Divisive Variational Bayesian Algorithms for the Clustering of large and complex Networks

With Applications to Earthquake Networks

Dissertation zur Erlangung des Grades Doktor der
Naturwissenschaften am Fachbereich Mathematik und Informatik
Freie Universität Berlin

vorgelegt von

Christian Tobias Willenbockel

Februar 2017

Betreuer:
Prof. Dr. Christof Schütte
Freie Universität Berlin
Fachbereich Mathematik und Informatik
Arnimallee 2-6
14195 Berlin

Gutachter:
Prof. Dr. Christof Schütte
Prof. Chris Wiggins, PhD (Columbia University, New York City, United States of America)

Tag der Disputation: 5. September 2017

Contents

List of Figures

List of Tables

Introduction

In this thesis, we address the challenge of model based clustering for the analysis of large and complex networks by the examples of earthquake networks. Networks are composed of vertices which can be connected by edges. Edges can be directed or undirected, weighted or unweighted. The edges of the network model the relationships between the vertices.
Our application for network analysis are the earthquake networks introduced by S. Abe and N. Suzuki in [1]. In the case of earthquake networks, the vertices are the cells of the geographic region under consideration and the edges model the successive occurrence of earthquakes in time and space. By construction, earthquake networks are directed and have multiple edges which we consider as weighted.
In order to analyse networks we use the approach of model based clustering according to the Stochastic Block Model (SBM) [51, 52, 108, 97, 35, 84, 73]. Vertices having a similar edge connection profile are considered as a block or cluster of the SBM, which is one possibility to define clusters. These blocks or clusters provide a way to analyse a given network, where each vertex is exclusively assigned to one cluster. There are parameters for each cluster, which govern the existence, directions and weights of each edge in the cluster. Thus, the topology and generation of a network can be analysed with the help of the SBM.
The SBM is a flexible model which can model different types of network structures. At the same time, the SBM is intuitively interpretable and allows link prediction based on the clusters and model parameters. The SBM can model hubs which are densely connected with all or most of the vertices in the network [71]. It can also model sets of densely linked vertices of the network, often called modules in the literature [71]. The existence and importance of hubs in earthquake networks was shown in [1]. It is also possible to model sets of vertices with the SBM which are sparsely linked to each other but are densely linked to another set of vertices. This is called disassortative linkage behaviour [71].
We chose the Poisson version of the SBM [84] as our main model because it allows for the inclusion of edge directions and edge weights which we need for the analysis of earthquake networks.
We want to estimate the Poisson SBM for a given network without prior knowledge of the optimal cluster assignments of the vertices, the model parameters and the optimal number of clusters. For this task we need an inference algorithm and a criterion which shows which is the best of these values and parameters of the SBM.
Direct optimisation of the likelihood of the SBM is computationally intractable for larger networks [35, 84]. In addition, we can not simplify the likelihood because the cluster assignment of each vertex depends on the cluster assignments of all other vertices in the case of the SBM, which is called network interdependency [35, 40]. Thus, we cannot apply the popular Expectation Maximisation (EM) algorithm [36] for infer-

