Bachlor's thesis

Stochastic Block Model

Department of Statistics
Ludwig-Maximilians-University Munich

by Cornelia Gruber
supervised by Prof. Dr. Göran Kauermann &
Benjamin Sischka
Munich, July 26, 2019

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit selbständig verfasst und dabei keine anderen als die angegebenen Hilfsmittel benutzt habe. Sämtliche Stellen der Arbeit, die im Wortlaut oder dem Sinn nach Publikationen oder Vorträgen anderer Autoren entnommen sind, habe ich als solche kenntlich gemacht. Die Arbeit wurde bisher weder gesamt noch in Teilen einer anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

.....	.....
Ort, Datum	Cornelia Gruber

Contents

List of Figures

List of Tables

1 Introduction

Nowadays, many people unwittingly take advantage of the advances in network analysis in their daily lives. Many new technologies depend on it and companies use it to make purchase recommendations based on previous purchases and offer more relevant products to their customers. There are many other systems in our everyday lives that can be adequately represented as networks, such as social networks, the Internet or citation networks.

What each of these networks have in common is that they consists of a number of *nodes* or *vertices*, that are connected by *edges*. Think of the internet and data connections between computers or routers, or people in a social network and their relationships.

An interesting characteristic that some networks feature is the natural division of nodes into groups, clusters or communities that have different connection behavior within groups than between groups. To come back to our example the communities could be different groups of friends. One common way to describe such groups is the *strong* definition of community, which states that the density, i.e the number of the edges between nodes within a community is higher than between different communities (see Radicchi et al., 2004).

Although most algorithms for detecting community structures are based on this assumption, I will not limit this thesis to it and also allow groups where the between connectivity exceeds the within connectivity.

One example of this property would be if clusters are defined by gender and edges represent romantic relationships. Which means even-though all males make up one community they have less romantic relationships between each other than to the community of all females.

Finding communities in a network is an important task in network analysis because they can differ strongly in their characteristics and therefore a consideration of the whole system could lead to wrong conclusions. Often statements can only be made about specific subgroups.

However, if you find yourself in the situation where a clustering of nodes in a network needs to be done, it is crucial to know the performance of different community detection algorithms depending on your network structure. In order to compare different algorithms with each other, you have to test them on a network whose structures are already known. On the basis of this ground truth one can then make statements about the abilities of the models in finding the given clusters.

This thesis will present a model for the generation of networks with community structure, the so-called stochastic block model (SBM). We will then compare different algorithms in terms of their detection accuracy and computation time on networks simulated with this approach.

2 Prerequisites & Definitions

This paragraph explains some basic principles that contribute to the understanding of the simulation study and the workings of the algorithms.

2.1 Stochastic block model

Network structures are usually described via graphs. A graph (G) is a set of nodes, also called vertices (V), which are connected by edges (E). Throughout this thesis an undirected and unweighted graph $G = (V, E)$ consisting of $n = |V|$ nodes and $m = |E|$ edges is considered. This means that the edges have no direction, i.e. no fixed start and end point and each existing edge has the same weight of 1. A graph can be represented by an adjacency matrix A: $A_{ij} = 1$ if there exists an edge from vertex i to j and $A_{ij} = 0$ if not. For an undirected graph the adjacency matrix is symmetric ($A_{ij} = A_{ji}$). Some algorithms can be extended to weighted graphs, meaning $A_{ij} \in \mathbb{R}$ instead of $A_{ij} \in \{0, 1\}$. However, only the base case is considered, where each edge is equally weighted. The degree $d(i) = \sum_j A_{ij}$ of vertex i is the number of directly connected nodes (its neighbours) and can be calculated as the sum of all edges that start from i.

The stochastic block model is a generative model for random graphs which usually produces networks containing community structure. This means that each node has a fixed community membership, which determines with which probability an edge exists to other nodes. The model is defined by the number of vertices n, the number of communities C, a probability vector $\alpha = (\alpha_1, \ldots, \alpha_C)$ specifying the distribution of nodes on the C communities and a symmetric matrix $W \in \mathbb{R}^{CxC}$ with entries in [0, 1] specifying the connectivity probabilities. An example how the adjacency matrix of a $SBM(n, \alpha, W)$ with

$$n = 300,\ \alpha = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3}),\ W = \begin{pmatrix} 0.12 & 0.01 & 0.01 \\ 0.01 & 0.12 & 0.01 \\ 0.01 & 0.01 & 0.12 \end{pmatrix}$$

can look like can be found in figure (1).

A graph G with community labels X is drawn under $SBM(n, \alpha, W)$ if X is an n-dimensional vector with i.i.d. components distributed under α and G is an graph with n vertices, where vertices i and j are connected with probability W_{X_i, X_j} independently of other pairs of vertices. The community sets, also referred as partitioning, are defined by $K_v(X) := \{i \in n : X_i = v\}$, for $v = 1, \ldots, C$ (see Abbe, 2017).

The SBM under which the networks are drawn will be symmetric, which means α is uniform and W takes the same value on the diagonal and the same value outside the diagonal.

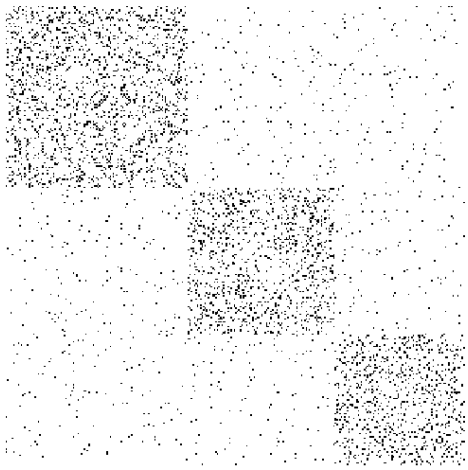

Figure 1: A heat-map where each black pixel represents $A_{ij} = 1$ in the adjacency matrix of a SBM with a probability of an edge within or between the groups of 12% or 1%, respectively. The term "Block Model" comes from the visual representation of an adjacency matrix when nodes are grouped by class membership and dense regions form blocks.

2.2 Modularity - a measure of community structure

Modularity is an objective function often used in community detection algorithms to determine how strong the division of vertices into groups or communities is for a given community assignment.

Modularity depends on following terms: e_{ij} denotes the fraction of edges connecting a node in community i to a node in community j, therefore e_{ii} denotes the fraction of edges within community i. If the fraction of all edges having at least one node in community i is denoted by $a_i = \sum_j e_{ij}$, then the expected value for e_{ii} would be $a_i a_i = a_i^2$ if edges were distributed randomly and independently of class membership. The modularity Q of a partitioning P with communities 1 to C is given by the sum of all communities (C) of the observed edge fraction (e_{ii}) minus the expected edge fraction at random distribution of the edges (a_i^2).

$$Q(P) = \sum_{i=1}^{C} (e_{ii} - a_i^2) \tag{1}$$

The value of modularity Q lies in the range [-1, 1] and is positive when the observed number of edges within the communities is higher than the expected number under random connections. In this case you can reasonably infer the existence of a latent cluster structure, as stated by Li and Schuurmans (2011) and Newman (2004). Therefore many algorithms try to maximize the modularity for a given network to uncover the underlying cluster structure. Modularity is used by the fast and greedy, the walktrap, the leading eigenvector and the louvain algorithm. All other algorithms considered rely on different objective functions.

