Universiteit Leiden

Computer Science

Ensemble community detection in real-world networks

Name: Lars Hopman
Date: 28/08/2019

1st supervisor: Frank Takes
2nd supervisor: Vincent Traag (CWTS)

MASTER'S THESIS

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

Community detection is an important subject in the field of complex networks. It has a wide variety of applications ranging from detecting groups in (online) social networks, to finding proteins with similar functionality inside a biological cell for a protein interaction network. In this thesis, we discuss four methodological problems which make community detection a non-trivial task. In response to these problems, we present two solutions. As a first solution, we propose a novel method for finding communities by creating an ensemble of partitions of different community detection algorithms using fast consensus clustering. Our method shows improvements on commonly used metrics in literature, such as conductance and modularity for networks with a relatively low clustering efficient. This approach is validated on over 40 real-world networks. As a second solution, we introduce a new method for assessing the quality of a partition in real-world networks. This is a problem due to the fact that multiple quality metrics exist. To solve this challenge, we use pareto fronts to find trade-offs between multiple metrics. Because similar quality metrics can be obtained for completely different partitions, the analysis should be done in context of the community size distribution which is included in our method. Our method of assessing the quality of a partition enables us to find trade-offs and provides a better understanding of the resulting partition.

Contents

Chapter 1

Introduction

Networks are structures which can be used to represent connected systems by using nodes and edges. In fact, a lot of structures can be modelled as a network. Examples include (online) social networks, infrastructure networks and biological networks. To retrieve knowledge from these networks, a lot of research is done in the field of network analysis.

An important topic in network analysis is community detection. Community detection is about finding groups within a network that contain nodes which are more tightly connected to each other compared to the edges going outside of the group. Practically, community detection helps finding for example functional modules in protein protein interaction networks. Figure 1.1 shows the Zachary Karate club network [1]. The nodes in this network represent the members of the club. An edge represents if an interaction outside the club between two members exists. Communities are represented with different colors. The division of a network in communities such that each node belongs to one community is called a partition. While the example from Figure 1.1 seems quite simple, various challenges arise with community detection algorithms. In this thesis, we define four problems which make community detection a hard problem to solve.

The first problem is that a lot of different community detection algorithms exist. These methods adopt different types of approaches, with options ranging from statistical to optimization-based. Solid overviews are provided by recent surveys [2, 3, 4]. In the past years, a lot of research has been done, leading to new algorithms [5, 6, 7]. The wide range of algorithms is a problem, because eacch methods still has problems of its own. For example, they can contain random components, which result in different solutions for multiple runs of the same algorithm. Other approaches can become blind for communities of specific sizes [8]. The second challenge is that there can exist multiple good solutions. This means that two completely different results can

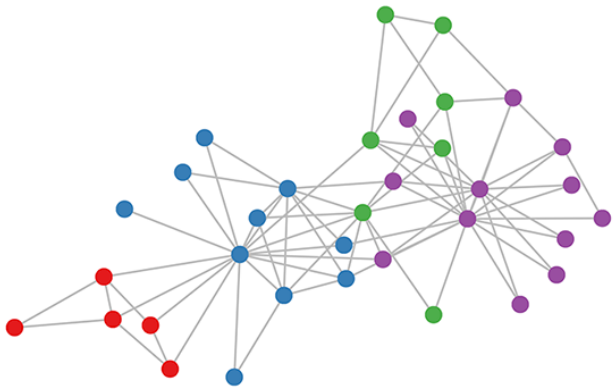


Figure 1.1: An example of the Zachary karate club network. The communities are displayed with different colors.

be interpreted as equally good. This is a problem because this makes comparing solutions in an objective way hard. This leads to the third problem. There are a lot of different definitions of a community. This results in a large number of available quality metrics. This makes solutions hard to compare because an improvement in one quality metric often leads to a decrease in another. The final problem is that some methods work better for specific network structures. When working with real-world networks, it is often hard to know what to expect in terms of structure of the network. This makes it hard to select the best community detection algorithm.

Our research question touches all the problems mentioned before: **Can existing community detection methods be combined in such a way, that the performance can be improved on commonly used quality metrics?**

To research this problem, we dive into the most important methods used in the literature. Of course, each method has its own advantages and disadvantages. Next, we observe the wide range of available metrics. To validate our work, we also need to test our implementation. We are aware that methodological research in community detection is often validated by the Lancichinetti–Fortunato–Radicchi (LFR) benchmark [9]. This benchmark creates a network with artificial communities. As these communities are known a-priori, algorithm results can be compared with the expected values. However, in this thesis our focus will be on real-world networks. This is a conscious choice because this gives us an intuition of the problems occurring with real-world data sets. Finally, both the proposed method and the current state-of-the-art methods have to be tested on these networks. It allows us to pinpoint why a method performs like it does, relating performance to network properties.

The outline of this thesis is as follows. First, we will provide some definitions of networks and community detection in Chapter 2. Next, we discuss related work (Chapter 3) to get an understanding of the field. We categorise the different types of algorithms, discuss them and dive into one algorithm within that category. The quality metrics popular in the literature are also discussed.

Chapter 4 will give an overview of the data sets used in the experiments. For this thesis, we will focus on a range of real-world networks from different categories. Further, Chapter 5 shows our approach. We introduce the algorithms used for our proposed method of ensemble community detection. Next, we elaborate on the functionality of the algorithm and how we asses the quality of the resulting partition. In Chapter 6, the experimental setup is discussed as well as the results from these experiments. We show when the proposed method results in increased performance and how to find the best partition when considering multiple criteria at the same time. Finally in Chapter 7, we draw conclusions based on these results and provide suggestions for future research.

Chapter 2

Definitions

This section provides a background and gives definitions for networks and communities. It should give a more formal background into networks and community detection.

2.1 Networks

Networks are simply a tuple of nodes and edges. Nodes may represent people, corporate boards or proteins, while edges may represent connections like friendships, bridges, or molecular binding interactions. Networks can be used to model a wide variety of applications such as from infrastructure networks, social networks and biological networks. We will adopt both the set notation and the matrix notation. A definition of a network can be found in Definition 1.

Definition 1 *A network G, is an ordered pair (V, E), where V is called the vertex set and E is called the edge set ($E \subseteq V \times V$.). Within each $e \in E$, we associate two vertices $u, v \in V$.*

Further, the number of nodes is defined as $n = |V|$ and the number of edges is defined as $m = |E|$. The adjacency matrix of network G is denoted as A. A is is an n×n matrix. If node i and node j are neighbours, $A_{ij} = 1$. This can be extended with weights where $A_{ij} = w_{ij}$. An example weighted network is a road network containing the distance between cities as weight. If node i and j are not connected, $A_{ij} = 0$.

One important aspect of networks is direction. Figure 2.1 graphically shows the differences between directed and undirected networks. To give a practical example, friendships can be modelled through an undirected network, as being friends (often) goes both ways. Modelling one-way streets in a road network however, can be done by using a directed network. More formally, a network is undirected if Equation 2.1 holds. This equation basically says that both edges (u, v) and (v, u) should be in edge set E meaning that the edge set is symmetric. This means that an edge goes both ways. If this does not hold, the network can be considered directed.

Figure 2.1: The difference between an undirected and a directed network visualised.

$$(u, v) \in E \Rightarrow (v, u) \in E. \tag{2.1}$$

Network properties

Networks have various properties. These properties can help to describe a network and its structure. In this subsection, we discuss some of the most important properties.

One of these properties is the degree of a node. Equation 2.2 shows that the degree is the number of edges connected to a node. Average degree (Equation 2.3) calculates this metric over the complete network. For example, how many friends a person has on average within a social network. The (2) in Equation 2.3 should be added for undirected networks, and can be discarded for directed networks.

$$deg(v) = \sum_j a_{ij}, \tag{2.2}$$

$$avg\ deg(G) = \frac{(2)m}{n} \tag{2.3}$$

Equation 2.4 gives the node clustering coefficient. This value indicates the extent to which a node v forms triangles with its neighbours. The network clustering coefficient calculates this metric over the complete network (Equation 2.5), which is the average of all the node clustering coefficients. The clustering coefficient is measures the degree to which nodes in a graph tend to cluster together.

$$C(v) = \frac{2 \cdot |(u, w) \in E : (u, v) \in E \wedge (v, w) \in E|}{deg(v) \cdot (deg(v) - 1)} \tag{2.4}$$

$$C(G) = \frac{1}{n} \cdot \sum_{v \in V} C(v) \tag{2.5}$$

Another metric is the network density (Equation 2.6). The density of a network decreases when the number of nodes within a network increases. Again, a distinction has to be made between directed and undirected networks. We can discard the (2) by the first and have to include it with the latter.

$$density(G) = \frac{(2)m}{n(n-1)} \tag{2.6}$$

2.2 Communities

The informal definition of a community is that nodes in a community are more connected to each other than to nodes in the rest of the network. To give a formal definition, we have to introduce some additional concepts like subgraphs (Definition 2). For the definition of a community, we follow Fortunato [10].

Definition 2 *A subgraph S of a network G is a network whose node set $V(S)$ is a subset of the node set $V(G)$, that is $V(S) \subseteq V(G)$, and whose edge set $E(S)$ is a subset of the edge set $E(G)$, that is $E(S) \subseteq E(G)$ such that each node in edge set E(S) is included in V(S).*

We define the number of nodes in a subgraph as n_s and number of edges in a subgraph as m_s. We define the internal degree k_v^{int} of a node in subgraph S as the number of edges connecting node v to other nodes inside subgraph S. For the external degree, we define k_v^{ext} as the number of edges from nodes within S connected to nodes outside S. Next, The internal degree k_{int}^S and external degree k_{ext}^S are defined as the sum of the degrees of its nodes within S for respectively the internal and external degree.

To work towards the definition of a community, we have to introduce some more variables. Intra-cluster density is defined as $\delta_{int}(S) = \frac{k_S^{int}}{n_s(n_s\text{-}1)/2}$. This basically means that we calculate a ratio between the number of edges within a subgraph and the total number of possible edges within S. A similar ratio can also be calculated

for the edges going outside of the subgraph. Inter-cluster density is defined as $\delta_{ext}(S) = \frac{k_{ext}^S}{n_s(n-n_s)}$.

For subgraph S to be considered a community, it has to hold for three constraints which are enumerated below. Basically, this aligns with the informal definition stated earlier this section.

1. First, $\delta_{int}(S)$ should be larger than the density $\delta(G)$ over the complete network. This indicates that the nodes within the subgraph have more edges within the subgraph, than expected based on the graph density.

2. Next, $\delta_{ext}(S)$ should be a lot smaller than $\delta(G)$. This indicates that the nodes within the subgraph have a lot less edges going outside the subgraph, than you would expect based on the graph density.