ence of the SBM [35, 40].
As a solution to solve these difficulties of fast deterministic inference of the SBM, the application of the Variational EM (VEM) algorithm [62, 55] was proposed in [35]. The VEM algorithm was expanded to the Variational Bayesian EM (VBEM) algorithm in [15] and proposed as an inference algorithm for the unweighted, undirected Bayesian SBM in [50, 72, 73].
The idea of the VBEM inference algorithm is to introduce variational distributions over the cluster assignments of the vertices and the model parameters. We estimate a variational bound of the likelihood, which is a functional depending on the variational distributions.
The variational bound of the VBEM algorithm, also called free energy [50], penalises the model complexity [15] and the converged free energy is used as a state of the art model selection criterion for the SBM and other mixture models to determine the optimal number of clusters and cluster assignment [111, 19, 50, 73, 8, 74, 116]. The free energy is a functional which is optimised dependent on the variational distributions in an EM like inference algorithm.
The optimisation of the free energy functional of the SBM is a non-convex optimisation problem, which strongly depends on start values [15, 22, 19, 50, 73].
So, we do not only have to find the global optimum of the free energy for the cluster assignments of **one** fixed number of clusters but for all possible numbers of clusters during the optimisation of the free energy of the SBM. We take into account this twofold optimisation problem by combining the inference process for the optimal cluster partition and optimal number of clusters: We start the inference with a cluster partition where all vertices are assigned to one cluster. Then the vertices of this chosen cluster is split into two new clusters and optimised. This split is accepted if the free energy model selection criterion was improved. Our algorithm continues with an optimisation (refinement) for error correction of the cluster assignments of the vertices of a carefully chosen cluster. Such an algorithmic approach is also called a **divisive** algorithm in the literature [56, 109].
Because we split and optimise the vertices in one cluster or block per step, we called our algorithm the Blockloading algorithm. Our Blockloading algorithm and its successors are the first fully divisive algorithms for Variational (Bayesian) EM inference of the (Poisson) SBM [118, 116].
We saw above that we can only identify a global optimum in the case of the SBM if we also found the optimal number of clusters. So, we expect to uncover several local optima with a number of clusters lower than optimal number of clusters with our Blockloading algorithm, before the global optimum can be found.We observed in numerical tests and will give examples that some of these local optima are similar in the sense that with the same number of splits of the 'right' clusters the global optimum is identified. We called these local optima **favourable local optima**.
We focused the algorithmic design of our Blockloading algorithms on efficient splits and optimisation of those clusters where the identification of such favourable local optima, or values near to it, is likely.
We note that a cluster assignment which corresponds to a favourable local optimum provides ideal start values for further splits and optimisation of a divisive algorithm to reach the global optimum.
With the successive splits and refinements of the existing clusters of a cluster assignment which is likely to correspond to a favourable local optimum or state near to it, we want to find one of the favourable local optima for each number of clusters and thus the global optimum. Because we check if the converged free energy was improved after

the optimisation of each cluster (subset), the optimisation process of finding the optimal number of clusters and the optimal cluster assignment is linked, and the optimal number of clusters is determined automatically by our Blockloading algorithms.
To increase the computational speed of our original Blockloading algorithm, we changed the way we store clusters which did not improve the model selection criterion anymore. These efforts resulted in our faster automatic Blockloading and no reset Blockloading algorithms. For our Blockloading++ algorithm, we introduce an even more sophisticated algorithmic design which focuses even better on splitting and optimising those clusters where favourable local optima are likely to be discovered. We propose an even more efficient procedure for the storage of the optimisation results and exploration of the search tree of clusters to efficiently identify favourable local optima and thus a global optimum.Thus, our Blockloading++ algorithm combines all our new algorithmic features and insights for efficient high quality inference of the SBM.
We proposed the Optimal Gap algorithm to predict during the inference process which clusters should be chosen for optimisation with our Blockloading algorithm for the most likely identification of favourable local optima. All our Blockloading algorithms are based on our BlockVB algorithm for optimised VBEM inference of the cluster assignments of subsets of the vertices of the network. For our BlockVB++ algorithm, which builds upon the BlockVB algorithm, we introduced adaptive informative hyperparameters which automatically adapt themselves to the subset of the given network. Our adaptive informative hyperparameters allow for the use of randomly initialised start values for the cluster assignments of the vertices which greatly increases the search space for the start values of the subsets and thus the quality of the results when compared to deterministically initialised start cluster assignments.
Real world networks like earthquake networks often have a large set of sparsely and irregularly connected vertices, which are called irrelevant in the literature [54]. We propose the Poisson SBM with irrelevant vertices (SBMIV) which explicitly models these irrelevant vertices, which are hard to cluster reliably with the normal SBM. For inference of the Poisson SBMIV we propose the Relevance Blockloading algorithm building on our previous work. It allows for a multi level identification of irrelevant vertices. Our Relevance Blockloading algorithm can be employed as a filtering algorithm where the determination of the relevance of vertices is separated from the cluster assignment. It can also be used as an embedded algorithm where the relevance and the cluster assignment of vertices are calculated in the same Expectation Step of our adapted VBEM inference algorithm.
We also introduce a new model selection criterion for the SBMIV, based on the free energy criterion and our new algorithmic framework. Our new relevance informative hyperparameters for the identification of the cluster of the irrelevant vertices make the inference of relevant vertices independent of other algorithms for the initialisation of start values.
We performed numerical tests of our methods on synthetic networks drawn from an SBM and an earthquake network to document the outperformance of other existing methods for inference of the SBM. In those tests, every run of our algorithm performed substantially better than the best result of the comparable algorithms for solving the SBM in terms of computational speed and quality of the results. This includes the Blockmodels package [79] which is the only other state of the art implementation of variational split-merge algorithms for inference of the SBM we are aware of. We demonstrated for an earthquake network that our Relevance Blockloading algorithm for the inference of the SBMIV can reliably identify irrelevant vertices and outperforms inference of the normal SBM with respect to the free energy criterion.