2.3 NMI - an accuracy measure for cluster-solutions

For each network drawn under SBM the real partitioning is compared to the partitioning results of the algorithms. The more the partitionings match, the better the algorithms perform and the better they find the true underlying community structure. To obtain an objective measure of the accuracy, the normalized mutual information NMI, following the notation in Yang et al. (2016), is used. Where P is the true partitioning with C classes as specified by the SBM and $\tilde{P}$ is the partitioning with $\tilde{C}$ clusters detected by the algorithm. First we need to define a confusion matrix N where element N_{ij} is the number of nodes which are in the true community i and in the predicted community j. Accordingly, the rows correspond to the "real" communities and the columns to the "found" communities. The sum of row i is denoted by $N_{i.}$ and analogously $N_{.j}$ is the sum of the column j.

$$
\begin{aligned}
NMI(P, \tilde{P}) &= \frac{I(P, \tilde{P})}{\frac{1}{2}\left[H(P) + H(\tilde{P})\right]} \\
&= \frac{-\sum_{i=1}^{C}\sum_{j=1}^{\tilde{C}} N_{ij}\ log(\frac{N_{ij}N}{N_{i.}N_{.j}})}{\frac{1}{2}\left[\sum_{i=1}^{C} N_{i.}\ log(\frac{N_{i.}}{N}) + \sum_{j=1}^{\tilde{C}} N_{.j}\ log(\frac{N_{.j}}{N})\right]}
\end{aligned}
\tag{2}
$$

The mutual information $I(P, \tilde{P})$ measures the amount of information by which the knowledge of the true classes increases when cluster membership is known. $I(P, \tilde{P})$ reaches its minimum of 0 if the clustering is random with respect to class membership and it reaches its maximum if the clustering perfectly recreates the true classes - but also if those clusters are further subdivided into smaller clusters (see Manning et al., 2008). Because a partitioning with $C = N$ has maximum mutual information, but also maximum entropy $H(\tilde{P}) = \log N$, the normalization by the denominator $\frac{1}{2}\left[H(P) + H(\tilde{P})\right]$ ensures a low value of $NMI(P, \tilde{P})$ for partitionings where each node forms a single cluster. The entropy H is a measure of uncertainty of a discrete distribution P and can be calculated as

$$
H(P) = -\sum_{x \in X} \mathbb{P}(x) \log \mathbb{P}(x) \tag{3}
$$

where X is the set of possible values for the distribution. Additionally, the normalization allows comparison of partitionings if the number of found clusters $\tilde{C}$ is different from the true number of clusters C and it ensures $NMI \in [0, 1]$.

3 Community Detection Algorithms

Because there are many different ways for detecting communities in networks several approaches are introduced. All algorithms, but `bm_bernoulli` from `blockmodels` are implemented in the `igraph` library in R (see Csardi and Nepusz, 2006).

Clustering algorithms can be categorised into different approaches. Hierarchical clustering approaches try to impose a hierarchy of different partitioning solutions. A distinction is made between agglomerative and divisive processes. In agglomerative, also known as "bottom-up", processes each object is initially in its own cluster and two clusters are merged as you rise up the hierarchy. While with divisive, or "top-down" approaches all objects are at first in one cluster and recursive splits are made until each object is in its own cluster. In contrast to partitioning algorithms such as k-means, not a single clustering solution is obtained, but a sequence of clusterings in which you have to decide which partitioning best represents the community structure in the network based on an objective function. So called model-based approaches attempt to reconstruct the distribution under which the network formed in order to group together the objects belonging to the same origin.

3.1 Fast and Greedy

The `cluster_fast_greedy` algorithm is an agglomerative hierarchical clustering approach. At first each element forms it's own cluster. Then the two cluster are joined which yield the largest increase in the current value of modularity. Because the nodes to merge are chosen in a locally optimal manner this algorithm is called "greedy". Once the optimal clustering is found further joining clusters will decrease the modularity. After the optimal value is found the two clusters which result in the smallest decrease of modularity are joined. Merging clusters is repeated until all nodes are in one cluster. The clustering where the maximal value of modularity was reached is then taken as final result (see Clauset et al., 2004). In this algorithm not only the final partitioning result but the nodes to merge are chosen based on modularity.

3.2 Walktrap

The `cluster_walktrap` algorithm is based on the idea that random walks on a graph tend to get "trapped" into densely connected parts corresponding to communities and was proposed by Pons and Latapy (2005). In a *random walk*, a walker "stands" on a vertex and moves randomly and uniformly to an adjacent vertex. The probability to transition from vertex i to vertex j is $P_{ij} = \frac{A_{ij}}{d(i)}$. Further P_{ij}^t denotes the probability to go from vertex i to j in t steps. When choosing the number of steps t one should consider the density of the network and decrease t in dense networks and increase it otherwise. However, it is empirically observed that best results are obtained with $3 \leq t \leq 8$ (see Pons and Latapy, 2005). Two vertices of the same community tend to "see" the distance to other vertices in the same way. Thus if i and j are in the same community, the probabilities to reach any other vertex k will be similar or equal for i and j ($\forall k$, $P_{ik}^t \simeq P_{jk}^t$). With this property it is possible to define r_{ij} as the euclidean distance between those probabilities normalized to the degree of a node k, $d(k)$, and then generalizing it to distances between sets of vertices (communities). Therefore you get:

$$r_{ij} = \sqrt{\sum_{k=1}^{n} \frac{(P_{ik}^t - P_{jk}^t)^2}{d(k)}} \tag{4}$$

P^t_{Cj} is the probability to walk from community C to vertex j in t steps and can be calculated as the average probability to walk from a vertex in community C to j:

$$P^t_{Cj} = \frac{1}{|C|} \sum_{i \in C} P^t_{ij} \tag{5}$$

Thus you can define the distance between two communities C_1 and C_2 as

$$r_{C_1C_2} = \sqrt{\sum_{k=1}^{n} \frac{(P^t_{C_1k} - P^t_{C_2k})^2}{d(k)}} \tag{6}$$

The clustering itself is an agglomerative approach where in every step k two communities are merged which lead to the smallest increase of the mean squared distance σ_k between each vertex and its community (see Ward's method, Ward (1963)) and is calculated as follows:

$$\sigma_k = \frac{1}{n} \sum_{C \in P_k} \sum_{i \in C} r^2_{iC} \tag{7}$$

where P_k denotes the partitioning in the k-th step. To choose the partitioning which captures the community structure best, the modularity Q is maximized (see equation (1)).

3.3 Leading eigenvector

`cluster_leading_eigen` is another algorithm which maximizes modularity of a partitioning. Compared to the previous algorithms, it is a divisive approach using eigenvalues and eigenvectors of a so called modularity matrix to detect possible splits into two communities so the network has high modularity. The modularity matrix B is $B = A - P$, where A is the adjacency matrix as defined in section (2). P contains the probabilities P_{ij} that an edge between vertices i and j exists given the degree of these nodes. Now it is possible to rewrite the modularity Q as Newman (2006) showed as

$$Q = \frac{1}{4m} s^T B s \tag{8}$$

Where m is the number of edged and s is a index vector with n elements

$$s_i = \begin{cases} +1 \text{ if vertex i belongs to group 1} \\ -1 \text{ if vertex i belongs to group 2} \end{cases} \tag{9}$$

It is possible to rewrite s as $s = \sum_i a_i u_i$ with $a_i = u_i^T s$ where u_i is the normalized eigenvector of the modularity matrix B. If s is inserted into equation (8) you get

$$Q = \frac{1}{4m} \sum_i a_1^2 \lambda_i \tag{10}$$

where λ_i is the eigenvalue of B corresponding to the eigenvector u_i and the eigenvalues are ordered according to their value with: $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$. Now maximizing

the modularity Q is equivalent to choosing a_i^2 in a way that places most of the weight in the sum (10) on the terms that correspond to the largest eigenvalues. The network is then separated by calculating the eigenvector of the modularity matrix for the largest positive eigenvalue and then dividing vertices into two communities based on the sign of the corresponding element in the eigenvector:

$$s_i = \begin{cases} +1 \text{ if } u_i^{(1)} \geq 1 \\ -1 \text{ if } u_i^{(1)} < 1 \end{cases} \tag{11}$$

If after some steps all elements in the eigenvector are of the same sign it means that the network has no underlying community structure anymore and the algorithm stops the division in case the modularity cannot be further increased (see Newman, 2006).