3. There should be a path between each node within subgraph S by edges from within the subgraph (connectedness).

A partition P is the division of a network into non-overlapping subgraphs. This means that each node can only be in one subgraph (or community; if the aforementioned constraints hold). One example partition is the singleton partition. In this partition, each node v is in its own subgraph S. To find a partition of communities for a given network, community detection algorithms can be used. These algorithms receive a network G as input and then calculate an output partition P which is a mapping of nodes and their community.

Chapter 3

Background & Related work

This chapter discusses the related work. Section 3.1 provides insights in several methods for community detection. Section 3.2 gives an overview of the metrics which are used to quantify the quality of the results.

3.1 Algorithms

First, we start with the different approaches for calculating a partition. Several categorisations of algorithms exist in literature.

In this thesis, we follow the categorisation proposed by Fortunato [4]. Other categorisations exist as well and can be based on the clustering notion they adopt [11]. The reason we choose the one by [4] is that it groups methods in terms of the methodological approach which is often used in the literature. Based on [4], we find the following categories and some example methods in Table 3.1.

Table 3.1: An overview of the different categories and examples of introduced methods.

Method category	**Methods**
Methods based on spectral properties	Spectral clustering [12]
Methods based on statistical inference	SBMs [13], Oslom [14]
Methods based on optimisation	Louvain [15], Leiden [5]
Methods based on dynamics	Walktrap, InfoMap [16, 17]
Methods based on edge removal	GN [18]
Other methods	LabelPropagation [19]

In this thesis, we will focus on methods based on optimisation and dynamics due to their performance on large networks and dominant adoption in the field of network science. As these are an integral part of our methodology, we discuss them in detail in Chapter 5. For the remaining methods, we give a short description. Because we will not use these methods in our proposed algorithm, we believe the formal aspects of these methods are out of scope for this thesis.

Methods based on spectral properties

Spectral clustering is an approach to detect communities using the spectral properties of the network [12]. An important benefit of spectral community detection is that it is simple to implement and based on linear algebra. The basic idea is to create a partition based on the eigenvectors. This approach has some drawbacks which are elaborated on by Nadler [20] and are also discussed by Von Luxburg [12]. Another drawback is that finding eigenvectors for large matrices is slow and memory-intensive. These are also the reasons that we will not focus on using spectral methods in this thesis.

Methods based on statistical inference

Another approach is statistical inference. The Stochastic Block Model (SBM) [13, 21] appears to be the most popular within this category. This works by fitting a generative network model. Stochastic blockmodels fall in the class of random network models and a lot of research has been done in this direction [22, 23]. While the ideas originate from decades ago, research is still done in this field. The most recent work can be seen in the overview work of Lee [24].

The idea is that we assume that the network is generated by a probabilistic model. This model has a block structure built into it, and the goal of using SBM community detection is to retrieve this. An important drawback of this type of approach is that we need to specify the number of communities beforehand. This number is usually unknown in real-world networks. Because this, our focus for this thesis will not be on methods based on statistical inference.

Other methods: LabelPropagation

Raghavan et al. [19] presented a method based on iterative propagation of labels throughout a network. The algorithm works as follows. The network is initialised as a singleton partition. This means that every node is in its own community. Next, the labels (representing the nodeś communities) are propagated throughout the network. This is done by iterating over all the nodes and choosing the community which occurs most frequent among the neighbours. In case of a tie, one node is selected at random from the most frequently occurring communities given an uniform distribution. The nodes are processed in a random order.

The expectation for a sensible stopping condition would be that the algorithm can be terminated if no labels are changing. However, due to the way ties are handled, it is possible that several nodes switch between iterations from the one to the other community. To tackle this problem, the algorithm runs until every node in the network has a label to which the maximum number of its neighbours belongs. Some improvements on the original LabelPropagation algorithm have been made over the years. For example the work of Fiscarelli [7], which introduces memory; MemLPA. While normal LabelPropagtion sometimes gets stuck in local optima, the goal of MemLPA is to prevent this. This is done by keeping a list containing a counter with labels for each node in memory. This list is updated after each iteration so assignments from earlier iterations are considered. Old assignments disappear by defining a maximum number of items in the list.

Methods based on optimisation

Nowadays, the most popular approach of doing community detection is by using methods based on optimisation. These methods try to maximize a function which indicates the quality of a partition. Methods on optimisation are always based on heuristics resulting in an approximation of the exact solution. The function to optimize which receives the most attention is modularity [18], introduced by Newman and Girvan. We elaborate on the quality function modularity in Subsection 3.2. As these approaches are part of our approach, we discuss modularity maximization methods in Subsection 5.1. Next to modularity, other quality functions exist like the Constant Potts Model (CPM) [25].

Methods based on dynamics

Communities can also be identified by using dynamics. Especially the idea of random walkers is often used in this category. The intuition is that if a nodes are tightly connected within the community, while there are only a few connections outside the group, the random walker would be trapped in a community for quite some time. Some popular methods are based on node similarity (such as WalkTap [26]), while others are a based on the so-called map equation. In this thesis, we focus on using the latter. Because these methods are part of our approach, we discuss them in detail in Subsection 5.1.

3.2 Quality metrics

Different quality metrics exist for quantifying the quality of a given partition. To explain the metrics relevant for this research, a distinction has to be made between community detection on real-world networks and community detection on artificial networks.

Artificial networks

Assessing the quality of a community detection method is often done with the Lancichinetti–Fortunato–Radicchi benchmark (LFR) [9] by Lancichinetti et al. With this benchmark, one generates several artificial networks with a priori knowledge of the communities within these networks. The algorithm uses several parameters for the degree distribution (γ) and community size distribution (β). It also has a mixing parameter (μ), which can be used to set the fraction of edges between communities. The actual comparison is often done with help of normalized mutual information (NMI), which compares the similarity of the artificial and discovered partitions. For this thesis however, we will focus on real-world networks, because these give a better intuition on actual applications where no information is a-priori available about the community structure. Because we cannot use this information, community detection on real-world networks requires other quality metrics than by using the LFR benchmark.

Metrics for real-world networks

In this subsection, we follow a categorisation often used in literature. According to Coscia [27], the landscape of quality metrics can be classified according to process, definition and performance. In their paper they introduce similarity as additional category. However, in this thesis, we follow the categorisation by Leskovec et al [28] because this grouping is quite popular in literature. They divide these metrics into two categories: single criterion scores and multi-criterion scores. It is discussed that there are two important criteria when thinking about the quality of a community. First, the number of edges inside a community. Second, the number of edges between the nodes in the community and the rest of the network. Multi-criterion scores combine both criteria while single criterion scores only use one of the two. Subsequent to these scores, which provide a single value, we will dive into another approach. In Section 3.2, we look into the distribution of different community sizes within a partition.

Single criterion scores

In this subsection we list the single criterion scores. We start with an overview of quality metrics to show that many different quality metrics exist. This supports one of the problems stated in the introduction of this thesis. After that, we dive into modularity as this is the most widely used metric in literature. It must be noted that many of these metrics are defined on S and not on the resulting partition P. These subgraph quality metrics can be used to get a feeling about the quality of the partition. For example by taking the average over all the subgraph scores which we will do in rest of this thesis when not stated otherwise. A lower $f(S)$ generally means a better community, except for the later introduced modularity.

- **Modularity ratio:** $\frac{m_S}{\mathbb{E}(m_S)}$. This metric calculates the number of edges between nodes in the community, compared to the expected number of these edges ($E(m_s)$) within the subgraph.
- **Volume:** This can be calculated by summing the degree $deg(v)$ for all the nodes within S. Formally, we can define volume as $\sum_{v \in S} deg(v)$.
- **Edges cut:** Defines how many edges need to be removed, to completely disconnect the subgraph S from the network. Denoted as c_S. Again, lower is better.

Modularity The concept of modularity [18], introduced by Newman and Girvan, is one of the most popular metrics to evaluate the quality of a partition. The value of modularity lies between -1 and 1 [29]. This value is positive when the observed connections within the same community exceed the expected number under random connections. Optimizing modularity is NP-Hard [30]. Different types of algorithms can be used for optimising modularity such as simulated annealing [31]. In practice, the most popular algorithm in this category is the Louvain algorithm (discussed in Section 5.1).

Equation 3.1 gives the formula for calculating modularity.

$$Q = \frac{1}{2m} \sum_{ij} \left[w_{ij} - \frac{d_i d_j}{2m} \right] \delta(c_i, c_j) \tag{3.1}$$

The weighted degree of node i is $d_i = \sum_j w_{ij}$. We consider w_{ij} as an element of adjacency matrix A. Under uniform random selection, the expected number of edges between two nodes is $\frac{d_i dj}{2m}$. Basically, modularity calculates the difference between the observed and the expected $(w_{ij} - \frac{d_i d_j}{2m})$. Here, $\delta(c_i, cj)$ is the Kronecker delta function which resolves to 1 if both values are in the same community. The value resolves to 0 if they are not in the same community.

Although modularity is a widely used function, it does have some drawbacks. Fortunato [8] shows that the resolution limit is a problem which occurs with modularity. This basically states that that the optimisation of modularity is blind to communities whose size is smaller than $\sqrt{2m}$. This means that smaller communities can remain undetected when working with networks at a larger scale. Figure 3.1 shows an example of the resolution limit. The network consists of a ring of cliques, each with four nodes. We would expect that each clique will be seen as a separate community. However, due to the resolution limit, multiple cliques combined into a single community, result in a higher modularity score (see dashed lines). Based on the notion that modularity is blind for communities smaller than $\sqrt{2m}$, we find that increasing the size of the ring by adding more cliques, also increases the community size. This means that multiple cliques are combined into one community. The result is that the final partition does not necessarily capture the real community structure of the network.

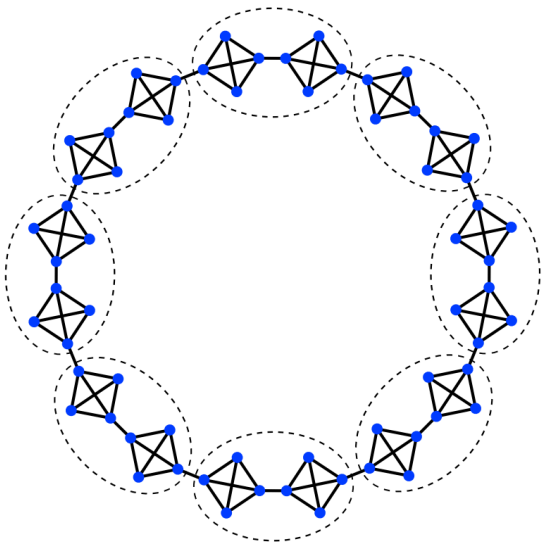


Figure 3.1: Standard resolution limit example of a ring with cliques of four nodes from [4]

Arenas et al. [32] proposed a solution to this resolution limit by adapting the function to optimize. Their method allows multiple resolution screening by providing each edge with a self-loop of the same magnitude r. This way, the connectivity of the network remains the same while it allows search at different scales. We use n_s as the number of nodes in the subgraph, while n and m represent respectively the number of nodes and edges in the complete network. Further, we use k_s^{in} as the internal degree and k_s as the total degree within the subgraph.

$$Q^{AFG}(r) = \sum_s \left[\frac{k_s^{in} + n_s r}{2n + nr} - \left(\frac{k_s + n_s r}{2m + nr}\right)^2\right] \tag{3.2}$$

Another possible solution is created by Reichardt and Bornholdt [33]. Their approach is to tune the null model with a parameter γ as shown in Equation 3.3.

$$Q^{RB}(\gamma) = \sum_s \frac{k_s^{in}}{2m} - \gamma\left[\frac{k_s}{2m}\right]^2 \tag{3.3}$$

Lancichinetti [34] shows that multiresolution modularity maximization is characterized by two things: the tendency to merge small communities and to split large ones. He shows that it is very difficult to tune the resolution such to avoid both biases simultaneously. We can conclude from this, that although solutions do exist, it is hard to actually use them in practice.

