July 2010. 95

[88] L. Freeman. Centrality in social networks conceptual clarification. *Social Networks*, 1(3):215–239, 1979. 95, 97

[89] Gert Sabidussi. The centrality index of a graph. *Psychometrika*, 31(4):581–603, December 1966. 96

[90] Phillip Bonacich. Power and Centrality: A Family of Measures. *The American Journal of Sociology*, 92(5):1170–1182, 1987. 97

Appendix A

Graph theory

Networks are analyzed by the use of the mathematical theory of graphs. In this section some elementary results from graph theory and linear algebra are reviewed. These methods are used in combination with the network methods presented in the next appendix and with the methods presented in the background on clustering and community detection in *Chapter 2*.

For a more extensive discussion of graph theory see Ref. [73], for more on graph theory in connection with social network analysis see e.g. Refs. [1, 74, 5, 3, 75].

A.1 Elementary graph theory

A *graph* is defined as an object $G = (V, E)$ consisting of a node (vertex) set $V(G)$ and an edge (link) set $E(G)$. The number of nodes, n, is equal to the size of the node set, $n = |V|$, and the number of edges, m, in the network is similarly $m = |E|$. Two nodes are said to be *adjacent*, *neighbors*, or *connected* if there exist an edge between them. If all k nodes in the graph are adjacent, the graph is said to be *k-complete*. A graph is *simple*, i.e. loop-less and lacks multiple edges, if there is at most one edge between each pair of nodes and no node is neighbor with itself.

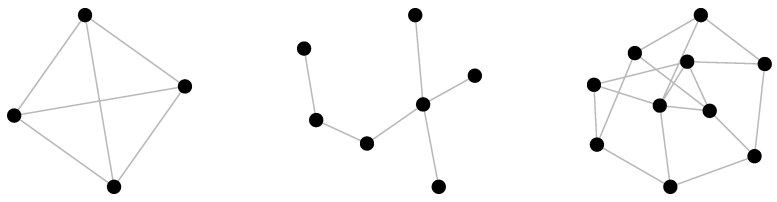

Figure A.1: Visualizations for three well-known graph(structures); (a) a 4-complete graph, (b) a tree, and (c) the Petersen graph.

A *walk* is an ordered set of alternating nodes and edges that starts in one node i and ends in another node j. If the walk only transverses each node

at most once, it is called a *path*. A *k-cycle* is a path where the first and last nodes are the same, and the path contains k edges. A graph is *connected*, if there exists a path between any given pair of nodes. The shortest path between two nodes is the *geodesic* and the longest geodesic is the *diameter* of the graph.

A graph without cycles is called a *tree* (or a forest if unconnected). A *subgraph*, G', of a graph G contains all edges that connect a subset of the node set, i.e. $V'(G) \subset V(G)$ such that $E'(G) \subset E(G)$ contains all edges connecting the nodes in $V'(G)$. One says that the edge set is spanned by the set of nodes. Two subgraphs are therefore disjoint and not connected. A *k-clique* is a k-complete subgraph.

A.2 Algebraic graph theory

The *adjacency matrix*, $\mathbf{A} = [A_{ij}]$, is often used to represent a graph. Here $A_{ij} = 1$ denotes the existence of an edge between nodes i and j, otherwise $A_{ij} = 0$ if no edge exist. An adjacency matrix is always symmetric with a zero-diagonal for simple graphs. The adjacency matrices for two of the graphs presented in *Figure A.1* are

$$\mathbf{A}_a = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}, \qquad \mathbf{A}_b = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Extending the adjacency matrix, a *weighted graph* is defined by using A_{ij} as the weight of the edge between nodes i and j. If there is no edge between the two nodes $A_{ij} = 0$ as before. The simplest analogy for a weighted graph is that the weight is some kind of cost or distance. Edges with large weights should be avoided if possible, to minimize the total cost or length of crossing.

The *degree* of a node, k_i, is the total number of edges that are connected to the node i and the density, ρ, the mean degree of a graph is defined as

$$k_i = \sum_{j=1}^{n} A_{ij}, \qquad \rho = \frac{\frac{1}{n}\sum_{i=1}^{n} k_i}{n-1} = \frac{\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij}}{n-1}, \tag{A.1}$$

where A_{ij} is the element in row i, column j in the adjacency matrix and n is the number of nodes in the graph.