3.4 Label propagation

The `cluster_label_prop` algorithm follows the basic idea that each vertex gets the label that most of its neighbors belong to. First of all, each node has its own label and the labels are updated by majority vote in the neighborhood of the node, with ties broken uniformly and randomly. Densely connected groups of nodes quickly reach consensus on a unique label, as the labels propagate. At the end of the propagation process, i.e. when the label of each node is equal to the most common label in its neighborhood, nodes with the same labels are grouped together as a community (see Raghavan et al., 2007).

Since the algorithm starts by giving each node its own label, the first iterations lead to many small and dense pockets with the same label. These consensus groups then gain impact and try to acquire more nodes. However, when a consensus group reaches the border of another group, they begin to compete for members. If a contested node has more edges to nodes from its current community than to the competing one, it will keep its label. Otherwise the label of the competing group will be adopted. This corresponds directly to the strong definition of communities which was introduced in section (1).

Because there is no optimization criterion, but rather a stopping condition, different solutions can be obtained for the same starting value. More precisely, since the algorithm breaks the ties uniformly and randomly, a node can select a random community early in the label propagation process when the possibilities of the ties are high. This allows several community structures to be reached from the same initial situation. Therefore multiple runs of the label propagation algorithm should be computed.

3.5 Louvain

The `cluster_louvain` algorithm, which is also known as `multilevel.community`, is an agglomerative, hierarchical approach proposed by Blondel et al. (2008). It is characterized by being one of the fastest algorithms which are considered in this

thesis. The procedure is as follows: initially, each node is assigned to a community on its own. In each step, a node is moved into the cluster, which increases modularity the most, similar to `cluster_fast_greedy`. Every individual cluster found in this way is combined into a single node and the process begins again with the merged communities. This greedy approach is repeated for each node thereby formed until no further improvement is possible in this step. Hence after each step the number of vertices decreases in the recently merged network and the computing time for the next step is decreased because fewer node combinations need to be considered. The algorithm stops if no further gain in modularity is possible or if only a single node is left.

3.6 Infomap

The `cluster_infomap` uses maps to describe the structure between nodes and edges in networks that represent interactions among the subgraphs of a network. As Rosvall and Bergstrom (2008) stated, "these local interactions induce a system-wide flow of information that characterizes the behavior of the full system". In order to comprehend the network structure you need to comprehend the flow of information in the network. Therefore, it is necessary to identify the modules that compose the network by finding an optimally compressed description of how information flows in the network. A group of nodes among which information flows quickly and easily can be described as a single densely connected module or cluster. Thus, the description of the information flow is a coding or compression problem and the objective function used in this algorithm is the minimization of the expected description length of a random walk, the so called *map equation*.

The core idea of the algorithm follows to a large extent the louvain method: neighboring nodes are joined together to form modules, which are then joined together to form the modules in the next step and so on. First, each node is assigned to its own module. Each node is then moved in random sequential order to the neighboring module, which leads to the largest decrease in the map equation. If no shift results in a reduction of the map equation, the node remains in its original module. This process is repeated each time in a new random order until no movement causes a decrease of the map equation. In other words the network is rebuilt, with the modules of the last level forming the nodes on the current level, and just like in the previous step, the nodes are grouped into modules. This hierarchical rebuilding of the network is repeated until the map equation cannot be further reduced (see Rosvall and Bergstrom, 2008).

3.7 Variational EM

The `bm_bernoulli` is an model based clustering approach using a variational expectation maximization (V-EM) algorithm. It was implemented in the R package `blockmodels` by Leger (2016) but the theoretical basis was established by Mariadassou et al. (2010). The so called membership matrix Z is defined as $Z_{iq} = 1$ if node i is a member of community q. This matrix is therefore a nxC matrix, where Z_i denotes the i-th row of matrix Z. In this approach the vertex i belongs to an

unknown community $\{K_1, \ldots, K_C\}$, thus to a unobserved or latent variable. A row in Z follows a multinomial distribution,

$$Z_i \sim M(1, \boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_C)) \tag{12}$$

where α_q is the probability that vertex i belongs to class K_q. Given the class memberships q and l, the edge between i and j, A_{ij} is distributed as $f(\cdot, \theta_{ql})$ where $f(\cdot, \theta_{ql})$ is a probability distribution known up to the parameter θ_{ql}.

$$A_{ij} | i \in q, j \in l \sim f(\theta_{ql}) \tag{13}$$

The bernoulli distribution is suitable for $f(\cdot, \theta_{ql})$ where the probability π_{ql} corresponds to θ_{ql}.

$$\pi_{ql} = \begin{cases} p_1 & \text{if } q = l \\ p_2 & \text{if } q \neq l \end{cases} \tag{14}$$

Now the model is specified by the mixture proportions $\boldsymbol{\alpha}$ and the connectivity matrix $\boldsymbol{\theta} = (\theta_{gl})_{q,l=1,\ldots,C}$ and the parameter of this model is denoted by $\gamma = (\boldsymbol{\alpha}, \boldsymbol{\theta})$. The matrix $\boldsymbol{\theta}$ is an estimator for the true connectivity matrix W. Because the edges are independent given the classes Z it is possible to make the following transformation based on Bayes' theorem: $\log \mathbb{P}(Z, A) = \log \mathbb{P}(Z) + \log \mathbb{P}(A|Z)$. It can be shown that the log-likelihood for the edges A and the indicator variables Z follows from equations (12) and (13):

$$\log \mathbb{P}(Z, A) = \sum_i \sum_q Z_{iq} \log \alpha_q + \sum_{i<j} \sum_{q,l} Z_{iq} Z_{jl} \log f_{ql}(A_{ij}) \tag{15}$$

As Mariadassou et al. (2010) showed the likelihood of the edge distribution can be obtained by summing $\mathbb{P}(Z, A)$ over all possible Z's: $\mathbb{P}(A) = \sum_Z \mathbb{P}(Z, A)$. This summation quickly becomes intractable since it involves C^n terms. Therefore a lower bound for the log-likelihood of the edge distribution is proposed which can finally be shown as

$$J(R_A, \gamma) = -\sum_i \sum_q \tau_{iq} \log \tau_{iq} + \sum_i \sum_q \tau_{iq} \log \alpha_q + \sum_{i<j} \sum_{q,l} \tau_{iq} \tau_{jl} \log f_{ql}(A_{ij}) \tag{16}$$