Multi-criterion scores

This subsection gives an overview of multi-criterion scores. These score combines both the number of edges in the community and the number of edges to nodes outside the community. Again, we provide this list to give a sense of the possible approaches for quality metrics in community detection. We do this to support the earlier mentioned fact that there are a lot of different quality metrics.

- **Expansion:** $\frac{c_S}{n_S}$ The ratio between the total of edges leaving community and n_S as the number of nodes within the subgraph.
- **Internal density:** $1 - \frac{m_S}{n_S(n_S-1))}$ The internal density of the community.
- **Cut ratio:** $f(S) = \frac{c_S}{n_S(n-n_S)}$ The part of the edges leaving the community compared to all possible edges within the community. This is denoted by ($n_S(n - n_S$).
- **Maximum-ODF:** The maximum fraction of edges of a node pointing outside the community within the complete community. Maximum-ODF again uses S as an input.
- **Average-ODF:** The average number of edges for a node that point outside the community.
- **Flake-ODF:** The number of nodes in the subgraph that have fewer edges inside, than outside the community.

Conductance Conductance [35, 36], can be considered one of the simplest notions of community quality. It can be thought of as the ratio between the number of edges inside the community and the number of edges leaving the community. Based on the equation $f(S) = \frac{c_s}{(2)k_{int}^s + c_s}$, we show that a lower conductance score is better. The number of edges pointing outside the community, or simply the number of edges cut, is denoted as c_s. The (2) in the denominator is added for undirected networks and can be discarded for directed networks. The conductance of a network G is the minimum conductance over all the communities S. Conductance has some interesting extensions like the network community profile plot (NCP) proposed by Leskovec [37]. This extension applies conductance over a range of size scales of communities.

Community size distribution

Research has shown that the same modularity value can give completely different results in terms of community assignment. We have also seen that, although we reach a good score in our chosen metric, it is possible that the final partition does not provide valuable results due to problems like the resolution limit. To get a better understanding, Dao [38] proposed a method to compare community size distributions. He shows that we can compute some high-level measures, for example: minimum community size, maximum community size, mean and median about the community size distribution.

Similar information can be visualized in a joyplot (or ridgeline plot). These plots show the distribution of community sizes for different solutions. The results on the vertical axis can consist of community detection algorithms or can different type of parameters which eventually result in different partitions. Figure 3.2 gives an example joyplot of the partitions when applied on the LES MISERABLES network [39]. This networks models the co-occurances of characters. The distributions are smoothed for visualization purposes, otherwise the figure would be very hard to read. This figure gives a lot more insights compared to the simple metrics defined earlier. For example the similarity in community size distribution between the Leiden algorithm and the Louvain algorithm (both algorithms will be introduced in Section 5.1). This can be explained because both methods contain similar components and optimize the same objective function. Initial results show that methods which are not based on optimizing modularity, generally prefer a smaller community size (although this difference is only small). This can be explained from the fact that optimizing for modularity becomes blind for specific sized communities when networks reach a certain size (the aforementioned resolution limit).

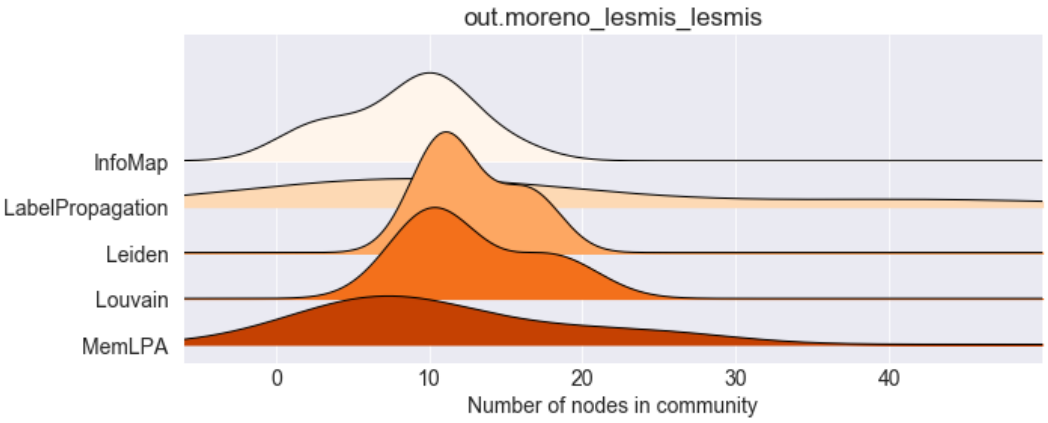


Figure 3.2: Joyplot with the community size distribution for different community detection algorithms on the LES MISERABLES network [39]. The different colors are used for aesthetics and to used the improve the difference between algorithms.

Chapter 4

Data sets

In this chapter, we introduce the data sets used for this thesis. To be able to verify the workings of our proposed method, we use a wide variety of networks. As stated in Section 3.2, we focus on real-world networks. Many of these networks are made available by the KONECT [40] network repository. We follow the categorisation from this repository, as this gives us a brief overview of the large range of networks we have tested. A summarization of the categories is displayed in Table 4.1. Table 4.2 lists all data sets.

Figure 4.1 gives an overview of the data sets and their nodes and edge counts. We observe that some categories, like HumanContact, hold overall smaller networks while social networks tend to be bigger. Generally we see an upward trend in the number of edges when the number of nodes is increased, which is expected. Next, most of the categories appear to be useful, as they group together similar networks.

To get a better understanding of the different data sets, we make a distinction based on network properties. Figure 4.2 shows a smoothed version of the clustering coefficient for the different categories. We see a clear difference for each of the different categories. The lines as the bottom represent individual measurements. For the infrastructure networks, we find a lower clustering coefficient. The large peak indicates that a lot of networks within this category show similar results. As one would expect, the Misc category contains a wide range of different networks. Therefore, it is hard to find a clear distribution for the clustering coefficient. From this network, we can conclude that the categories provided by KONECT are actually quite valuable and related to higher order network properties.

Table 4.1: An overview of the different type of networks categories.

Category	**Description**
Animal	*Animal networks consist of interactions between animals.*
Authorship	*Authors who have written work together.*
Coauthorship	*Unipartite network connecting authors who have written works together*
Communication	*Includes networks with messages between persons, like mail*
Computer	*Actual computer networks, nodes are computers and edges are connections*
HumanContact	*Networks of real contact between persons,Often this is collected by giving RFID tags to people*
HumanSocial	*Real-world social network between humans. In contrast to HumanContact, edges represent a state*
Infrastructure	*Actual infrastructure. For example, roads and power grids.*
Lexical	*Made of words from natural languages and the relations between them.*
Metabolic	*Model the set of processes that determine the properties of a cell.*
Misc	*Contains a wide variety of networks such as actor collaboration, co-purchasing networks and more*
OnlineContact	*Consist of people and interactions between them. Contact networks are unipartite*
Social	*Represents connections in online social networks like Facebook or Twitter.*

Figure 4.1: Overview of the data sets, with each color/marker representing the category. The horizontal axis represent the number of nodes while the vertical axis represents the number of edges.

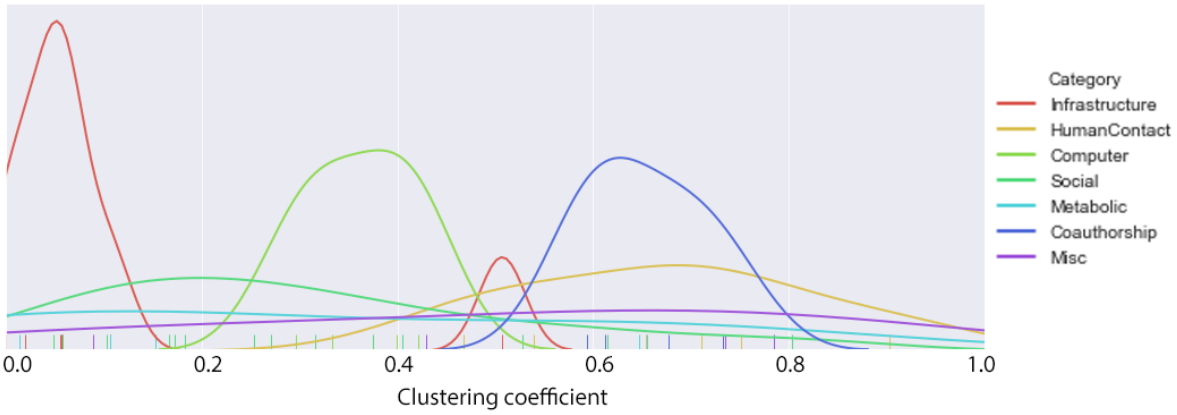


Figure 4.2: Overview of the different categories and their distribution of clustering coefficient. The horizontal axis represents the clustering coefficient.