The *Laplacian matrix* of a graph is defined as

$$L_{ij} = \begin{cases} k_i & \text{for } i = j \\ -1 & \text{for } i \neq j \text{ and } A_{ij} \neq 0 \\ 0 & \text{otherwise} \end{cases}, \tag{A.2}$$

or equivalently $L_{ij} = k_i\delta_{ij} - A_{ij}$, where δ_{ij} is the Kronecker delta function. Since both the adjacency matrix and the Laplacian matrix are symmetric, results from standard linear algebra state that the eigenvalues are real and it is also possible to show that they are non-negative, $\lambda_i \geq 0$. This result is used in the community detection method proposed in Ref. [44], which is discussed in detail in *Section 2.4.3*. [5]

A.3 Spectral graph partitioning

The graph partitioning problem is concerned with how to divide a graph, $G = (V, \mathbf{W})$, into k disjoint partitions (or parts) by finding a *cut set* that minimizes the sum of weights of cut edges. The cut set is the set of edges that if removed results in the k components corresponding to the partitions of the graph.

Let $\mathbf{W} = [W_{ij}]$ denote a $n \times n$-similarity matrix and $P = \{P_1, P_2, \ldots, P_k\}$ denote the partitions of the nodes, i.e. $\bigcup_i^k P_i = V$. The sum of the weights of the required *cut of partition* is $\mathsf{cut}(P, \mathbf{W}) = \sum W_{ij}$, where nodes i and j do not belong to the same cluster.

There are several methods to find the cut set of edges that minimizes the total cut, but the problem in general[1] is NP-complete and is therefore time consuming to solve. An estimate of the cut set can be found by using a result from linear algebra called an *eigendecomposition* on the adjacency matrix of the graph. [15]

Eigendecomposition is a special case of a more general result the *Singular Value Decomposition* (SVD), which is presented in *Theorem 3*. The SVD is a change of coordinates from the original basis to a basis comprised by the eigenvectors of the matrix. The square root of the eigenvalues are the new coordinates corresponding to the new basis.

Theorem 3 (Singular Value Decomposition)**.** *Any $m \times n$-matrix $\mathbf{X}$ can be factored into*

$$\mathbf{X} = \mathbf{Q}_1 \Sigma \mathbf{Q}_2, \tag{A.3}$$

where the columns of $\mathbf{Q}_1$ are the eigenvectors of $\mathbf{X}\mathbf{X}^\top$ and the columns of $\mathbf{Q}_2$ are eigenvectors of $\mathbf{X}^\top\mathbf{X}$. The r singular values on the diagonal of Σ are the square roots of the non-zero eigenvalues of both $\mathbf{X}\mathbf{X}^\top$ and $\mathbf{X}^\top\mathbf{X}$.

The theorem states that a matrix can be decomposed by the following sum

[1] When k_i=2, the problem is solved in polynomial time by matching. [76]

$$\mathbf{X} = \mathbf{Q}_1 \Sigma \mathbf{Q}_2 = \sum_{i=1}^{r} u_r \sigma_r v_r^\top, \tag{A.4}$$

and as the eigenvalues, σ_i, are sorted the terms are decreasing. A natural approximation to the value of the matrix $\mathbf{X}$ is therefore found by truncating the sum in (A.4) at some $k < r$, i.e. only using the k eigenvectors associated with the k largest eigenvalues. [77]

The adjacency and Laplacian matrices are positive definite, i.e. they have positive eigenvalues, and this simplifies the expression in (A.3) to a special case which is the slightly simpler *eigendecomposition*. This method is used on sparse square adjacency or similarity matrices, $\mathbf{X}$, by using the following expression

$$\mathbf{X} = \mathbf{P} \Lambda \mathbf{P}^{-1}, \tag{A.5}$$

where $\mathbf{P}$ is a matrix with orthogonal eigenvector as columns and $\mathbf{\Lambda}$ is a matrix with eigenvalues as the diagonal. As before the expression can be expanded to a similar form to (A.4). Using the approximation by truncating the series at some $k < r$, i.e. using the first k eigenvectors, the matrix is approximated by a low-dimensional representation. [19]

Using this method on a similarity matrix, results in an approximation matrix which can be used in clustering, e.g. by using the Fiedler[2] vector, k-means clustering or eigenvector clustering. [19, 78]