R_A stands for some distribution on Z where $\log \mathbb{P}(A; \gamma)$ reaches its maximum for $R_A(Z) = \mathbb{P}(Z|A)$. In the following the factored multinomial distribution with density h, probability vector $\tau_i = (\tau_{i1}, \ldots, \tau_{iC})$ and $\sum_q \tau_{iq} = 1$ is considered for R_A

$$R_A(Z) = \prod_i h(Z_i, \tau_i)$$

Accordingly if $\mathbb{P}(Z|A; \gamma)$ was solvable maximizing $J(R_A, \gamma)$ with respect to γ would be equivalent to maximizing $\log \mathbb{P}(A; \gamma)$. With this specification $J(R_A, \gamma)$ can be solved and τ_i is regarded as a variational parameter which needs to be optimized so that the best fit between $R_A(Z)$ and $\mathbb{P}(Z, A; \gamma)$ is achieved. Now that the model specifications are set the estimation of the parameters is possible.

$$
\begin{aligned}
R_A^{(n+1)} &= \arg\max_{R_A} \; J(R_A, \gamma^{(n)}) \\
\gamma^{(n+1)} &= \arg\max_{\gamma} \; J(R_A^{(n+1)}, \gamma)
\end{aligned}
\tag{17}
$$

The variational EM approach consists of two steps which are repeated until the estimates converge:

1. **Pseudo E step:** Maximization of J in regards to R_A to obtain an estimate for the parameter τ
2. **M step:** Maximization of J in regards to γ to obtain an estimate for $\boldsymbol{\alpha}$ and $\boldsymbol{\theta}$ with $\hat{\alpha} = \frac{1}{n}\sum_i \tau_{iq}$

Integrated Classification Likelihood (ICL)

The V-EM process is repeated for a various number of groups. In order to choose the model m_C with the optimal number of classes C the so called Integrated Classification Likelihood (ICL) criterion which was proposed by Biernacki et al. (2000) is maximized

$$
ICL(m_C) = \max_{\gamma} \; \log \mathbb{P}(A, \tilde{Z}|\gamma, m_C) - \frac{1}{2}\left[\underbrace{p_C \log(n(n-1))}_{\text{penalizing } \boldsymbol{\theta}} - \underbrace{(C-1)\log(n)}_{\text{penalizing } \boldsymbol{\alpha}}\right] \tag{18}
$$

where $\boldsymbol{\theta}$ contains p_C independent parameters and Z is replaced by its predicted values $\tilde{Z}$. In the ICL criterion the probability for the observed edge and class distribution is maximized while equally penalizing the length of the parameter vectors $\boldsymbol{\theta}$ and $\boldsymbol{\alpha}$. The number of independent parameters in $\boldsymbol{\theta}$ is p_C relating to a maximum number of $(n(n-1))$ edges and relating to all n nodes the number of independent parameters in $\boldsymbol{\alpha}$ is $C-1$. This is because α_C must be chosen in a way that $\boldsymbol{\alpha}$ sums up to 1.

4 Simulation Study

4.1 Network generation

It is of interest how clear the separation of the clusters has to be in order to obtain meaningful results from the algorithms. Therefore we define the probabilities to form an edge between different clusters as μ, following the mixing parameter in Yang et al. (2016). A low value of μ indicates a network with clear separation of communities, with the extreme case $\mu = 0$ meaning there are no connections between the clusters. We will consider networks with different numbers of nodes (n=300, n=500 and n=1500) and communities (C=3, C=5 and C=15). For each combination of n and C networks with step wise increasing μ are simulated to observe how the accuracy of the found partitionings changes.

In order to ensure the simulated networks resemble real world networks the density d of a network, i.e. the proportion of observed edges of all possible ones, is set rather low to an expected density of $d = 0,05$ (see Melancon, 2006). A symmetric SBM network with C communities, meaning $\frac{1}{C}$ of nodes belongs to each community, with an CxC connectivity matrix W_1,

$$W_1 = \begin{pmatrix} 1-\mu & & \mu \\ & \ddots & \\ \mu & & 1-\mu \end{pmatrix}$$

has the expected density d.

$$d = \frac{1}{C}(1-\mu) + \frac{C-1}{C}\mu = \frac{1-\mu+(C-1)\mu}{C} \tag{19}$$

This equation can be explained intuitively because every node has the probability $1-\mu$ of connecting to nodes in its own community, thus to $\frac{1}{C}$ of all nodes and a probability μ of connecting to the remaining nodes, thus to $\frac{C-1}{C}$ of all nodes. However, to ensure a constant expected density of $d = 5\%$ we multiply each probability in W_1 by a density parameter π, with

$$\pi = \frac{dC}{(1-\mu)+(C-1)\mu} \tag{20}$$

This gives

$$W_2 = \begin{pmatrix} (1-\mu)\pi & & \mu\pi \\ & \ddots & \\ \mu\pi & & (1-\mu)\pi \end{pmatrix}$$

The probabilities for an edge connecting different communities changed to $\mu\pi$. For every value of μ in 0.03 steps between 0 and 0.75 multiple networks with otherwise constant parameters n, C and d are simulated so the variance of the accuracy can be estimated. For algorithms where a change in accuracy in higher μ ranges was to be expected, μ values between [0, 1] were considered.

Note that when converting from $\mu\pi$ to μ it is always important to pay attention to the number of communities, because the density parameter π is always dependent on C (see equation (20)).

Since many algorithms use the definition that communities have a higher connectivity within clusters than between clusters, it is expected that those algorithms fail if the connectivity between clusters approaches or exceeds the connectivity within clusters. The value of equality $\mu\pi = (1-\mu)\pi$ is reached for $\mu = 0.5$ or $\mu\pi = d = 0.05$ (see figure (2)). This can be shown by the following transformations of the equations:

$$\mu = 0.5 \Rightarrow \mu\pi = 0.5\pi = 0.5\ \frac{dC}{0.5+(C-1)0.5} = 0.5\ \frac{dC}{0.5C} = d \tag{21}$$

We therefore expect the algorithms to fail as $\mu\pi$ approaches $d = 0.05$. However, algorithms which do not assume communities have a higher connectivity within but rather just take the connectivities $\mu\pi$ and $(1-\mu)\pi$ to be different might get

reasonable results once the difference between $(1-\mu)\pi$ (within connectivity) and $\mu\pi$ (between connectivity) increases again.

In figure (2) the x-axis corresponds to the $\mu\pi$ values in simulated networks with three communities ($C = 3$) and the y-axis shows the ratio between $1-\mu$ and μ. Hence, for each $\mu\pi$ the factor can be seen by which $1-\mu$ is greater or smaller than μ. The exact values can be taken from table (1).

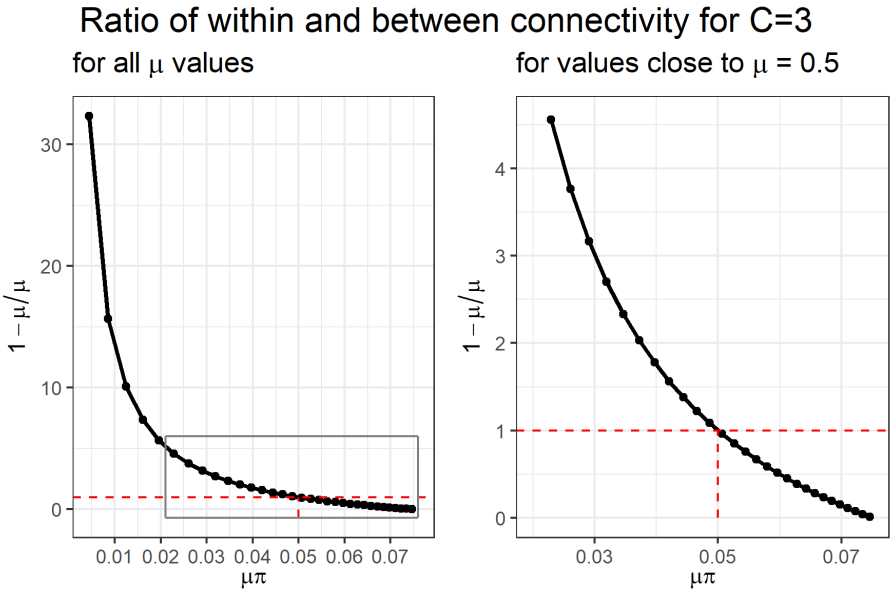

Figure 2: Ratio of within and between connectivity ($\frac{1-\mu}{\mu}$) depending on $\mu\pi$ for $C = 3$. Left: μ values in [0, 0.75]. Right: μ values in [0.15, 0.75] All points to the left of the vertical red dashed lines correspond to the strong definition of community ($1-\mu > \mu$).