Table 4.2: Overview of the data sets

Data set	Average degree	Clustering coefficient	Density	Number of edges	Number of nodes	Category
out.tntp-ChicagoRegional	1.769598	0.000000	0.001207	1298	1467	Infrastructure
out.sociopatterns-infectious	13.487805	0.467220	0.032978	2765	410	HumanContact
out.as-caida20071105	4.032559	0.333351	0.000152	53381	26475	Computer
out.arenas-email	9.622242	0.254032	0.008500	5451	1133	Communication
out.youtube-u-growth	5.816738	0.168090	0.000002	9375374	3223585	Social
out.ucidata-gama	7.250000	0.539187	0.483333	58	16	HumanSocial
out.as20000102	4.292555	0.399239	0.000663	13895	6474	Computer
out.com-youtube	5.265046	0.172258	0.000005	2987624	1134890	Social
out.arenas-meta	9.006623	0.655140	0.019926	2040	453	Metabolic
out.wordnet-words	8.999678	0.655310	0.000062	656999	146005	Lexical
out.dblp_coauthor	8.078133	0.735391	0.000006	5179945	1282461	Authorship
out.petster-carnivore	50.337069	0.527765	0.000081	15699276	623766	Social
out.loc-brightkite_edges	7.353095	0.270675	0.000126	214078	58228	Social
out.maayan-vidal	4.293648	0.106378	0.001371	6726	3133	Metabolic
out.com-dblp	6.622089	0.732135	0.000021	1049866	317080	Coauthorship
out.livejournal-links	18.898082	0.344557	0.000004	49174464	5204175	Social
out.contiguous-usa	4.367347	0.507044	0.090986	107	49	Infrastructure
out.livemocha	42.132945	0.058236	0.000405	2193083	104103	Social
out.moreno_names_names	10.300056	0.720819	0.005813	9131	1773	Lexical
out.arenas-pgp	4.553558	0.440288	0.000426	24316	10680	OnlineContact
out.flixster	6.276330	0.205821	0.000002	7918801	2523386	Social
out.ca-cit-HepPh	224.144591	0.593976	0.007979	3148447	28093	Coauthorship
out.facebook-wosn-links	25.640112	0.253149	0.000402	817035	63731	Social
out.adjnoun_adjacency_adjacency	7.589286	0.189785	0.068372	425	112	Lexical
out.douban	4.223952	0.048037	0.000027	327162	154908	Social
out.wikiconflict	34.713119	0.477703	0.000297	2027871	116836	OnlineContact
out.hyves	3.960180	0.103493	0.000003	2777419	1402673	Social
out.ca-AstroPh	21.101699	0.676854	0.001124	198050	18771	Coauthorship
out.ego-facebook	2.064404	0.802965	0.000715	2981	2888	Social
out.sociopatterns-hypertext	38.867257	0.539530	0.347029	2196	113	HumanContact
out.actor-collaboration	78.688307	0.784772	0.000206	15038083	382219	Misc
out.petster-friendships-cat-uniq	72.802605	0.405529	0.000486	5449275	149700	Social
out.petster-friendships-dog-uniq	40.047706	0.182532	0.000094	8546581	426820	Social
out.moreno_kangaroo_kangaroo	10.705882	0.873564	0.669118	91	17	Animal
out.moreno_lesmis_lesmis	6.597403	0.735525	0.086808	254	77	Misc
out.roadNet-PA	2.834132	0.056204	0.000003	1541898	1088092	Infrastructure
out.mit	52.895833	0.751503	0.556798	2539	96	HumanContact
out.com-amazon	5.529855	0.429740	0.000017	925872	334863	Misc
out.ucidata-zachary	4.588235	0.587931	0.139037	78	34	HumanSocial
out.moreno_train_train	7.593750	0.711229	0.120536	243	64	HumanContact
out.petster-hamster	13.710635	0.614647	0.005654	16631	2426	Social
out.moreno_propro_propro	2.435294	0.152989	0.001303	2277	1870	Metabolic
out.subelj_euroroad_euroroad	2.413969	0.019962	0.002058	1417	1174	Infrastructure
out.as-skitter	13.080877	0.296292	0.000008	11095298	1696415	Computer
out.flickr-links	18.132883	0.374615	0.000011	15551250	1715254	Social
out.flickrEdges	43.741585	0.089392	0.000413	2316948	105938	Misc
out.arenas-jazz	27.696970	0.633447	0.140594	2742	198	HumanSocial
out.petster-friendships-hamster-uniq	13.491927	0.167109	0.007265	12534	1858	Social
out.roadNet-CA	2.815590	0.055424	0.000001	2766607	1965206	Infrastructure
out.dolphins	5.129032	0.302932	0.084082	159	62	Animal
out.loc-gowalla_edges	9.668062	0.316284	0.000049	950327	196591	Social
out.ca-cit-HepTh	213.444910	0.611741	0.009318	2444798	22908	Coauthorship
out.moreno_zebra_zebra	8.222222	0.875914	0.316239	111	27	Animal
out.moreno_beach_beach	15.627907	0.653434	0.372093	336	43	HumanContact
out.topology	6.197750	0.421236	0.000178	107720	34761	Computer
out.maayan-pdzbase	2.301887	0.013268	0.010909	244	212	Metabolic
out.contact	15.503650	0.902887	0.056790	2124	274	HumanContact
out.roadNet-TX	2.785182	0.057460	0.000002	1921660	1379917	Infrastructure
email-Enron.txt	10.020222	0.715642	0.000273	183831	36692	Communication
out.opsahl-powergrid	2.669095	0.106539	0.000540	6594	4941	Infrastructure
out.dnc-corecipient	23.022075	0.791403	0.025439	10429	906	OnlineContact
out.reactome	46.640430	0.646518	0.007373	147547	6327	Metabolic

Chapter 5

Approach

In this chapter, we discuss the approach of our proposed algorithm to combine community detection methods in such a way, that performance can be improved on commonly used quality metrics. This includes the relevant existing algorithms on which we build in Section 5.1, ways to aggregate partions in Section 5.2 and our proposed ensemble community detection algorithm in Section 5.3.

5.1 Algorithms

In this section we introduce the algorithms used in our approach.

Louvain

Louvain, named to the affiliation of the creators, is an algorithm by Blondel et al. [15] based on the idea of greedy modularity optimisation. The literature shows that this algorithm is very popular, mainly because of the good performance in terms of modularity combined with the execution time of the algorithm.

Algorithm The functionality of the Louvain algorithm is as listed below. Figure 5.1 gives a visual explanation of how the algorithm works. The figure shows the different steps which are applied inside a single iteration.

1. Start with a singleton partition by assigning each node to its own community.
2. For each node, we try to realise a modularity improvement. We do this by observing the adjacent nodes and their communities. For each combination of the selected node and the neighbouring node, calculate the resulting modularity when the community of the selected node is merged into the neighbouring node. The node combination which results in the largest modularity improvement is merged. If there is no modularity improvement possible, leave the node in the same community. This step is continued until no changes in community assignment occur.
3. Create a new network by aggregating the communities to nodes and create edges when communities are connected. The weights of the edges represent the number of connections between the communities. This is nicely visualized in Figure 5.1 and is called the community aggregation step.
4. Again, iterate over all the nodes of the aggregated network and try to optimize modularity by merging communities.
5. Transfer the communities from the aggregated network back to the original network. Go back to step 2 until no further improvements in modularity possible.

The algorithm outputs one partition per iteration of step 2. Aynaud et al. [41] state that both the average size of the communities and the modularity increase from one iteration to another. This happens by definition. If the size of communities does not increase, it means that the algorithm is finished.

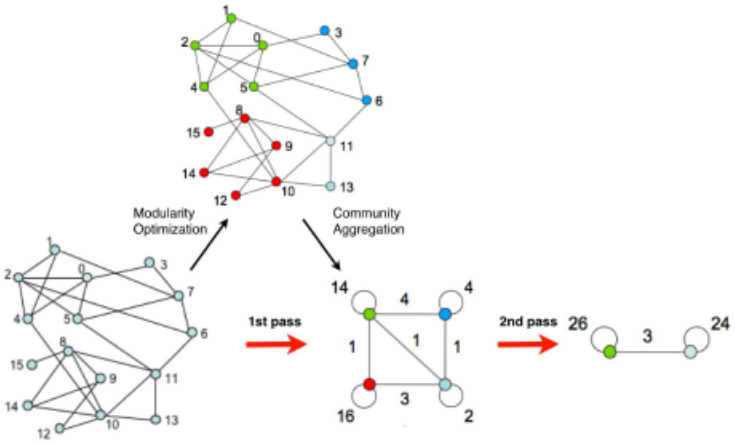


Figure 5.1: Louvain algorithm step-by-step from [15].

Advantages The Louvain algorithm has several advantages compared to non-optimisation based methods. The algorithm is known to run in $\mathcal{O}(n \log n)$ [15]. Overall, it gives quite good results within a reasonable amount of time. The algorithms also incorporated a resolution parameter in their modularity function to handle communities at multiple scales.

Drawbacks The Louvain algorithm has some parts which are not deterministic. First, the point where two neigbouring nodes offer the same modularity increase, and second the order in the nodes are processed. It has to be noted that in most cases, the difference between the best and the worst modularity obtained over several experiments only amounts to a few percent. Another problem is that the result can contain communities which contain disconnected nodes.

Leiden

The Leiden algorithm [5] by Traag et al. can be considered an improved version of the Louvain algorithm. It has some additional benefits where the nodes within a community are guaranteed to be connected. It also has some improvements which help the algorithm to run faster than the Louvain algorithm, while providing better modularity scores.

Algorithm In contrast to the Louvain algorithm, an iteration of the Leiden algorithm consists of five steps: local moving nodes, refinement of the partition, aggregation of the network, local move of the aggregation network, refinement of the aggregation network. Again, we show the functionality both visual (Figure 5.2) and by using an enumeration.

1. Start with a singleton partition, like we did with Louvain.
2. In the first step, the algorithms moves nodes from one community to another if the modularity value increases. Basically this is similar as with Louvain, with some small improvements which speed up this step. The authors call this the local moving phase.
3. The next step is the refinement phase. In this phase, we start with a refined partition which is a singleton partition. Next, apply local moving of nodes, but only within each community of the partition resulting from step 1. Next to that, they can only be merged if sufficiently well connected to their community in the partition from the previous step. This phase also contains some randomness, as nodes are note always labelled with the community which yields the largest modularity increase. This selection is done randomly from the collection of neighbours given a certain probability. The larger the modularity increase,

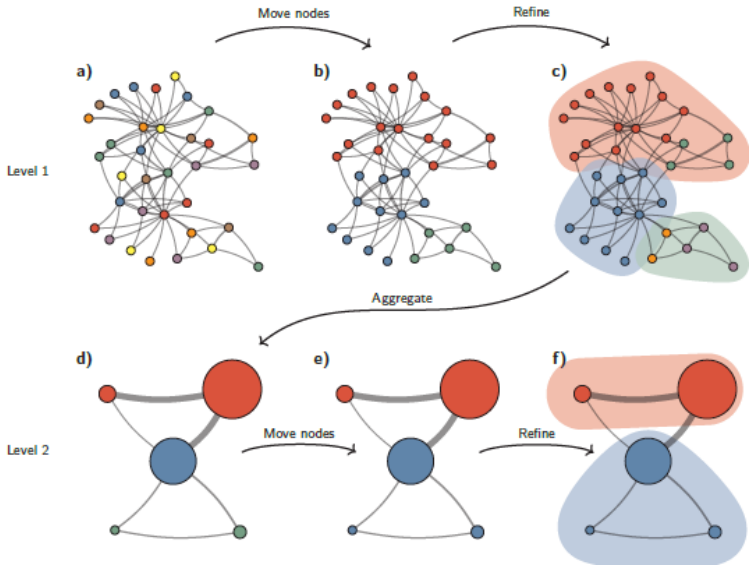


Figure 5.2: Leiden algorithm step-by-step from [5].

the higher the probability that a community is selected. Due to the fact that communities can only be assigned within the subgraph from the previous step, this step makes sure that the number of communities does not increase.