The work in progress on this thesis was published in two technical reports [116, 117]. This thesis contains large parts of these technical reports.

Acknowledgements

My gratitude belongs to all people who supported me during the scientific endeavour of writing this thesis.
First of all, I thank my advisor Christof Schütte for giving me the interesting topic of this thesis and the freedom to pursue my own ideas. His helpful advice, trust, patience and support during the work on this thesis were invaluable for me.
A special thanks to Tim Conrad for general discussions and advice.
Another big thank you to my room mates Sulav Duwal, Iuri Kozhan, Stefan Rüdrich, Maximilian Weiss and Kaveh Pouran Yousef for a pleasant workplace atmosphere and always being there for a talk.
I also thank the Helmholtz-Kolleg GEOSIM for providing an interesting side course program and activities.

This thesis was funded by a GEOSIM fellowship and Dahlem Research School (DRS) grant.

Chapter 1

Background and Overview

In the introduction we presented an overview of our most important contributions and insights of this thesis. We continue with an overview of the most important topics, ordered by chapters, concerning the development of our new algorithms. This includes a review of important existing inference methods and models for clustering methods related to the SBM and a discussion of their limitations. Then, we present our contributions in this thesis in a concise way. We close this chapter with a short summary of the results of the numerical tests where the superiority of our newly introduced methods compared to existing methods is documented.

The Stochastic Block Model (SBM) [51, 52, 108, 97, 35, 84, 73] is a well established and widely used model for the clustering of networks [97, 35, 84, 125, 76]. The topology and generation of a network can be analysed with the help of model based clustering. The results of the SBM can be easily interpreted and link prediction of edges follows easily [97]. Often, only the graph of a network is given without further information. This is also the case for our main application, the earthquake networks. Estimation of the SBM consists of exclusively assigning the vertices of the graph to clusters and inferring the probabilities for the existence of an edge dependent on the inferred cluster membership of the vertices.

The SBM was shown to be a flexible model in [71] which can identify vertices which share a common edge connection profile [73]. The SBM can model hubs which are densely connected with all or most of the vertices in the network [71]. The existence and importance of hubs in earthquake networks was shown in [1]. It can model sets of densely linked vertices of the network, often called modules in the literature [71]. It is also possible to model sets of vertices with the SBM which are sparsely linked to each other but are densely linked to another set vertices, which is called disassortative linkage behaviour [71].

There are many more models related to the SBM which provide different features like multiple memberships for vertices [10, 74] or weighted graphs [84]. One can also model sparsely and irregularly connected vertices as irrelevant [46, 47, 44, 54, 74, 117]. The possibility to integrate otherwise known information was presented in [84, 125, 123, 76]. An SBM which corrects for the degree distribution of vertices was proposed in [65]. The Infinite Relational Model (IRM) [67] can model a potentially infinite number of clusters according to a statistical process contrary to the SBM where the number of clusters is fixed. For all these models we face the same basic challenge of reliable and fast high quality inference we stated at the beginning.

We provide a review of the SBM in **chapter 2**.