μ	$\mu\pi$	$\frac{1-\mu}{\mu}$	μ	$\mu\pi$	$\frac{1-\mu}{\mu}$	μ	$\mu\pi$	$\frac{1-\mu}{\mu}$	μ	$\mu\pi$	$\frac{1-\mu}{\mu}$
0.03	0.004	32.3	0.27	0.032	2.7	0.51	0.051	0.9	0.75	0.064	0.33
0.06	0.008	15.6	0.30	0.035	2.3	0.54	0.053	0.8	0.78	0.066	0.28
0.09	0.012	10.1	0.33	0.037	2.0	0.57	0.054	0.7	0.81	0.067	0.23
0.12	0.016	7.3	0.36	0.040	1.7	0.60	0.056	0.6	0.84	0.068	0.19
0.15	0.020	5.6	0.39	0.042	1.5	0.63	0.058	0.5	0.87	0.070	0.14
0.18	0.023	4.5	0.42	0.044	1.3	0.66	0.060	0.5	0.90	0.071	0.11
0.21	0.026	3.7	0.45	0.047	1.2	0.69	0.061	0.4	0.93	0.072	0.07
0.24	0.029	3.1	0.48	0.049	1.0	0.72	0.063	0.3	0.96	0.073	0.04

Table 1: Table of all μ and corresponding $\mu\pi$ and $\frac{1-\mu}{\mu}$ values for $C = 3$

4.2 Results

4.2.1 Accuracy

Now that the simulation framework is established, the results of the different algorithms can be compared. A property that can be observed for almost all algorithms are that the more nodes there are, the longer the accuracy remains reasonable high as shown in figures (4) and (5). In addition, it can be observed in all algorithms that they perform best at very low $\mu\pi$ values and that the average NMI decreases when the values approach $\mu = 1 - \mu$, which corresponds to $\mu\pi = 0.05$. For values close to $\mu\pi = 0.05$, the algorithms fail completely and have an average NMI of 0. This can be explained because once the probabilities for edges within and between communities align, the algorithms have no indication of class membership. All except the variational-EM and the walktrap algorithm stay at 0 NMI for values $\mu\pi > 0.05$ even though the difference between μ and $1-\mu$ increases again. This can be derived directly from the strong definition of community on which the other algorithms are based on.

cluster_fast_greedy

The clustering results obtained from the fast and greedy algorithm start at the maximum value of NMI = 1 for networks with 3 or 5 true communities, however for networks with 15 clusters the NMI only reaches approximately 0.90 for 1500 nodes or 0.8 for 300 nodes. For networks with a between connectivity probability of more than 0.045 the results are not any better than random class assignment would be because the NMI drops to nearly 0.

cluster_walktrap

The walktrap algorithm provides consistently satisfactory results, since even for $C = 15$ clusters high NMI values of nearly 1 are achieved, while at the same time having the lowest standard deviation of the compared algorithms with values below $SD = 0.08$. Another peculiarity is the small increase of NMI for the walktrap algorithm for N = 1500 and C=3 at $\mu\pi \approx 0.06$ which corresponds to $\mu \approx 0.7$ (see table (1)) . The accuracy stays at $NMI \approx 0.3$ which is unfortunately not high enough for reasonable results. One reason for this increase might be the definition of communities via similar transition probabilities and not via the strong definition that edges have to be more common within communities than between. Additionally, this improvement can not be observed for different combinations of number of nodes and communities, so you should not rely on the walktrap algorithm if you suspect communities with lower within density than between.

cluster_leading_eigen

Detecting cluster structure in graphs based on eigenvalues is a reasonable approach for networks with a low number of communities as you can see quite high NMI values

for $C = 3$. However at $C = 5$ and especially $C = 15$ the maximal value of NMI is not even reached for very densely connected communities with $\mu\pi$ approaching 0.

cluster_label_prop

The label propagation algorithm has only a very limited parameter set for which the predicted class assignment is close to the true one. While the algorithms that were already described crashed at $\mu\pi$ values of 0.04, the label propagation algorithm shows this decrease already at $\mu\pi = 0.03$, referring to the results with 3 clusters. In figure (2) you can see that a value of $\mu\pi = 0.04$ for $C = 3$ means that the probability for an edge within a community is 1.5 times as high as the between probability, whereas at $\mu\pi = 0.03$ this factor is 3 . Therefore, a greater distinction between these probabilities is needed compared to the other approaches in order to obtain accurate results using the label propagation method.

cluster_louvain

The louvain method for detecting communities yields promising results as the NMI is reasonably high for all considered number of communities. However the standard deviation of this algorithm is noticeably higher than of `cluster_walktrap` or `cluster_fast_greedy`. Outstanding is, that the standard deviation also increases for increasing n, which indicates a very sudden drop of the NMI for a higher number of nodes compared to a lower n.

cluster_infomap

In terms of accuracy, The Infomap algorithm is on a similarly low level as the `cluster_label_prop`. It is noteworthy that with $C = 3$ and $n = 1500$ the standard deviation stays at 0, which means that the algorithm first matches the clusters perfectly and once the threshold of $\mu = 0.15$ or $\mu\pi = 0.02$ is exceeded there is no match at all. The same can be observed for the parameter combination $C = 15$, $n = 700$ and $\mu = 0.09$ or $\mu\pi = 0.03$ (see table 3).

bm_bernoulli

The only model-based clustering which is considered throughout this paper shows some interesting tendencies. It immediately catches the eye that this algorithm increases again in NMI for networks with $C = 3$ as well as $C = 5$ and $n = 500$. This can be explained by the fact that the model on which the V-EM is based on is very similar to the SMB that generated the networks. Accordingly, cluster structures in networks with three communities and $\mu\pi$ values exceeding 0.07 or μ values exceeding 0.87 and networks with 5 communities and $\mu\pi > 0.06$ corresponding to $\mu > 0.9$ can be detected sufficiently well. Unfortunately this algorithm does not reach a NMI of 1 for 3 true communities and therefore other algorithms should be favoured if a low number of clusters is expected. The NMI for C=3 and C=5 seems to be

symmetrical, but it is important not to assume the same behavior for 15 clusters, because to get higher values for $\mu\pi$, the μ values would have to exceed 1 given the density stays at 5%. However, this would violate the condition that was set for the density of the networks, namely that the density should be a maximum of 5 % to resemble real networks.