4. In this phase, we create an aggregated network based on the refined partition. This means that we create a network with the communities as node and edges when the nodes from two different communities have an edge between them.

5. Finally, the whole algorithm is simply repeated on the aggregated network, without starting from a singleton partition, but from the partition that was previously found.

Advantages The Leiden algorithm has several advantages over the Louvain algorithm. The algorithm has several guarantees. For example, the connectedness of communities. With Louvain, it is possible that a community exists of nodes that are not connected. This cannot happen with the Leiden algorithm. Second, the improved performance in terms of run time. In the local moving phase, the Leiden algorithm only reconsiders the nodes that have changed.

Drawbacks Although the Leiden algorithm has several advantages over the Louvain algorithm, it still has some drawbacks. Most of these drawbacks are inherited from the base method. Just like the Louvain algorithm, it is sensitive to the resolution limit because of the function they both optimize (modularity). It also contains a random component which results in possibly different results when running the algorithm multiple times.

InfoMap

InfoMap is a community detection method introduced by Rosvall [16, 17] based on the Map Equation. It uses the idea of a random walker through a network. Therefore, we can say that InfoMap has a focus on the flow (or dynamics) inside the network which is quite different from the other approaches.

Algorithm InfoMap tries to apply principles from minimum description length (MDL) statistics [42] to find structures within a network. The idea is that by using a random walker, the walker will visit nodes which are

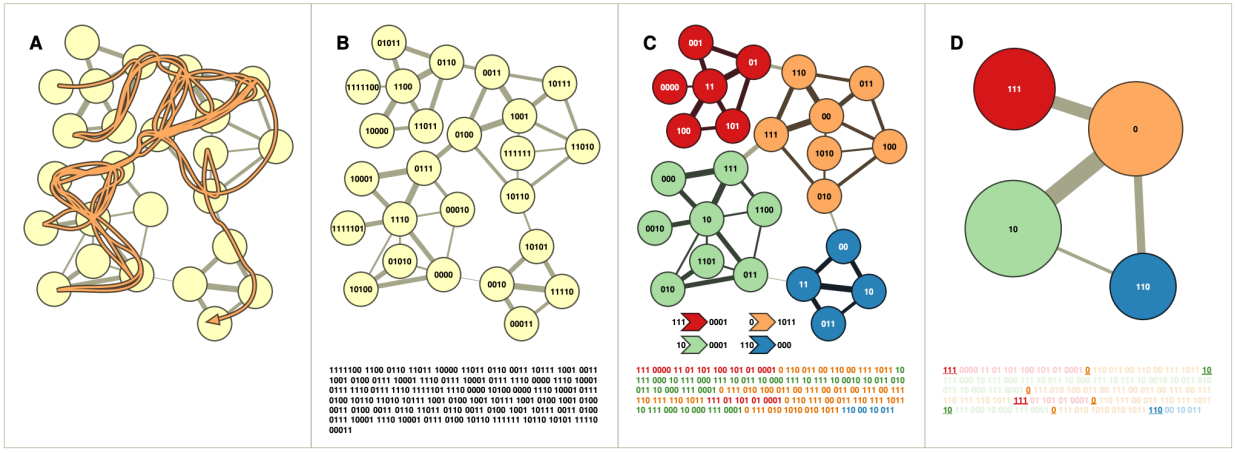


Figure 5.3: Infomap algorithm step-by-step from [16].

tightly connected more often.

Figure 5.3 gives an overview of the algorithm. Below we enumerate the steps executed by this algorithm.

1. The algorithm with random walkers traverses through the network. As the walker progresses through the network, the idea is that it stays within parts which are more tightly connected. The visualization of this algorithm (Figure 5.3) shows this by the thicker lines.
2. In the second step, we create the so-called codewords of the Huffmann Codebook [43]. Huffman code is used to compress information and here, we use this to apply a code to each node.
3. InfoMap uses multiple codebooks together with an index codebook. Different codebooks correspond to different communities. Communities receive unique names but the names of nodes within communities are reused.
4. Finally, reporting only the module names, and not the locations within the modules, provides an optimal coarse-graining of the network. In practice of course, the mapping from step 3 is used for the actual partition.

Advantages One of the advantages compared to Leiden or Louvain, is that InfoMap is not as vulnerable to the resolution limit [44]. This is because it does not optimize modularity. InfoMap has good results on the LFR benchmark and even outperforms Louvain for some networks [2].

Drawbacks One of the advantages of Leiden and Louvain is that they are relatively fast algorithms. Unfortunately, InfoMap is a bit slower as shown by Lancichinetti et al. This can become a problem at scale [9]. Yang [2] also shows that InfoMap becomes unreliable for larger values of μ (the mixing parameter) when running the LFR-benchmark.

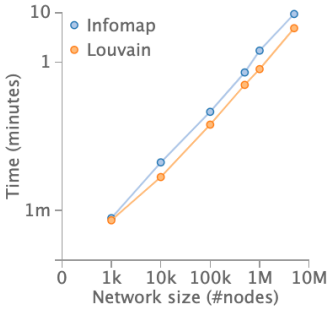


Figure 5.4: Difference between InfoMap and Louvain in terms of speed , image from [45]. The number of nodes in a network is on the horizontal axis and the run time on the vertical axis.

5.2 Aggregation

This section is about combining multiple partitions into one. We discuss are consensus clustering and its extension fast consensus clustering.

Consensus clustering

Consensus clustering [46], as proposed by Lancichinetti and Fortunato, is a method to generate stable results out of a set of partitions. Most community detection methods are not deterministic and can provide different results for multiple runs. Consensus clustering can be used to provide stability and robust results. One of the main advantages is that it can be used with all types of community detection algorithms, as long as they support weighted networks. It is important to note, that the goal is not to find the optimal partition, but to find the most stable partition.

The algorithm has two parameters. Parameter r is the number of partitions that need to be combined and the threshold parameter τ is added to have some flexibility regarding the speed of the algorithm.

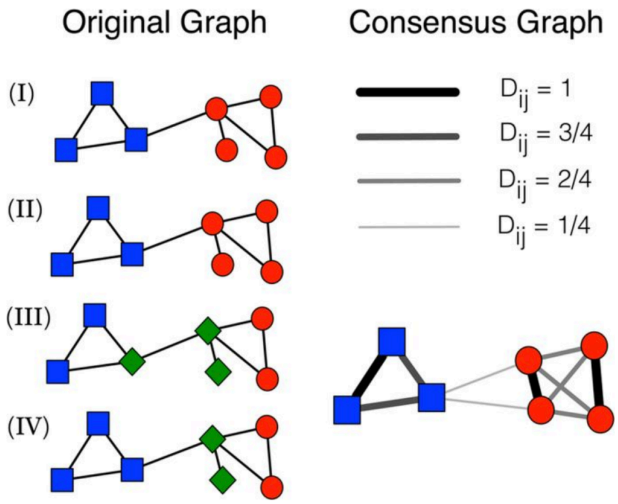


Figure 5.5: Resulting partition from consensus clustering on a network from [46]

Figure 5.5 shows a network with four partitions. Some of them have two communities (solution 1 and 2), while others (solution 3 and 4) have three of them. Each iteration, the edge weights are updated based on whether nodes are in the some community in the previous iteration. In the figure, this is shown by the thickness of the edges. The algorithm uses these weights as input for a new run of the community detection. This process continues until all partitions are equal.

Of course, consensus clustering has its drawbacks. Because we combine results of many partitions, the consensus matrix becomes a dense matrix very quickly. For computational reason, everything below threshold τ is discarded. Unfortunately, by discarding certain edges below the threshold, it is possible that some nodes become disconnected from the rest of the network. This is resolved by connecting them to the neighbour with the highest weight. Another drawback of this dense matrix is that it becomes slow for larger networks resulting in a time and complexity of $\mathcal{O}(n^2)$. Even though it has quite some drawbacks, the advantages should not be forgotten. Consensus clustering supports a range of different algorithms and is still is a useful method for combining partitions.

Fast consensus clustering

Fast consensus clustering is a faster technique to calculate consensus partitions. It can be considered an improvement to normal consensus clustering. One important problem with the original algorithm is the calculation of the consensus matrix, which is used to determine how often a certain edge is in the same community. The calculating of this matrix can become slow for larger networks.

To solve this problem, Tandon et al. [47] introduce fast consensus clustering. With fast consensus clustering, the consensus matrix is only computed for a subset of all node pairs. The algorithm has several parameters, a threshold parameter (τ) and a parameter for the stopping condition (δ). This parameter influences the convergence. The algorithm stops when the fraction δ of the edges has a weight not equal to 1. Instead of calculating for all node pairs, we only calculate the weights for the nodes which are actually connected. The main difference is that we compute up to $2m$ elements of the consensus matrix, which becomes useful when the network is very sparse. The advantage is that the calculation of consensus matrix can reach space and time complexity $\mathcal{O}(m)$, which is a lot faster than traditional consensus clustering.

Using a subset to reduce the necessary calculations, also has its drawbacks. The resulting partitions are more noisy and are even unlikely to converge at all. This is no problem, because the algorithm has a different stopping condition compared to the original consensus clustering. Fast consensus clustering stops when enough entries have are different from 0 and 1.

5.3 Ensemble community detection

Recall from Section 1 that the goal of our method of ensemble community detection is to combine multiple community detection algorithms to improve the performance compared to a single community detection algorithm.

Our algorithm combines existing algorithms and fast consensus clustering. It can be considered a modification to the procedure mentioned in Section 5.2. The most important novelty lies in the step of deriving the input partitions. Instead of applying the same community detection algorithm multiple times, we work with the results from different algorithms. Our algorithms of choice in this thesis are InfoMap and Louvain. The approach is modular, one can swap either of these for a different one if desired, as long as the algorithm supports weighted networks.

The algorithm has several parameters. First, the delta (δ) which is used in the stopping criterion. Next, the threshold parameter (τ). The algorithm runs every ratio between two community detection algorithms restricted by the maximum number of algorithms at the same time (n_p). We show this ratio by using a percentage Louvain compared to a percentage InfoMap (i.e. 75%/25%). The algorithms of choice can be considered an

input as well. The resulting partitions are interpreted as explained in Section 6.1. The final partition is chosen by looking at predefined quality metrics in combination with the community size distribution.

Figure 5.6 shows how our algorithm operates. Below, we explain how to get the resulting partition for a single ratio, in our example 75%/25% with $n_p = 4$.