When partitioning graphs, the adjacency matrix is instead approximated by SVD or eigendecomposition. It turns out that one can find a good graph partitioning by using the second largest eigenvector in the Laplacian matrix. This will define the semi-optimal cut and thereby relaxing the NP-hard *discrete graph partitioning problem* (see below for details). In Ref. [78], the authors discuss the problem and found that using more than one eigenvector is better. In this paper, the approach from Ref. [78], using k eigenvectors (as many eigenvectors as partitions), is used to partition networks (see *Chapters 2.2* and *4.3* for details). [79]

A.4 Common problems related to graphs

This short review of graph theory is concluded by discussing some common graph related problems and algorithms.

[2]The eigenvector corresponding to the second smallest eigenvalue of the Laplacian matrix, is called the *Fielder vector*. This method is used by sorting the elements of the Fiedler vector and separating them into two parts, at some arbitrary point, e.g. zero. The row numbers corresponding to each element in the cluster sets are then the nodes that are members of the cluster.

A.4.1 Maximum common subgraph

The *maximum common subgraph* of two graphs, G and H, is often used to find how similar two graphs are. That is a subgraph of both G and H with the maximum number of nodes. The standard algorithm for this problem is *subgraph isomorphism detection* presented in Ref. [80]. An *isomorphism* is bijection $f : V(G) \to V(H)$, where G and H are simple graphs, such that $ij \in E(G)$ if and only if $f(i)f(j) \in E(H)$. This problem is difficult to solve for most problems and requires a large amount of computations, one can show that this problem is NP-complete. [81]

A.4.2 Kernighan-Lin graph partitioning

Graph partitioning problems are also common in the fields of mathematics and computer science. The simplest possible graph partitioning problem is the bisection of a graph, i.e. dividing it into two groups such that the edges between the groups are as few as possible. The most well-known method to solve this problem is the *Kernighan-Lin* algorithm which is based on five steps outlined in *Algorithm 8*.

Algorithm 8 Kernighan-Lin

(i) randomly divide the nodes into two groups,

(ii) calculate how the cut size would change if two nodes i and j where interchanged, for each pair of nodes i and j,

(iii) interchange the two nodes that corresponds to the largest reduction or smallest increase of the cut size,

(iv) repeating (ii) with the restriction that a node can only be moved once, until all nodes have been moved,

(v) when all nodes have been moved once, rerun the algorithm using the grouping with the smallest cut size generated in (iii).

The drawback with this algorithm is its high complexity, $\mathcal{O}(n^4)$, and that is does not always generate the optimal division of the graph. [5]

A.4.3 Shortest path

Another common problem is to find the shortest path between two points, which is encountered in calculations of the geodesic and the diameter of a graph. The most used algorithm to find the shortest path is the greedy Dijkstra's algorithm. Which is both of low complexity and find the optimal path for most graphs. [82, 73, 5]

A.4.4 Random walk on graphs

A random walk is a simple stochastic process (Markov Chain) on a *chain* (a kind of walk). On a graph the situation is complicated with nodes that have different degrees. Using a weighted network, the surfer/walker would like to use the edges with the smallest (or largest) weight more often than others. One can show that the transition matrix of this Markov Chain is the adjacency matrix of the graph. Using standard techniques from the study of stochastic processes, the stationary probabilities, and other interesting quantiles are simple to calculate.

For more information on random walks see Refs. [19, 75, 71, 83] and Markov Chains, see e.g. Ref. [84].

Appendix B

Social Network Analysis

Social Network Analysis (SNA) is the combination of work carried out in many different sciences during the last century. Some results originate from sociology, psychology, physics and mathematics. Social networks differs from complex networks in general by their intricate clustering structure (social communities) and their positive correlation between the degrees of adjacent nodes. The first difference is the main topic of this thesis and is discussed in more detail in *Chapter 2.3*. The meaning of the second is that nodes with high degree often have high degree neighbor nodes. [85]

Important research questions in SNA includes the flow of information on the network structure, finding leaders and authority figures, finding the clustering (community) structure of the network and modeling its dynamic behavior. Visualizing a large Social Network (SN) is a very challenging task, therefore some quantifiable measures are needed to describe the network structure. One example of such measures are *centrality measures* which can be applied on a node or edge and for the entire network. Generalized measures for weighted graphs also exist see e.g. Refs. [86, 87]. The most common centrality measures in community detection methods are the *Closeness*, *Betweenness* and *Eigenvector centrality*. [88, 1, 5]