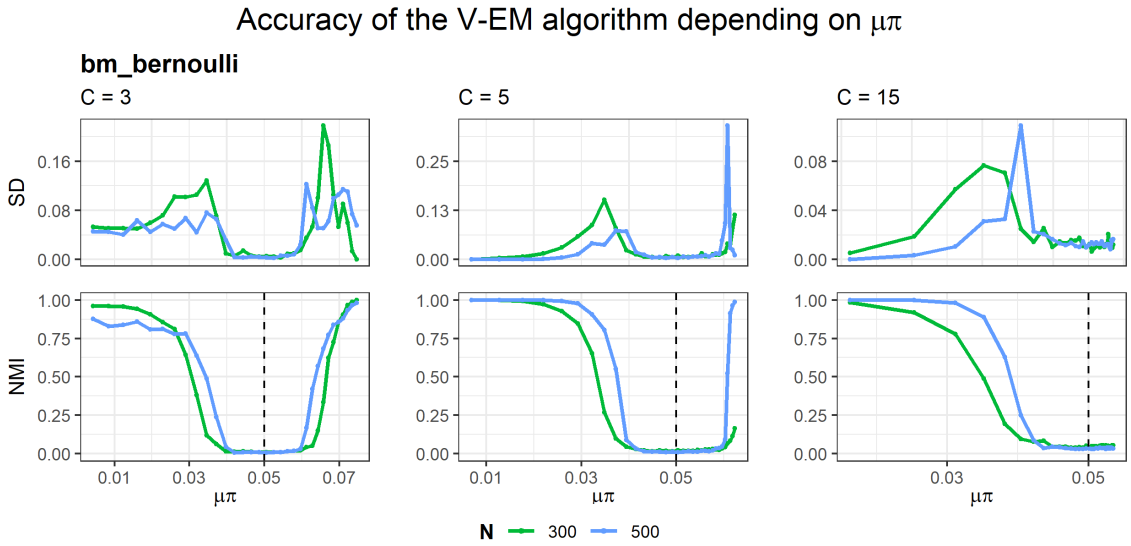

Figure 3: Accuracy Plot 3 - Variational EM: Change of the mean value of NMI and its standard deviation (SD) depending on between connectivity ($\mu\pi$) for different number of nodes. Each point corresponds to calculations on 10 different networks which were simulated with the same parameters. All points to the left of the vertical dashed line correspond to the strong definition of community ($1 - \mu > \mu$). Please notice that the vertical axis for SD has different scale ranges and that networks with n=300 and n=500 were considered due to limitations in computing power.

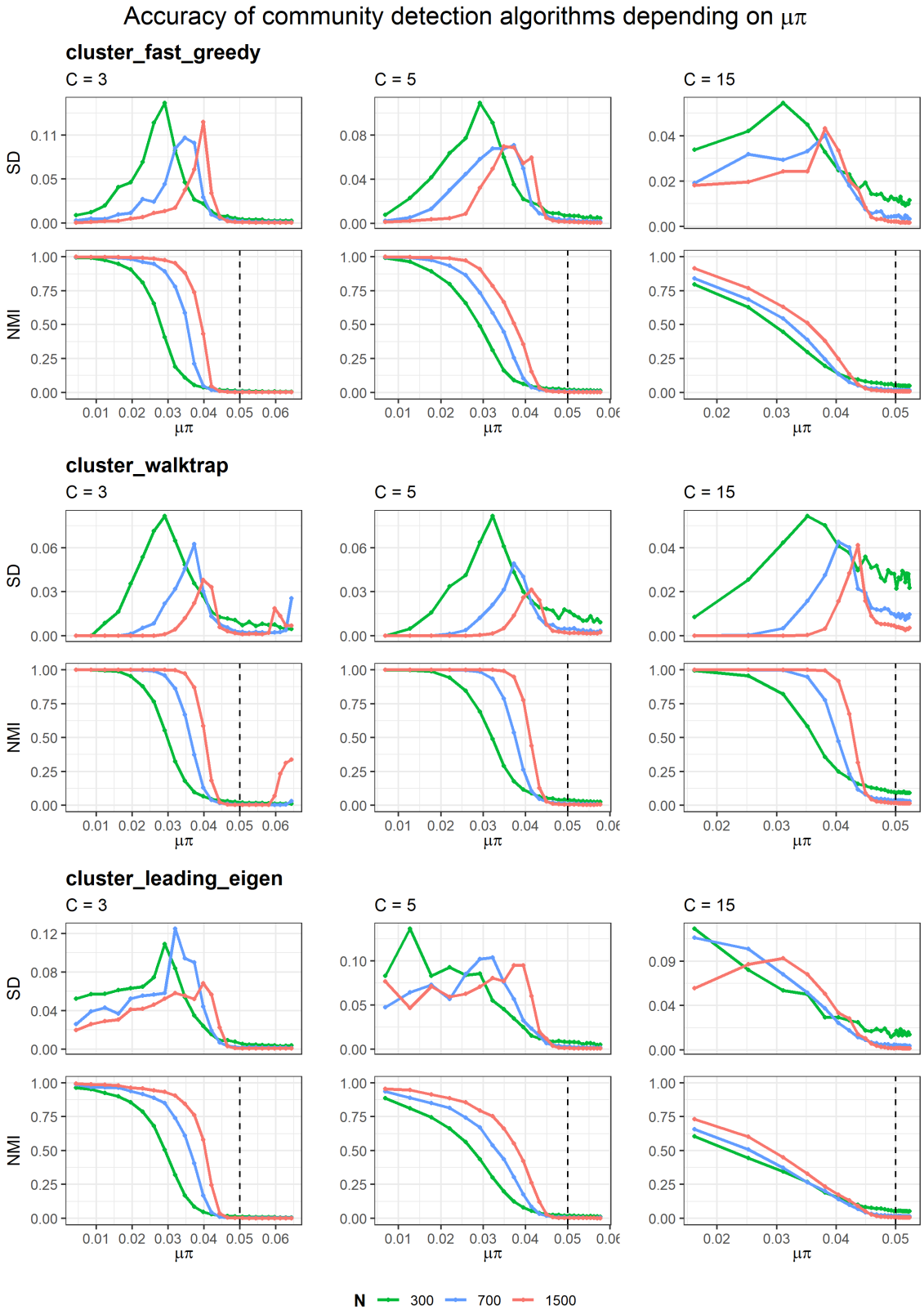

Figure 4: Accuracy Plot 1 - Change of the mean value of NMI and its standard deviation (SD) depending on between connectivity ($\mu\pi$). Each point corresponds to calculations on 50 different networks which were simulated with the same parameters. All points to the left of the vertical dashed line correspond to the strong definition of community ($1-\mu > \mu$). Please notice that the vertical axis for SD has different scale ranges.