1. First, we start by running the the community detection algorithms. In our example, we run Louvain three times and InfoMap once. The input network is the same for both algorithms.
2. Second, we use the resulting partitions as input and we update the weights in the same way as with fast consensus clustering. This is explained in Section 5.2 and visualized in Figure 5.6. Basically, the resulting weight of an edge is higher, if both nodes from an edge are in the same community when considering multiple input partitions. This step outputs a network with updated weights.
3. This output network is then used as input network. The community detection algorithms are run again like in step 1, in the same ratio, but with the new network. The algorithm stops when the fraction δ of the edges for the consensus clustering step has a weight not equal to 0 or 1. We finally return the resulting partition.

Pareto fronts

Several solutions exist when working with multiple quality metrics. To visualize multiple quality metrics at the same time, we propose that pareto fronts can be used. Pareto fronts are widely used in economics and in the field of multi-objective optimisation [48]. Figure 5.7 gives an example visualization of a pareto front. In this example, both modularity and conductance should be minimized. The vertical axis is inverted to easily find good solutions. The points at the left bottom corner are perceived better than points to the right and on top of in the figure. The point (7/3) can be considered pareto efficent, which basically means that neither objective function can be improvement without decreasing the other.

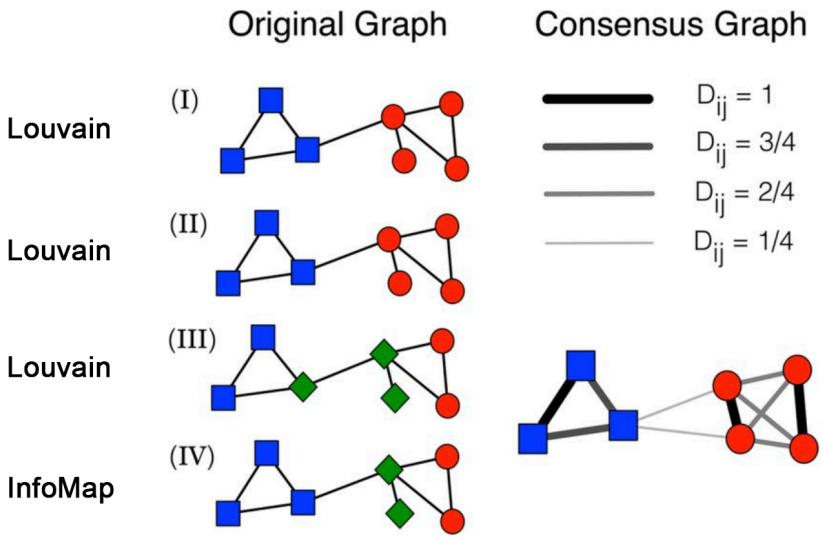


Figure 5.6: An example of ensemble community detection. We see that the input partitions from different algorithms can be used. For this particular example, a ratio of 75% Louvain / 25% is used with $n_p = 4$

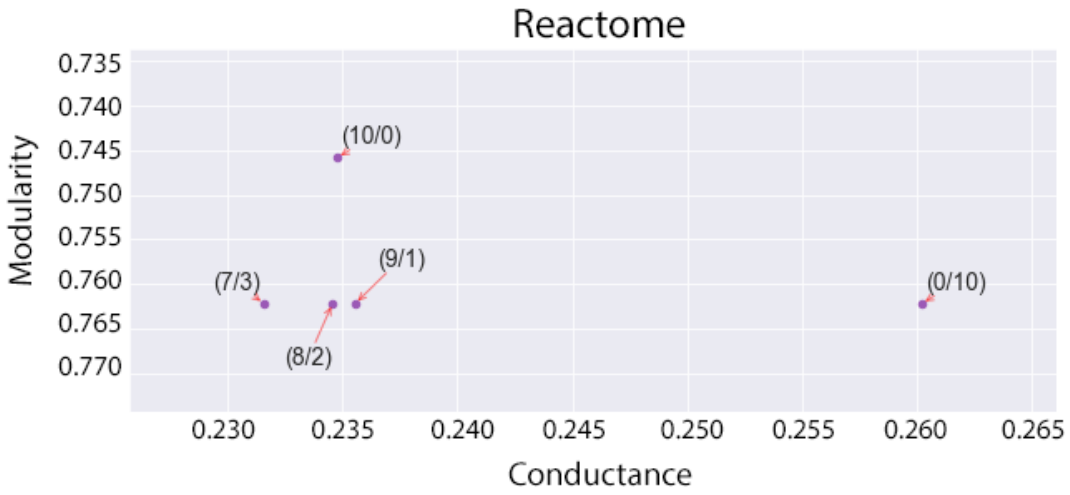


Figure 5.7: An example visualization of a pareto front for the REACTOME [49] biological network.

Chapter 6

Results

In this chapter, we discuss the results of our experiments. We start with our experimental setup in Section 6.1. Next, we try to pinpoint when our algorithms performs better or worse by assessing the relation between network properties and performance in Section 6.2. Later, we try to solve the problem regarding the range of different options for quality metrics in Section 6.3. We assess the suitability of pareto fronts together with community size distribution of the data sets.

6.1 Experimental setup

The experiments have been executed on a machine provided by the LIACS Data Science Lab. The machine contains 1TB of RAM and 64 cores of Intel Xeon E5-4667v3 CPUs. This enabled us to run a lot of experiments in parallel. The source code is written in Python and we use the popular `igraph` package. For fast consensus clustering, we used the code provided by the authors [47].

In terms of parameters for consensus clustering, we modified the parameters for the clustering algorithm slightly to improve convergence speed. We tried to stay close to the original paper, because the scope of this thesis is not about optimizing fast consensus clustering parameter. For the threshold parameter, we used $\tau = 0.35$. For the convergence threshold, we will use the value $\delta = 0.1$ from the original paper. The paper states that results are stable for thresholds at least up to 10%. We test all the possible ratios given $n_p = 10$.

To compare our ensemble method with Louvain and InfoMap, we compare our results versus the baseline. This baseline is chosen by picking the partition leading to the best result for modularity between 100% InfoMap or 100% Louvain. This makes sense, because this would be the algorithms we would have used if we did not have used our proposed ensemble approach. Although we would expect that Louvain performs better given the fact that it optimizes modularity, initial research has shown that this is not always the case. This is why we can use this as baseline. As introduced before, we look at improvements of conductance, modularity, expansion and the internal density. To make sure that the community size do not change a lot, we have to keep an eye on that as well. To provide an overview of the results, we included these results in percent change compared to the baseline in the table.

As the Louvain algorithm returns a dendogram of partitions at different scales, we pick the partition on the lowest level possible. This means that the input partition for the consensus clustering has the most communities available compared to the other options available with the Louvain algorithm. We are aware that partitions on the higher levels yield better modularity, but initial experiments have shown that using these partitions result in problems with convergence of the algorithm.

A lot of community detection algorithms contain random components. This includes the algorithms used in our experiments: InfoMap and Louvain. This means that multiple runs can lead to different results. Fortu-

nato [4] already mentioned the stabilising, as in robust to randomness, effects of consensus clustering. While this is true for the experiments from [4], we are not sure if the same properties apply for our ensemble community detection. To test this, we will run the experiments multiple ($n = 5$) times, to verify if the same properties hold. The results shown in this thesis are the average over all these runs.

Assessing quality of community detection results

Using a combination of both community size distribution (with a joyplot) and other metrics provides us with a solid overview of the resulting partitions. None of the approaches by itself is sufficient to fully grasp and determine the quality of a given partitions. This is especially the case when working with large networks where visualisations of the networks become hard to interpret. Our experiments try to find two things.

First, we look at some basic network properties. For now, the properties we consider are average degree, density and the clustering coefficient. Because we are trying to pinpoint how algorithm results are affected by these network properties, we look at the pearson correlations between these properties and the quality metrics discussed below. This way, it should be clear on what kind of networks the proposed algorithm provides good results.

In Section 3.2, we discussed several ways to measure the quality of a partition (or subgraph). No single method fits all the aspects of a partition quality. To tackle this problem, we will have to look at multiple metrics. For now, we will look at the following metrics: modularity (maximise), conductance (minimise), expansion (minimize), internal density (maximise) and the community size distribution. Essentially, we are evaluating the results on multiple quality metrics at the same time. To get a better understanding of how to asses the quality of multiple metrics, we use pareto frontiers. These pareto fronts show the trade-offs between certain metrics and allow us to make choices between the different algorithm settings. We combine these Pareto fronts with the community size distribution to make an actual assessment of the resulting partitions.

6.2 Network properties and their effect on ensemble community detection

This section is about how network properties affect the results of our ensemble community detection algorithm.

Table 6.1 gives an overview of the resulting metrics on different data sets. Each column is this table is the percentage difference compared to the baseline (discussed in 6.1). Based on these results, we find that the proposed algorithm gives better results for some networks, but not all. Note that given how these metrics are defined, the goal is that modularity increases, while conductance should decrease. The table directly shows that a reduction of modularity does not mean a reduction of conductance. Given the results from Table 6.1, we can see that our method results in better conductance scores than it does on modularity.

Figure 6.1 shows the pearson correlations between the network properties and the resulting quality metrics. This figure is based on the data provided in Table 6.1. We compared the solutions against a baseline solution (best modularity between InfoMap and Louvain). The difference between our best partition is calculated in percents. For example, an increase in average degree results in a decrease in modularity, but an increase in expansion. We already saw a performance improvement for the conductance performance measure. Figure 6 shows that there exist a correlation ($\rho = 0.64$) between conductance change and the clustering coefficient. This means that if the clustering coefficient goes up, conductance goes up as well. We can conclude from this, that our method performs better on networks with a smaller clustering coefficient because we want to minimise the conductance. Also, while there exists a correlation ($\rho = -0.49$) between modularity change and the clustering coefficient. Although correlation does not imply causation, it gives some intuition in how certain values affect each other.