Example 9. A comparison of the three measures is presented above, in *Figure B.1*. The network is composed of 62 dolphins, living off Doubtful Sound, New Zealand, which shares an edge if they have some frequent association with each other. The node degree, the closeness centrality, the betweenness centrality, and the eigenvalue centrality are presented for the same network for comparing the different measures. [62]

The node degree is higher in the center of the two clusters. The closeness centrality is somewhat more difficult to interpret, when most nodes have the same closeness. The betweenness measure clearly indicated the two nodes that acts as a bridge between the two clusters. The flow of information is large through these nodes and severing the edge between then would separate the clusters further apart. The eigenvector centrality is highest in

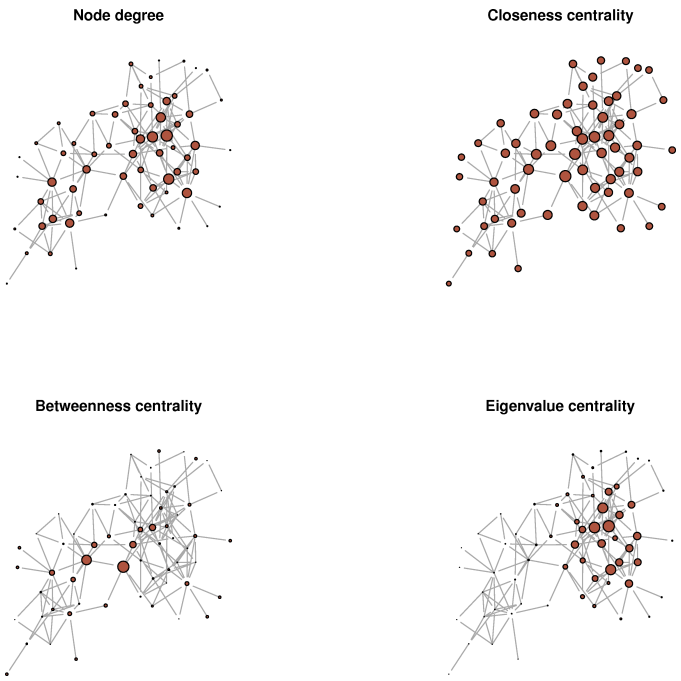

Figure B.1: The degree distribution, closeness centrality, betweenness centrality, and eigenvector centrality for an association network of 62 dolphins [62].

the middle of the larger upper cluster, indicates somewhat larger prestige to be one of the center nodes of that cluster.

All the compared centrality measures capture different aspects of the structure of the social network. This implicates that multiple measures should be used when analyzing large networks, that can not be visualized in a clear way.

For more information about centrality measures not discussed in this appendix, e.g. Katz centrality, PageRank, Hubs, and authorities, transitivity, reciprocity, structural balance, similarity etc., see Refs. [1, 5, 74].

B.1 Closeness centrality

Closeness centrality measure how central a node is by the distance to all the other nodes in the network. The measure is defined as the inverse distance to all other nodes in the network [89]

$$c_c(i) = \left[\sum_{j \in V} d(i,j)\right]^{-1} (n-1)\,, \tag{B.1}$$

where $i, j \in V(G)$, the node set of the graph and $d(i,j)$ is the geodesic

distance between nodes i and j. To calculate the closeness of a node, one must therefore calculate the geodesic between all pairs of nodes in the network. This makes the estimation of the closeness centrality quite time consuming, first to calculate all the shortest paths using the Dijkstra's algorithm (see previous appendix) and then comparing the results. One can show that the complexity of closeness calculations is $\mathcal{O}(n^2 \log n + nm)$. [75]

B.2 Betweenness centrality

Betweenness centrality uses the number of geodesics between pairs of nodes, in which a single node v_i or edge participate e_i as a measure of centrality. A more central (important) node or edge should have many paths transversing it. This in some sense accounts for the amount of information that flows thorough an edge or a node. This measure is defined by Refs. [88, 75] as

$$c_b(i) = \sum_{s \neq t \neq i \in V} \frac{\sigma(s,t|v)}{\sigma(s,t)} \frac{2}{(n-1)(n-2)}, \tag{B.2}$$

where $\sigma(s,t|i)$ is the total number of shortest paths between s and t passing through i (an edge or node), and $\sigma(s,t) = \sum_i \sigma(s,t|i)$ i.e. the total number of unique shortest paths between nodes s and t. The simplest implementation of this calculation is of order $\mathcal{O}(n^3)$, but a clever solution discussed in [74] reduces the complexity to $\mathcal{O}(nm)$. A related measure is the max-flow centrality measure in which all paths between pairs of nodes intersecting a selected node v_i or edge e_i are counted.