The inference method has to infer the unknown (hidden) cluster assignments of the vertices and to estimate the model parameters of the SBM. This is a difficult task because the likelihood of the SBM requires the summation over all possible cluster partitions, which is intractable for bigger networks [35].
Moreover, the cluster assignment of each vertex depends on the cluster assignments of all other vertices [35, 40] and thus the 'The dependency graph of $\boldsymbol{Z}$ [...]' [101] (the cluster assignments) ' [...] is a clique.' [101] (fully connected graph). This difficulty prevents the use of the well established Expectation Maximisation (EM) algorithm [36], [35, 40].
As one way to overcome this difficulties for the inference of the SBM, the use of the Variational Expectation Maximisation (VEM) algorithm of [62, 55] was proposed in [35] for the simple (undirected, unweighted) SBM (Bernoulli SBM) and later adapted to the directed and weighted SBM with Poisson distributions (Poisson SBM) in [84].
The VEM algorithm was expanded to the Bayesian case with the Variational Bayesian Expectation Maximisation (VBEM) algorithm in [15]. The VBEM algorithm provides a tractable variational bound of the intractable likelihood which can be also used as a state of the art model selection criterion [15] for the SBM [50, 71, 73]. This variational bound is optimised dependent on a factorised approximating distribution over the hidden variables [15].
The VBEM algorithm is convex with respect to each factor of the variational distribution which leads to monotonic improvement of the variational bound and sure convergence of the algorithm [19, 22]. It was noted in [15, 111, 50] among others, that the result of the VBEM algorithm can depend on the starting values. This is also true in the case of the Stochastic Block Model where such examples can easily be found. So the quality of the result of the VBEM algorithm depends on the choice of the initial cluster assignments of the vertices and the model parameters [50, 73]. Therefore only convergence to a local optimum is guaranteed by the VBEM algorithm for the SBM, which shows that it is a non-convex optimisation problem. This is also true for VBEM inference for other mixture models [111]. Consequently, it is recommended in the literature [111, 50, 73] to run the VBEM algorithm with different start values and to choose the best result of these calculations in the hope of finding the global optimum or a solution near to it. We review the general properties of the EM, VEM and VBEM algorithm in **chapter 3**.

A VBEM algorithm for a restricted version of the Bernoulli SBM was proposed in [50] and for the Bernoulli SBM in [71, 73]. The theory of a general VBEM algorithm for the Bayesian SBM, restricted to distributions from the exponential family, which also includes weighted distributions was presented in [8]. An implementation for some weighted distributions, like the Poisson distribution among others, was provided in [7] as an accompanying software to [8].
We present our derivation and propositions of the VBEM algorithm for the Bayesian Poisson SBM following the more general approach of [19, 71] and document the whole algorithm with all update equations in **chapter 4**.

The problem of convergence to a local optimum of the VBEM algorithm and other variational methods is also linked to the problem of model selection of the SBM. Model selection for the SBM consists of finding the optimal number of clusters, K^*, together with the optimal cluster assignments or cluster partition of the vertices and the model parameters according to the chosen model selection criterion.

Model selection criteria like the Akaike Information Criterion (AIC) [11, 19] or the Bayesian Information Criterion (BIC) [106, 19] cannot be used for model selection of the SBM because of the computationally intractable likelihood of the SBM as we saw above [84, 73]. We have to use instead a model selection criteria suitable for the SBM like the converged variational bound of VBEM (converged free energy / ILvb) [50, 73], the Integrated Complete Likelihood Criterion (ICL) [18, 35] or the exact Integrated Complete Likelihood Criterion (ICL_{ex}) [29].
For several years until 2014, the following approach, which is also called the batch approach in the literature [125], was used to apply the VEM and VBEM algorithms for inference of the SBM with an unknown number of clusters [35, 50, 84, 71, 74]: To find the best model and the optimal number of clusters of the SBM, K^*, using a variational method, the VBEM algorithm is initialised for different numbers of clusters, K, **and** for different initial start values for the cluster assignments of the vertices, $\boldsymbol{Q}^{(start)}$, to take into account that the VBEM algorithm converges only to a local optimum.
For all values of the number of clusters, K, under consideration, the converged variational bound, also called converged free energy [38, 50], of the SBM ([50, 73]) or another suitable model selection criterion for the SBM is calculated. Then the result with the optimal converged free energy and its corresponding cluster partition matrix, $\boldsymbol{Q}^*$, and number of clusters, K^*, is chosen.
We review the batch algorithm and important state of the art model selection criteria for the Poisson SBM in **chapter 6**.