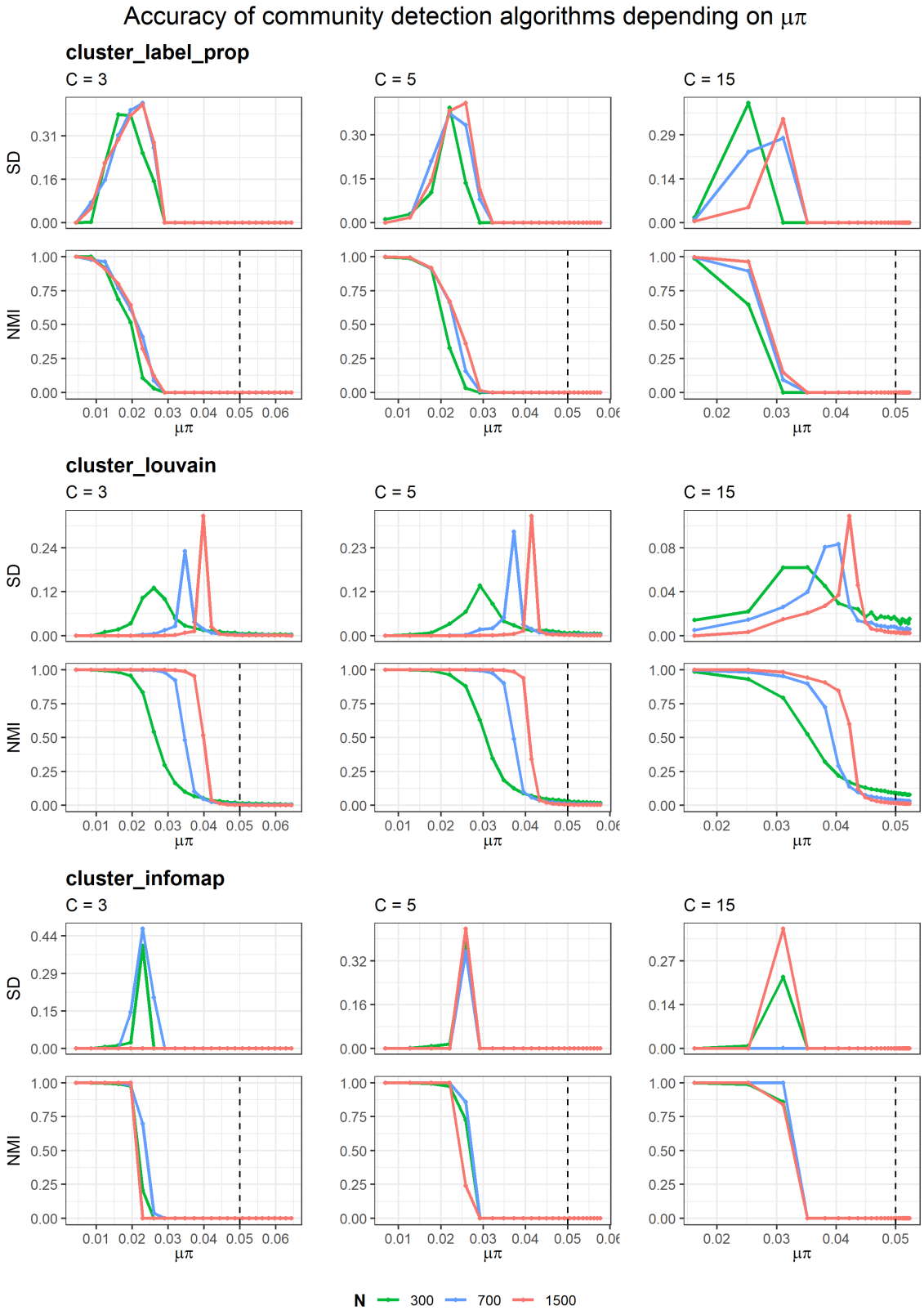

Figure 5: Accuracy Plot 2 - Change of the mean value of NMI and its standard deviation (SD) depending on between connectivity ($\mu\pi$). Each point corresponds to calculations on 50 different networks which were simulated with the same parameters. All points to the left of the vertical dashed line correspond to the strong definition of community ($1-\mu > \mu$). Please notice that the vertical axis for SD has different scale ranges.

4.2.2 Computational effort

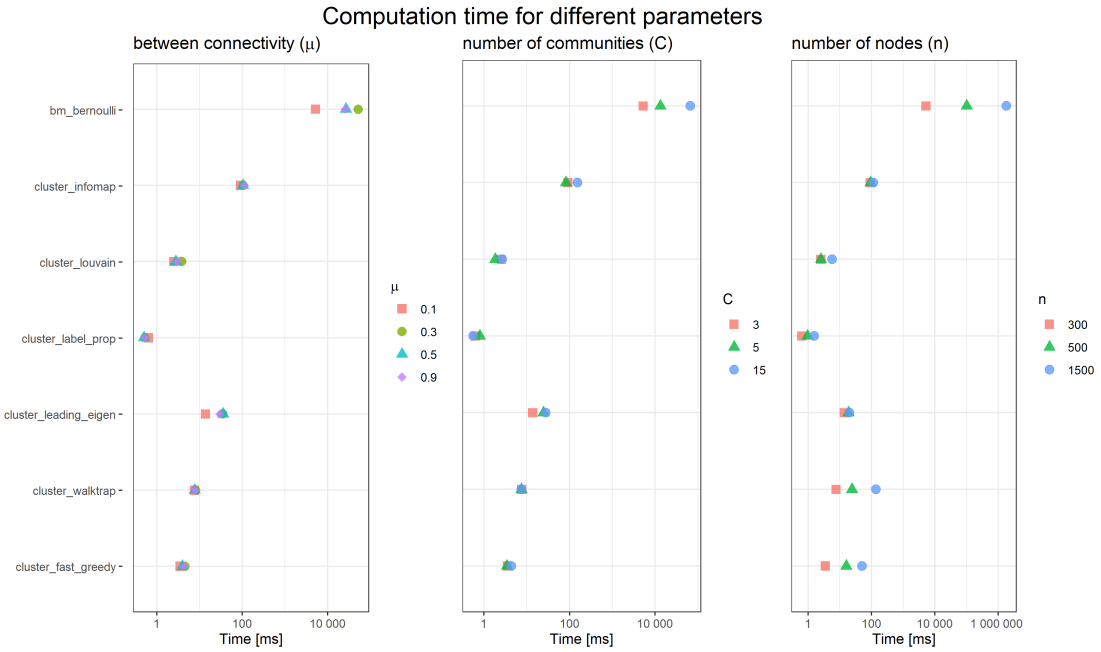

Figure 6: Runtime of different algorithms depending on between connectivity, number of communities and number of nodes. Each point represents the average runtime for the corresponding algorithm on 10 trials. The default values were $\mu = 0.1$, $C = 3$ and $n = 300$ and only one variable varied at a time.

When selecting the algorithm, attention should of course be paid to accuracy, but the computation time must also be taken into account. The runtime has to be considered especially if communities have to be found in very large or many networks. Figure (6) shows the runtimes depending on the between-connectivity (μ), the number of clusters (C) and nodes (n).

One observation is that for most of the algorithms the points corresponding to different μ and C values in figure (6) overlap (plot left and middle). Therefore the probability of edges between clusters (μ) and the number of clusters (C) do not seem to have a general influence on the runtime. Only for the variational EM algorithm `bm_bernoulli` a clear difference between the different μ values is observed. However there is no apparent regularity in how changing μ corresponds to changing runtime. For a network with $\mu = 0.3$ the algorithm takes considerably longer than for the other μ values. It seems plausible that at $\mu = 0.3$, which corresponds to a value of $\mu\pi = 0.035$, the accuracy of the algorithm plummets (see figure (3)) because the V-EM has considerable difficulty calculating the parameters, and therefore prolonging the time needed for the estimators to converge.

Similar findings can be seen when changing the number of communities. Here the V-EM algorithm again has a clear difference in computing time depending on the number of clusters, whereas the other algorithms do not change very much. The increasing length of parameters to be estimated with increasing number of clusters probably causes the longer computation time. Similarly, one can expect the leading eigenvector algorithm to require several steps more until the maximum modularity

is found when the number of clusters increases and accordingly the eigenvalues must be recalculated in each step.