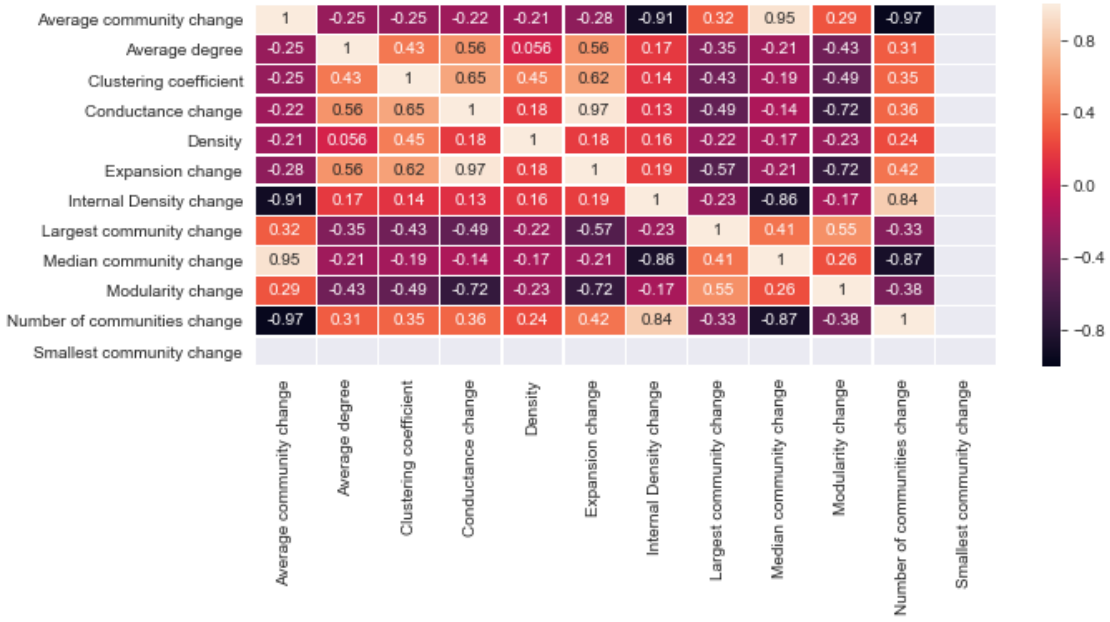
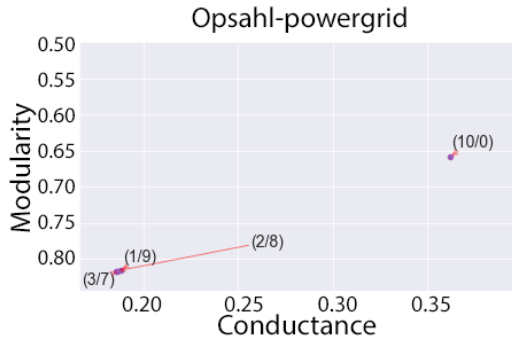


Figure 6.1: Correlation matrix showing how a change in certain network properties reflects in a change of quality metrics.

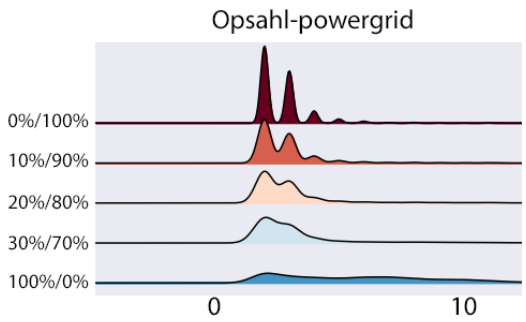
If we consider the expansion and internal density metrics, we find a correlation ($\rho = 0.62$) between expansion and the clustering coefficient. This strengthens our indication that our ensemble community detection algorithm is sensitive to the clustering coefficient of the network. A decrease in clustering coefficient results

Table 6.1: An overview of results of our algorithm on a collection of data sets compared to a baseline. Reported values are percentage of change.

data_set	Δ Modularity	Δ Conductance	Δ Min com.	Δ Max com.	Δ Median com.	Δ |com.|
out.arenas-jazz	-75.12	0.00	0.0	0.00	0.00	0.00
out.ca-AstroPh	-74.96	0.00	0.0	0.00	0.00	0.00
out.petster-hamster	-64.41	0.00	0.0	0.00	0.00	0.00
out.petster-friendships-hamster-uniq	-64.07	0.00	0.0	0.00	0.00	0.00
out.arenas-email	-61.24	0.00	0.0	0.00	0.00	0.00
email-Enron.txt	-60.44	0.00	0.0	0.00	0.00	0.00
out.moreno_zebra_zebra	-54.51	0.00	0.0	0.00	0.00	0.00
out.as-caida20071105	-0.18	-22.35	0.0	181.85	166.67	-61.45
out.moreno_propro_propro	-0.13	-58.03	0.0	4.26	33.33	-41.56
out.adjnoun_adjacency_adjacency	0.00	0.00	0.0	0.00	0.00	0.00
out.tntp-ChicagoRegional	0.00	-64.13	0.0	23.08	0.00	-1.56
out.arenas-pgp	0.00	-47.40	0.0	0.00	0.00	-10.84
out.contiguous-usa	0.00	-34.81	0.0	14.29	11.11	-3.00
out.ucidata-zachary	0.00	-56.49	0.0	54.55	0.00	-5.00
out.reactome	0.00	0.00	0.0	0.00	0.00	0.00
out.ego-facebook	0.00	0.00	0.0	0.00	0.00	0.00
out.maayan-pdzbase	0.00	-62.75	0.0	108.33	0.00	-3.21
out.maayan-vidal	0.16	-37.62	0.0	0.00	33.33	-38.77
out.as20000102	0.85	-21.49	0.0	198.14	0.00	-5.96
out.dolphins	6.02	-26.27	0.0	42.86	14.29	-9.00
out.opsahl-powergrid	24.16	-61.24	0.0	227.27	50.00	-21.87
out.subelj_euroroad_euroroad	27.33	-63.66	0.0	260.00	0.00	-6.76



(a) Raw results with conductance on the horizontal axis and modularity on the vertical axis. The points correspond to the partitions, and are annotated with the ratio.



(b) Community size distribution for the OPSAHL-POWERGRID network. On the horizontal axis the community size is shown, while on the vertical axis the number of occurrence of a certain community size density is shown.

Figure 6.2: Results for the OPSAHL-POWERGRID network.

in a decrease in expansion. As we would like to minimise this value, we can again conclude that our algorithm returns a partition with more community like subgraphs for networks with a lower clustering coefficient.

Important to note is that while we test all the different combinations between InfoMap and Louvain, not every combination is guaranteed to converge (see Section 5.2). This is no problem as long as at least one of the combinations converges.

Our method has more impact on conductance and expansion than on the internal density of the communities. This indicates that one should always look at multiple quality metrics, while assessing the quality of a given partition.

6.3 Validating real-networks with multiple metrics

We have seen that our community detection algorithm can lead to better results for networks with a low clustering coefficient. So, we have to validate our results on multiple quality metrics. We picked two networks to show an example of how to analyse community detection results by using this method.

Figure 6.2a and Figure 6.3a show the pareto fronts for the OPSAHL-POWERGRID and DOLPHINS network. The pareto fronts show both modularity and conductance. For practical reasons in both figures, we inverted the vertical axis. This way, both quality metrics can be minimised. Values lower and more to the left should be interpreted as better. The ratio between Louvain and InfoMap (InfoMap/Louvain) is added as an annotation.

We see that for the OPSAHL-POWERGRID network (Figure 6.2a), which is an infrastructure network, our algorithm outperforms full Louvain or InfoMap. For the ratios (30%/70%, 10%/90%, 20%/80%), we reach better results for both modularity and conductance. Because the results are close to each other, we can use community size distribution to look into the differences between the partitions. Figure 6.2b shows these community size distributions. Again, we smoothed the result for visualization purposes. The missing measurements on the vertical axis indicate that this ratio has not converged.

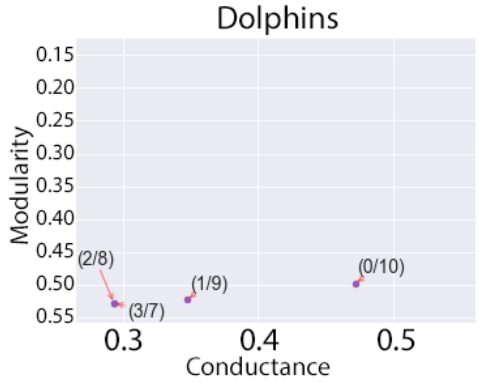
Figure 6.3a shows the Pareto front for the Dolphins network. This network has edges when there has been a frequent association between two dolphins. Here, we see that our algorithm outperforms even the 100% Louvain community detection. It does this with a large difference, especially when considering conductance. Even though the ratios (20%/80% and 30%/70%) give the best performance when considering these two metrics,

one can also take (10%/90%) or even (0%/100%) into account when deciding on what partition to use. Figure 6.3b shows the different community size distributions. Again, we see that the algorithm has not converged for some ratios (i.e. 50%/50%). This is not a problem as we have other, better performing partitions as well. Next, we observe large differences between the better scoring partitions (visualized on the left), and the ones that are discarded for visualization purposes. Instead of smaller communities, a distribution with more variance in community sizes leads to better results in this case. We observe that only a few of the combinations have converged.

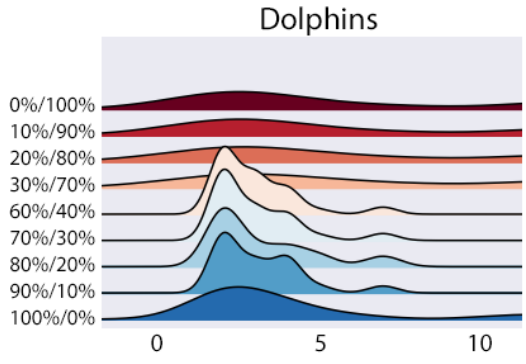
Advantages Our method has several benefits. First, our algorithm return better results when networks have a low clustering coefficient. Next, we can use this method with all kinds of different algorithms, as long they support weighted networks. The algorithm returns partitions for all the different ratio outputs. Using pareto fronts, this gives us freedom to decide on what quality metrics to prefer. We can conclude that our proposed method gives, most of the times, at least similar results as 100% InfoMap or Louvain. In some cases, we can reach an improvement in one of both metrics at the cost of worsening the other one. Our algorithm performs especially well when network have a low clustering coefficient.

Drawbacks One of the drawbacks of the method is that it is relatively computationally expensive. Although the calculation of the consensus partition can reach linear complexity (as it origins from fast consensus clustering), the overall performance will decrease when slower community detection algorithms are used. The algorithm will run each combination between the community detection algorithms. This is a constant factor in the complexity. In our implementation, this is ran in parallel. But, if this is not possible, the calculation will slow down a lot. Next, in the proposed combination of Louvain/InfoMap, we show that the results not necessarily improve on all the networks. Also, because no single metric is perfect, even with multiple metrics in apareto front and given the community size distribution, human interpretation remains to be needed.

Discussion Some recent research in this field is done by Poulin [**poulin2019ensembleoriginal**, 6], leading to their ensemble clustering algorithm (ECG) algorithm. In their paper, they apply Louvain in combination with consensus clustering on data generated with the LFR benchmark. We want to stress the differences in our approach compared to the approach chosen by Poulin. Although they use consensus clustering to combine



(a) Raw results with conductance on the horizontal axis and modularity on the vertical axis. The points correspond to the partitions, and are annotated with the ratio. The results for (8/2), (9/1), (10/0), (7/3) and (6/4) fall outside the figure and are discarded for visualization purposes.



(b) Figure showing the community size distributions. If we combine them with the information from Figure 6.3a, it gives us a good overview of the quality of specific mappings. On the horizontal axis the community size is shown, while on the vertical axis the number of occurrence of a certain community size density is shown.