In the algorithmic setup of the batch algorithm, most of the computational time is used solely for determining the optimal number of clusters and only a fraction of the time is invested into calculating the optimal cluster assignments of the vertices, $\boldsymbol{Q}^*$. In particular, there is no algorithmic procedure for utilising the information contained in already calculated cluster partition matrices of the vertices for different numbers of clusters. These drawbacks of the batch approach and the convergence to only a local optimum limit the complexity and size of graphs variational methods can handle with meaningful results.
After research on the VBEM inference of the SBM and the Overlapping Stochastic Block Model (OSBM), it was even discussed in the outlook of [71] that: 'In future work, we will investigate Markov chain Monte Carlo techniques as alternative approaches to estimate the posterior distribution of OSBM. [...] In particular, they appear to be less dependent on the initialisation of the $\boldsymbol{\tau}$ matrix than variational approaches.' [71], where the $\boldsymbol{\tau}$ matrix denotes the start cluster assignments of the vertices.
Following this suggestion, we have a closer look at Markov chain Monte Carlo (MCMC) based methods for the inference of models like the SBM. The use of traditional Gibbs sampling inference, which is a MCMC technique, for the SBM was proposed in [108] and [97]. It converges theoretically to the true posterior and statistics [62, 97] but convergence is slow in practice and may require a very large number of samples [62]. Therefore the Gibbs sampling inference of [97] is only suitable for networks of up to two hundred vertices [97, 35, 73].
In the seminal paper [57] (later published as [58]) the application of traditional Gibbs sampling to a mixture model nearly similar to the SBM was addressed. There it was noted that: 'Traditional Markov chain Monte Carlo methods for Bayesian mixture models, such as Gibbs sampling, can become trapped in isolated modes corresponding to an inappropriate clustering of data points', [57]. So, the application of traditional Gibbs sampling inference to a mixture model basically similar to the SBM has in practice comparable limitations regarding the quality of the results as the application of

variational methods with the batch algorithm to the SBM.
As a solution the split merge Gibbs sampler was proposed in [57] which replaced the traditional approach for Gibbs sampling. In the split-merge Gibbs sampler the inference consists of splitting a randomly chosen existing cluster into two new clusters (split move) or merging two randomly chosen existing clusters into a new one (merge move). The split or merge move is then accepted with a certain probability. It was shown in numerical tests that the split-merge Gibbs sampler outperforms the traditional approach in terms of quality of the results [57]. This results lead to further research in the field of split-merge Gibbs sampling and is until today a field of ongoing research [33, 59, 26, 113].
The split-merge sampler was used as the the basis of an algorithm for inference of the IRM model in [67] which can at the same time infer the optimal number of clusters and the cluster assignments [67]. The split-merge sampler of [57] lead to numerous follow up papers which is until today the basis for the inference of the IRM for example in [67, 46, 68, 92, 105, 12].
Traditional Gibbs sampling has usually slower convergence than variational methods like VEM and VBEM [62] and does not scale for large data sets [111]. In addition, it is more difficult to determine convergence of Gibbs sampling than for variational methods [62]. This is the reason why variational inference methods were explored as a faster alternative for MCMC methods in the first place [62, 15, 111, 35, 50, 71].
It was already remarked in [57] that the random selection of the clusters to split or merge together with dependence on the acceptance probability for the split or merge move are limitations of their split–merge sampler.
The improvements of the split–merge Gibbs sampler compared to the traditional Gibbs sampler for models very similar to the SBM show that split-merge concepts are an option for the improvement of variational methods, too.
Before we come to split and merge algorithms for variational inference of the SBM, we have a look at other algorithms, which also use split and merge moves for clustering. Now, we review split and merge concepts for clustering which are in fact even older than the split–merge Gibbs sampler of [57] and date at least back to the 1980s [56].
An algorithm which uses split moves is the bisecting K-means algorithm (see e.g. [109]). The usual inference algorithm for K-means consists of an EM like algorithm [80, 19] that also converges to a local optimum [19]. Contrary to Gibbs sampling the K-means algorithm is a deterministic algorithm like variational methods.
In the bisecting K-means algorithm all vertices are placed in one cluster. Then a cluster is split in each iteration. The best split is chosen, based on a similarity criterion. The splits of the clusters continue until the desired number of clusters is reached. An optional application of the normal K-means algorithm to the existing cluster assignments for refinement is possible.
Such an algorithm is called a divisive algorithm in the literature [56]. The bisecting K-means algorithm was shown to outperform the batch version of K-means [109].
There are also no merge moves like in the split–merge sampler.
Another divisive algorithm for the application of K-means for clustering is the X-means algorithm [99]. Here the split move is performed for all clusters simultaneously and a model selection criterion is used to determine the optimal number of the clusters. Then the normal K-means algorithm is applied to refine the resulting cluster assignments.
The opposite concept are agglomerative clustering algorithms, where all vertices or small sets of vertices are placed in a cluster [56]. Then these small clusters are merged and the merge step is accepted if it improves a model selection criterion. Of course, both steps can be used successively as we have seen in the case of split-merge Gibbs

sampling. Another example, where such split and merge and refine moves were combined is [32]. The bisecting K-means algorithm was shown in [109] to outperform agglomerative algorithms of e.g. [56] for documents clustering both in speed and quality of the results.