B.3 Eigenvector centrality

Eigenvector centrality accounts for the prestige of a node in the network. This measure is closely related to the PageRank algorithm that is (was) used in the search engine Google and is defined following Refs. [90, 75] as

$$c_e(j) = \alpha \sum_{\{i,j\} \in E(G)} c_{Ek}(i), \tag{B.3}$$

where $\mathbf{c}_{Ek} = (c_{Ek}(l))^\top$, $l = 1, 2, \ldots, n$, are the solution to the eigenvalue problem $\mathbf{A}\mathbf{c}_{Ek} = \frac{1}{\alpha}\mathbf{c}_{Ek}$, and $\mathbf{A}$ is the adjacency matrix of the network. Often one uses the largest eigenvalue as α^{-1} and it is a good choice for simple undirected graphs [90]. Standard methods from computational linear algebra can solve this problem in $\mathcal{O}(n^2)$. [75]

Appendix C

Notation

$\mathbf{X} = [X_{ij}]$	Matrix, $\mathbf{X}$, with X_{ij} as the element on row i and column j.
$\mathbf{x} = (x_i)$	Vector, $\mathbf{x}$, with x_i as the element with index i.
$\mathbb{P}[X]$	Probability of $x = X$.
$\mathbb{E}[X]$	Expected value of a random variable X.
$\mathbb{V}[X]$	Variance of a random variable X.
$\mathsf{MSE}[\mathsf{X}]$	Mean Square Error of X, i.e. $\mathsf{MSE}[\mathsf{X}] = \mathbb{E}\left[(X - \mu_X)^2\right]$.
$\mathsf{Cov}[\mathsf{X}, \mathsf{Y}]$	Covariance of X and Y, i.e. $\mathsf{Cov}[\mathsf{X}, \mathsf{Y}] = \mathbb{E}\left[(X - \mu_X)(Y - \mu_Y)\right]$.
$\delta(\cdot)$	Kronecker delta function, $\delta(x, y) = 1$ if and only if all variables in the argument are equal and $\delta(x, y) = 0$ otherwise.
$\lceil \cdot \rceil$	Ceiling function, rounds upwards to the nearest integer.
$\mathbf{A}$	Adjacency matrix, $A_{ij} = 1$ if an edge between nodes i and j exist and 0 otherwise.
$G(V, E)$	A graph with node (vertex) set, $V = V(G)$, and edge set, $E = E(G)$.
k_i	Degree of node $i \in V(G)$.
n	Number of nodes in a network G, $n = \lvert V(G) \rvert$.
m	Number of edges in a network G, $m = \lvert E(G) \rvert$.
$\mathcal{N}(\mu, \sigma^2)$	Normal distribution with mean μ and variance σ^2.
$\mathcal{U}(a, b)$	Uniform distribution between in the interval $[a, b]$.
$\mathcal{B}(p)$	Bernoulli distribution with success probability p.

Appendix D

Abbervations

NP	Non-Polynomial complexity.
TFC	Two-step Fusion of Communities (*Section 4.3.1* on p. 39)
NFC	Node-based Fusion of Communities (*Section 4.3.2* on p. 41)
CFC	Community-based Fusion of Communities (*Section 4.3.3* on p. 42)
EB	Divisive algorithm based on betweenness (*Section 2.4.1* on p. 16)
GA	Greedy agglomerative method (*Section 2.4.2* on p. 18)
SM	Spectral method (*Section 2.4.3* on p. 18)
SP	Spin glass algorithm (*Section 2.4.4* on p. 19)
WT	Random walk on networks (*Section 2.4.5* on p. 21)
LP	Label propagation (*Section 2.4.6* on p. 22)
NMI	Normalized Mutual Information (*Section 5.4.2* on p. 55)