Figure 6.3: Results for the DOLPHINS network.

different partitions, their focus is solely on the Louvain algorithm. This idea is in fact already describe by Fortunato in his overview work. Our algorithm tries to combine both InfoMap (Section 5.1) and Louvain (Section 5.1) given in a certain ratio into a consensus partition. Next to that, their work is validated by using the LFR benchmark instead of using a range of real-world networks.

Chapter 7

Conclusion and Future Work

Community detection on real-world networks is a widely researched subject. Many different methods for creating a partition exist. While the general idea of a community feels quite natural, practice shows that community detection on real-world networks is difficult. To find out why community detection is a non-trivial task, we conducted an extensive literature review. As a result, in our introduction, we define four problems: the large number of different algorithms, the fact that there is no single good solution, the large number of metrics and the fact that specific algorithms perform better on specific structures. To tackle some of these problems, we propose two solutions.

As first contribution, we propose an ensemble community detection method. Our method uses the framework of fast consensus clustering to combine multiple community detection algorithms. The algorithm outputs partitions for all the ratios between the two algorithms. These partitions can then be analyzed based on the desired quality metrics. One of the benefits of our approach is that we can use any community detection method as long as they support weighted networks. For our experiments, we used the widely used Louvain algorithm which is oriented at modularity optimization, together with InfoMap which is a dynamics oriented approach. To validate the performance of our algorithm, we tested the performance on over 40 real-world networks from different categories. We show that the proposed algorithm performs better on networks with a low clustering coefficient resulting in lower conductance scores and increased modularity.

The second contribution addresses two problems: the problem with the large number of metrics and the fact that there is no single good solution. We use Pareto fronts to optimize for multiple metrics at the same time. This enables us to find trade offs between different metrics. These results are then interpreted in context of the community size distribution. These distributions can then be analyzed with domain knowledge to decide which partition is the most useful for that particular use case.

Several topics can be considered as future work. First, improvements to the algorithm. For example, changing the ratio between the algorithms for each iteration of the consensus clustering algorithm can improve the performance and use the strong points of certain algorithms. Other improvements can be made by considering other algorithms instead of Louvain and InfoMap. Research can also expand on the number of algorithms, for example by adding more than two algorithms. Next to improvements to the algorithm, additional research can be done by validating the performance of the current algorithm with the LFR-benchmark. This way, we test against artificial networks resulting on a more object estimation about the quality of the method.

Ensemble community detection proves to be an easily extendable framework for applying community detection with the possibility of swapping out algorithms. By using this approach, the benefits from the selected algorithms can be combined in a new and better ensemble algorithm.

Bibliography

[1] Wayne W Zachary. "An information flow model for conflict and fission in small groups". In: *Journal of Anthropological Research* (1977), pp. 452–473.

[2] Zhao Yang, René Algesheimer, and Claudio J Tessone. "A comparative analysis of community detection algorithms on artificial networks". In: *Scientific Reports* 6 (2016), p. 30750.

[3] Himansu Sekhar Pattanayak, Harsh K Verma, and AL Sangal. "Community Detection Metrics and Algorithms in Social Networks". In: *Proceedings of theInternational Conference on Secure Cyber Computing and Communication (ICSCCC)*. IEEE. 2019, pp. 483–489.

[4] Santo Fortunato and Darko Hric. "Community detection in networks: A user guide". In: *Physics Reports* 659 (2016), pp. 1–44.

[5] V. A. Traag, L. Waltman, and N. J. van Eck. "From Louvain to Leiden: guaranteeing well-connected communities". In: *Scientific Reports* 9.1 (2019), p. 5233. DOI: 10.1038/s41598-019-41695-z.

[6] Valérie Poulin and François Théberge. "Ensemble clustering for graphs: comparisons and applications". In: *Applied Network Science* 4.1 (July 2019), p. 51. ISSN: 2364-8228. DOI: 10.1007/s41109-019-0162-z. URL: https://doi.org/10.1007/s41109-019-0162-z.

[7] Antonio Maria Fiscarelli et al. "A Memory-Based Label Propagation Algorithm for Community Detection". In: *Proceedings of the International Conference on Complex Networks and their Applications*. Springer. 2018, pp. 171–182.

[8] Santo Fortunato and Marc Barthelemy. "Resolution limit in community detection". In: *Proceedings of the National Academy of Sciences* 104.1 (2007), pp. 36–41.

[9] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. "Benchmark graphs for testing community detection algorithms". In: *Physical review E* 78.4 (2008), p. 046110.

[10] Santo Fortunato. "Community detection in graphs". In: *Physics reports* 486.3-5 (2010), pp. 75–174.

[11] Fragkiskos D Malliaros and Michalis Vazirgiannis. "Clustering and community detection in directed networks: A survey". In: *Physics Reports* 533.4 (2013), pp. 95–142.

[12] Ulrike Von Luxburg. "A tutorial on spectral clustering". In: *Statistics and Computing* 17.4 (2007), pp. 395–416.

[13] Brian Karrer and Mark EJ Newman. "Stochastic blockmodels and community structure in networks". In: *Physical Review E* 83.1 (2011), p. 016107.

[14] Andrea Lancichinetti et al. "Finding statistically significant communities in networks". In: *PloS one* 6.4 (2011), e18961.

[15] Vincent D Blondel et al. "Fast unfolding of communities in large networks". In: *Journal of Statistical Mechanics: Theory and Experiment* 2008.10 (2008), P10008.

[16] Martin Rosvall and Carl T Bergstrom. "Maps of information flow reveal community structure in complex networks". In: *arXiv preprint physics.soc-ph/0707.0609* (2007).

[17] Martin Rosvall, Daniel Axelsson, and Carl T Bergstrom. "The map equation". In: *The European Physical Journal Special Topics* 178.1 (2009), pp. 13–23.

[18] Mark EJ Newman and Michelle Girvan. "Finding and evaluating community structure in networks". In: *Physical Review* 69.2 (2004), p. 026113.

[19] Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. "Near linear time algorithm to detect community structures in large-scale networks". In: *Physical Review E* 76.3 (2007), p. 036106.

[20] Boaz Nadler and Meirav Galun. "Fundamental limitations of spectral clustering". In: *Proceedings of the Advances in neural information processing systems*. 2007, pp. 1017–1024.

[21] Cristopher Moore. "The computer science and physics of community detection: Landscapes, phase transitions, and hardness". In: *arXiv preprint arXiv:1702.00467* (2017).

[22] Paul W Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. "Stochastic blockmodels: First steps". In: *Social Networks* 5.2 (1983), pp. 109–137.

[23] Tiago P Peixoto. "Hierarchical block structures and high-resolution model selection in large networks". In: *Physical Review X* 4.1 (2014), p. 011047.

[24] Clement Lee and Darren J Wilkinson. "A review of stochastic block models and extensions for graph clustering". In: *arXiv preprint arXiv:1903.00114* (2019).

[25] Vincent A Traag, Paul Van Dooren, and Yurii Nesterov. "Narrow scope for resolution-limit-free community detection". In: *Physical Review E* 84.1 (2011), p. 016114.

[26] Pascal Pons and Matthieu Latapy. "Computing communities in large networks using random walks." In: *J. Graph Algorithms Appliedf* 10.2 (2006), pp. 191–218.

[27] Michele Coscia. "Discovering Communities of Community Discovery". In: *arXiv preprint arXiv:1907.02277* (2019).

[28] Jure Leskovec, Kevin J Lang, and Michael Mahoney. "Empirical comparison of algorithms for network community detection". In: *Proceedings of the International Conference on World Wide Web*. ACM. 2010, pp. 631–640.

[29] Wenye Li and Dale Schuurmans. "Modular community detection in networks". In: *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*. 2011.

[30] Ulrik Brandes et al. "On modularity clustering". In: *IEEE Transactions on Knowledge and Data Engineering* 20.2 (2008), pp. 172–188.

[31] Scott Kirkpatrick, C Daniel Gelatt, and Mario P Vecchi. "Optimization by simulated annealing". In: *science* 220.4598 (1983), pp. 671–680.

[32] Alex Arenas, Alberto Fernandez, and Sergio Gomez. "Analysis of the structure of complex networks at different resolution levels". In: *New Journal of Physics* 10.5 (2008), p. 053039.

[33] Jörg Reichardt and Stefan Bornholdt. "Statistical mechanics of community detection". In: *Physical Review E* 74.1 (2006), p. 016110.

[34] Andrea Lancichinetti and Santo Fortunato. "Limits of modularity maximization in community detection". In: *Physical Review E* 84.6 (2011), p. 066122.

[35] Jianbo Shi and Jitendra Malik. "Normalized cuts and image segmentation". In: *Departmental Papers (CIS)* (2000), p. 107.

[36] Ravi Kannan, Santosh Vempala, and Adrian Vetta. "On clusterings: Good, bad and spectral". In: *Journal of the ACM (JACM)* 51.3 (2004), pp. 497–515.

[37] Jure Leskovec et al. "Statistical properties of community structure in large social and information networks". In: *Proceedings of the International Conference on World Wide Web*. ACM. 2008, pp. 695–704.

[38] Vinh-Loc Dao, Cécile Bothorel, and Philippe Lenca. "Estimating the similarity of community detection methods based on cluster size distribution". In: *Proceedings of the International Conference on Complex Networks and their Applications*. Springer. 2018, pp. 183–194.

[39] Donald Ervin Knuth. *The Stanford GraphBase: a platform for combinatorial computing*. AcM Press New York, 1993.

[40] Jérôme Kunegis. "KONECT: the Koblenz network collection". In: *Proceedings of the International Conference on World Wide Web (WWW)*. ACM. 2013, pp. 1343–1350.

[41] Thomas Aynaud et al. "Multilevel local optimization of modularity". In: *Graph Partitioning* (2013), pp. 315–345.

[42] Jorma Rissanen. "Modeling by shortest data description". In: *Automatica* 14.5 (1978), pp. 465–471.

[43] David A Huffman. "A method for the construction of minimum-redundancy codes". In: *Proceedings of the IRE* 40.9 (1952), pp. 1098–1101.

[44] Tatsuro Kawamoto and Martin Rosvall. "Estimating the resolution limit of the map equation in community detection". In: *Physical Review E* 91.1 (2015), p. 012809.

[45] Online; accessed 27-Aug-2019. URL: `https://www.mapequation.org/code.html`.

[46] Andrea Lancichinetti and Santo Fortunato. "Consensus clustering in complex networks". In: *Scientific Reports* 2 (2012), p. 336.

[47] Aditya Tandon et al. "Fast consensus clustering in complex networks". In: *arXiv preprint arXiv:1902.04014* (2019).

[48] Nicola Beume, Boris Naujoks, and Michael Emmerich. "SMS-EMOA: Multiobjective selection based on dominated hypervolume". In: *European Journal of Operational Research* 181.3 (2007), pp. 1653–1669.

[49] G Joshi-Tope et al. "Reactome: a knowledgebase of biological pathways". In: *Nucleic acids research* 33.suppl_1 (2005), pp. D428–D432.