An algorithm which uses agglomerative clustering for the Bernoulli SBM combined with greedy schemes [95, 21, 29], is the the greedyICL proposed in [29]. GreedyICL maximises the ICL_{ex} model selection criterion, also proposed in [29], by calculating the optimal cluster assignment of each vertex with the other vertices hold fixed. After convergence of the greedy scheme, the cluster partition is optimised according to an agglomerative scheme. GreedyICL inference converges to a local optimum and depends on start values [29]. Contrary to the batch algorithm, greedyICL determines the optimal number of clusters during the optimisation of the cluster assignments. It was shown to outperform, among others, the batch variational algorithms of [35] and [73].

Another agglomerative algorithm was proposed for the fine gained Stochastic Block Model [121], which is a variant of the Stochastic Block Model, is the BLOS algorithm. The BLOS algorithms is based on the EM algorithm and is therefore no variational method like the VEM or VBEM algorithms. The EM can be applied because of differences of the fine gained SBM compared to the SBM [121]. Like in other greedy or agglomerative clustering the vertices are placed in a number of clusters which by far exceeds the expected number of optimal clusters. Then the inference, based on merge moves and optimisation, restricted to the clusters, is performed until convergence [121]. The BLOS algorithm was shown to outperform traditional variational batch algorithms like the VEM algorithm [35] or the VBEM algorithms of [50, 73].

Online clustering algorithms for the SBM based on the Classification EM (CEM) and VEM algorithm were proposed in [124, 125, 123]. They can also be applied to a given network as an offline algorithm for a considerable speed up of the computational time at the cost of quality of the results, when compared to the batch algorithm [125, 123]. The number of clusters remains fixed throughout the inference process of the online clustering algorithm like in the case of the batch algorithm. Online clustering is based on inference of subsets of the data.

Another subset based inference algorithm, the Stochastic Variational Inference (SVI) algorithm [48], which is based on Stochastic Optimisation [100], was proposed first for the Mixed-Membership Stochastic Block Model (MMSB) in [42] and then for the Bernoulli SBM [39]. The SVI algorithm, which is used as an extension to VBEM, converges to a local optimum in the case of the MMSB and SBM [48, 41] like normal variational methods. It was shown with numerical tests that the SVI lead to a considerable improvement of the computational speed and quality of the results in case of the MMSB [42, 41]. In the case of the SBM, there was a clear increase of the computational speed when compared to the batch algorithm but an overall similar quality of the results, save for a very large synthetic network where the batch algorithm failed to converge [39]. The start cluster assignments for the application of the SVI to the SBM are initialised with the spectral clustering algorithm in [39]. The number of clusters is kept fixed during the inference process of the SVI algorithm like in the case of the batch algorithm [39].

We conclude that all approaches which lead to any improvement over the batch ap-

proach we have mentioned so far restrict the inference process to subsets of the data. Now, we have a look at improved variational methods for the SBM which at least offer split or merge options.

Now, we come to algorithms for variational inference of the SBM which use split and merge moves. We are only aware of two approaches for this kind of algorithm: the Wmixnet package [75] and its successor the Blockmodels package [79] and our own Blockloading algorithm and its variants [118, 116, 117].
The Wmixnet software of [75] offers an inference algorithm for different types of the SBM (e.g. weighted or with covariates) with a split-merge option based on the VEM algorithm. As the first step of the inference Wmixnet still uses the batch VEM algorithm for fixed numbers of clusters. As a second step, it offers a 'smoothing' option which includes an 'ascend' mode for splitting clusters of an existing partition and a 'descend' mode in which existing clusters are merged. This smoothing option is proposed to check if the algorithm can escape out of 'bad' local optima.
So the reuse of the existing partition during inference process is still limited when compared to other split merge algorithms existing at the time of [75] like bisecting K-means [109] or the split-merge Gibbs sampler of [57].
The VEM algorithm which is used in the Wmixnet package is known to be dependent on a second algorithm to initialise the start cluster assignments [35, 84, 122]. Therefore Wmixnet relies mainly on the spectral clustering algorithm of [102] to initialise the start values.
Together with the VEM algorithm the ICL criterion [18, 35] is used for model selection. It was noted by [84] that the ICL criterion for networks is known to have a tendency '[...] to underestimate the number of classes in the case of small graphs, [...]' ([73]) because it is an asymptotic model selection criterion.