As expected, increasing the number of nodes has the greatest influence on the computation time of the parameters considered. The V-EM algorithm has the most extreme time extension. Here the average calculation on a network with 1500 nodes takes 1.7 million milliseconds (see figure (6)) which corresponds to about 29 minutes. For very large networks, this algorithm is extraordinarily dependent on computation power.

Moreover it is interesting that the Infomap algorithm takes almost the same amount of calculation time regardless of the choice of parameters, as you can see by the points overlapping in figure (6).

5 Conclusion

Heuristics are used in most of the evaluated algorithms in order to arrive at probable statements or practicable solutions with limited knowledge and little time. For example, the walktrap algorithm approximates the class memberships using short random walks and provides very reliable results with very short calculation time.

The variational EM was the only algorithm which tries to estimate the model under which the networks were generated and the only one which could handle networks with higher between than within class probability, but the results for low μ values were considerably worse than with the algorithms using heuristic approaches. However the estimated model might be used to determine the probabilities of forming new edges in a link prediction scenario. Because not only the membership is predicted but all the parameters as well. So depending on ones goals the high computational effort can be justified.

Moreover, it can be observed that all algorithms that use modularity as a measure of quality produce fairly accurate results. These are the fast and greedy, the walktrap, the leading eigenvector and the louvain algorithm listed here. Accordingly, one can assume that modularity is a useful criterion for judging the extent of cluster structure in a network.

Finally, it can be said that the walktrap algorithm provides reliable results for highest μ values while the computation time remains sufficiently short and is a good starting point of a community detection analysis.

References

Abbe, E. (2017). Community detection and stochastic block models: recent developments, *The Journal of Machine Learning Research* **18**(1): 6446–6531.

Biernacki, C., Celeux, G. and Govaert, G. (2000). Assessing a mixture model for clustering with the integrated completed likelihood, *IEEE transactions on pattern analysis and machine intelligence* **22**(7): 719–725.

Blondel, V. D., Guillaume, J.-L., Lambiotte, R. and Lefebvre, E. (2008). Fast unfolding of communities in large networks, *Journal of statistical mechanics: theory and experiment* **2008**(10): P10008.

Clauset, A., Newman, M. E. and Moore, C. (2004). Finding community structure in very large networks, *Physical review E* **70**(6): 066111.

Csardi, G. and Nepusz, T. (2006). The igraph software package for complex network research, *InterJournal* **Complex Systems**: 1695.
URL: *http://igraph.org*

Leger, J.-B. (2016). Blockmodels: A r-package for estimating in latent block model and stochastic block model, with various probability functions, with or without covariates, *arXiv preprint arXiv:1602.07587* .

Li, W. and Schuurmans, D. (2011). Modular community detection in networks, *Twenty-Second International Joint Conference on Artificial Intelligence*.

Manning, C. D., Raghavan, P. and Schütze, H. (2008). *Introduction to Information Retrieval*, Cambridge University Press, New York, NY, USA.

Mariadassou, M., Robin, S., Vacher, C. et al. (2010). Uncovering latent structure in valued graphs: a variational approach, *The Annals of Applied Statistics* **4**(2): 715–742.

Melancon, G. (2006). Just how dense are dense graphs in the real world?: a methodological note, *Proceedings of the 2006 AVI workshop on BEyond time and errors: novel evaluation methods for information visualization*, ACM, pp. 1–7.

Newman, M. E. (2004). Fast algorithm for detecting community structure in networks, *Physical review E* **69**(6): 066133.

Newman, M. E. (2006). Finding community structure in networks using the eigenvectors of matrices, *Physical review E* **74**(3): 036104.

Pons, P. and Latapy, M. (2005). Computing communities in large networks using random walks, *International symposium on computer and information sciences*, Springer, pp. 284–293.

Radicchi, F., Castellano, C., Cecconi, F., Loreto, V. and Parisi, D. (2004). Defining and identifying communities in networks, *Proceedings of the National Academy of Sciences of the United States of America* **101**: 2658–63.

Raghavan, U. N., Albert, R. and Kumara, S. (2007). Near linear time algorithm to detect community structures in large-scale networks, *Physical review E* **76**(3): 036106.

Rosvall, M. and Bergstrom, C. T. (2008). Maps of random walks on complex networks reveal community structure, *Proceedings of the National Academy of Sciences* **105**(4): 1118–1123.

Ward, J. H. (1963). Hierarchical grouping to optimize an objective function, *Journal of the American statistical association* **58**(301): 236–244.

Yang, Z., Algesheimer, R. and Tessone, C. J. (2016). A comparative analysis of community detection algorithms on artificial networks, *Scientific reports* **6**: 30750.

A Appendix

μ	$\mu\pi$	$\frac{1-\mu}{\mu}$	μ	$\mu\pi$	$\frac{1-\mu}{\mu}$	μ	$\mu\pi$	$\frac{1-\mu}{\mu}$	μ	$\mu\pi$	$\frac{1-\mu}{\mu}$
0.03	0.007	32.33	0.27	0.037	2.70	0.51	0.050	0.96	0.75	0.058	0.33
0.06	0.013	15.67	0.3	0.039	2.33	0.54	0.052	0.85	0.78	0.058	0.28
0.09	0.018	10.11	0.33	0.041	2.03	0.57	0.053	0.75	0.81	0.059	0.23
0.12	0.022	7.33	0.36	0.043	1.78	0.6	0.054	0.67	0.84	0.060	0.19
0.15	0.026	5.67	0.39	0.045	1.56	0.63	0.054	0.59	0.87	0.060	0.15
0.18	0.029	4.56	0.42	0.046	1.38	0.66	0.055	0.52	0.9	0.061	0.11
0.21	0.032	3.76	0.45	0.048	1.22	0.69	0.056	0.45	0.93	0.061	0.08
0.24	0.035	3.17	0.48	0.049	1.08	0.72	0.057	0.39	0.96	0.062	0.04

Table 2: Table of all μ and corresponding $\mu\pi$ and $\frac{1-\mu}{\mu}$ values for $C = 5$

μ	$\mu\pi$	$\frac{1-\mu}{\mu}$	μ	$\mu\pi$	$\frac{1-\mu}{\mu}$	μ	$\mu\pi$	$\frac{1-\mu}{\mu}$	μ	$\mu\pi$	$\frac{1-\mu}{\mu}$
0.03	0.016	32.33	0.27	0.045	2.70	0.51	0.050	0.96	0.75	0.052	0.33
0.06	0.025	15.67	0.3	0.046	2.33	0.54	0.050	0.85	0.78	0.053	0.28
0.09	0.031	10.11	0.33	0.047	2.03	0.57	0.051	0.75	0.81	0.053	0.23
0.12	0.035	7.33	0.36	0.048	1.78	0.6	0.051	0.67	0.84	0.053	0.19
0.15	0.038	5.67	0.39	0.048	1.56	0.63	0.051	0.59	0.87	0.053	0.15
0.18	0.040	4.56	0.42	0.049	1.38	0.66	0.052	0.52	0.9	0.053	0.11
0.21	0.042	3.76	0.45	0.049	1.22	0.69	0.052	0.45	0.93	0.053	0.08
0.24	0.044	3.17	0.48	0.050	1.08	0.72	0.052	0.39	0.96	0.053	0.04

Table 3: Table of all μ and corresponding $\mu\pi$ and $\frac{1-\mu}{\mu}$ values for $C = 15$