The first fully divisive and subset optimisation inference algorithm for the SBM, based on the VBEM algorithm, is to the best of our knowledge, our Blockloading algorithm [118]. The Blockloading algorithm reuses the existing cluster partition during the inference process. We also proposed a version for weighted networks among other features in [116]. We refer to the contributions section for a more detailed description.

The successor of the Wmixnet package [75] is the Blockmodels package, proposed in [79]. It offers inference solutions for different versions of the SBM like weighted SBMs or the inclusion of covariates. Contrary to the Wmixnet package it re-uses already calculated partitions like in [109, 57, 118, 116].
The split steps of the clusters are separated from the merge steps in the Blockmodels package [79] unlike for example in [57, 118, 116] where the steps are integrated.
Like the Wmixnet package, the Blockmodels package relies on the VEM algorithm together with the ICL model selection criterion. The Blockmodels package also mainly uses spectral clustering for the start cluster assignments.

We provide a review of the most important subset based and split–merge concepts and methods in **chapter 7**.

Before we proceed with our contributions, we remark that we provide a more detailed introduction of our Stochastic Block Model with irrelevant vertices (SBMIV) and our original inference algorithms for the SBMIV in **chapters 11, 12 and 13**.

Our Contributions We sum up our main results and contributions of this thesis, ordered by chapters:

• Complete and thorough presentation including all proofs and discussion of the VBEM algorithm for the Bayesian Poisson Stochastic Block Model in **chapter 4**. The use of the Poisson distribution allows us to cluster weighted networks.

• Our new BlockVB algorithm provides an optimised way to calculate the cluster assignments of subsets of the vertices of the network with respect to the free energy. We present it in **chapter 4**.

• Our original fully Bayesian BlockVB++ algorithm which allows optimisation with randomly initialised cluster assignments of the vertices due to our newly introduced adaptive informative hyperparameters in **chapter 5**. So, there is no need for a second algorithm like Spectral Clustering to initialise the start cluster assignments of the vertices contrary to other state of the art variational methods. Thus we can use randomly initialised start cluster assignments of the vertices to explore a bigger space of possibilities for the identification of a global optimum.

• We propose our Blockloading algorithm which was, to the best of our knowledge, the first divisive algorithm for the SBM based on VBEM inference for automatic identification of the number of clusters, together with integrated error correction, based on our BlockVB algorithm. We present it in **chapter 8**.
The Blockloading algorithm was designed for a high quality inference of large and complicated networks and earthquake networks according to the SBM. Our algorithmic design of the Blockloading algorithms is focused on avoiding to get trapped in unfavourable (bad) local optima during the inference process. It was designed from the beginning with variational methods in mind. It greatly improves the quality of the results and the computational speed compared to the traditional batch algorithm and, to the best of our knowledge, all other implementations of variational inference of the SBM and thus allows for high quality inference on earthquake networks with variational methods.
The existing cluster assignment is reused as start values during the inference process.

• In **chapter 8**, we discuss which local optima are suitable start values and enable a divisive algorithm like our Blockloading algorithm to identify a possible global optimum. We give our original definition for such local optima and call them 'favourable local optima' contrary to the 'unfavourable local optima' where a divisive algorithm can get stuck in. Together with our new definition of favourable local optima we define favourable split and merge moves of a divisive algorithm.
We will see that our Blockloading algorithm ensures by its design that there is a high chance for identification of all favourable local optima and thus a global optimum.
We will see though, that there is room for improvement of the efficiency of our Blockloading algorithm.

• We discuss possibilities for the improvement of our Blockloading and propose two other algorithms, the automatic Blockloading and the no reset Blockloading algorithm

in **chapter 8**.
These efforts lead to our even more sophisticated Blockloading++ algorithm which greatly improves upon the computational speed and efficiency compared to our previously proposed Blockloading based algorithms. The new algorithmic design of our Blockloading++ algorithm ensures even better that the inference process focuses on those vertices and clusters where it is most likely that an improvement of the model selection criterion can be achieved. At the same time our algorithmic design of our Blockloading++ algorithm ensures that unfavourable local optima are avoided during the inference process in the first place, so that the algorithm does not have to escape out of them.

• In **chapter 9**, we review the possibility for initialising start values for VBEM inference of the SBM. We review the Largest Gaps (LG) algorithm of [27] and show how to use it for the initialisation of start values, especially for very large networks. Our discussion of the LG algorithm leads to the introduction of our Optimal Gap (OG) algorithm which was designed to be less susceptible to outliers than the LG algorithm at a slightly increased computational cost. The OG algorithm, which builds on the same principles as the LG algorithm, is especially built for the identification of possible favourable splits of subsets of a network. This feature is an add-on to the possibilities our Blockloading framework offers.

• With numerical tests on synthetic - and earthquake networks in **chapter 10** we show that our Blockloading and Blockloading++ algorithms achieve quality of the results and computational speed previously unattainable by other state of the art variational methods we tested for the inference of the SBM.
We compare with the reproduction of a synthetic test of [29] of a network with 10.000 vertices, that our Blockloading algorithm clearly outperforms the greedyICL algorithm of [29] with a perfect result which is not achieved by the other algorithms tested: the colsbm algorithm of [85], the vbmod algorithm of [50] and the spectral clustering version of [107].
We also show that the the implementation of the Weighted Stochastic Blockmodel [8] in the WSBM 1.2 package of [7] fails for the inference of this test of this large and complicated network. We show that the Blockmodels package from [78] (documentation also in [79]) is not able to return results for such large networks due to too slow speed.

• We show with a numerical test of a weighted and directed network generated according to a Poisson SBM that our Blockloading algorithm achieve perfect results with respect to the ground truth, too.

• We show with a numerical test of the Southern California earthquake network that all our Blockloading(++) algorithms outperform the Blockmodels package of [79], the WSBM 1.2 package of [8, 9] and the traditional batch VBEM algorithm for the Poisson SBM.

• We show in the same test of the Southern California Earthquake Network that our Blockloading(++) algorithms provide fast and reliable high quality results with respect to the model selection criterion. Therefore, our original algorithms are the only methods, we are aware of, which provide this kind of inference for the earthquake network. Our new methods are the first to provide insights in the network topology and network

generation process according to a (weighted) SBM.

• In **chapter 12** we present a special SBM with irrelevant vertices (SBMIV), which explicitly models sparsely and irregularly connected vertices of weighted networks. We derive a VBEM inference algorithm for the SBMIV and present all update equations and proofs for it in **chapter 13**.

• Building upon our Blockloading and Blockloading++ algorithm we propose our Relevance Blockloading algorithm in **chapter 13**. Our Relevance Blockloading algorithm is designed for the inference of the SBMIV. The inference with our Relevance Blockloading algorithm includes early identification and exclusion of the inference process of noisy and sparsely connected vertices. These vertices could otherwise disturb the inference process.

• We show with a numerical test of the Southern California Earthquake Network in **chapter 14**, that our Relevance Blockloading algorithm successfully identifies sparsely and irregularly connected vertices. This test also shows that inference of our weighted SBMIV with our Relevance Blockloading outperforms inference of our Blockloading algorithm of the Poisson SBM with respect to the free energy model selection criterion.

Chapter 2

Review of the Stochastic Block Model

The Stochastic Block Model (SBM) originates from the field of mathematical sociology [82, 51, 108, 97]. The SBM can be traced back to [82], which was a deterministic version of the block model.
Here, we will review the modern version of the Stochastic Block Model (SBM), introduced in [108, 97, 35], which is a flexible model for the generation of networks for model based clustering.
The SBM is now a widely adapted model which is also used for scientific applications outside the field of mathematical sociology for model based clustering, like for the analysis of protein–protein interaction networks [35, 71], ecological networks [84, 75] or the world wide web [123].
We proposed the application of the SBM for the clustering of the earthquake networks, introduced in [1], in [118, 116]. We will test the clustering of these earthquake networks according to the SBM with our new inference methods in chapter 10.
Our improved inference algorithms of the later chapters will also lead us to the introduction of our new version of the SBM, the Stochastic Block Model with irrelevant vertices (SBMIV) [117], in chapters 11, 12 and 13.
We recall from chapter 1, that it was shown in [71] that the SBM covers a wide area of possible connection behaviours in a network, like hubs, assortative and dis-assortative edge connections and combinations of these. The need of developing viable inference methods for existing versions of the SBM for complicated and large networks was also emphasised in recent publications like [29, 42].
Our review of the SBM follows the presentation in [108, 35, 84, 73]. We start our review with a presentation of the frequentist version of the SBM.
Then the generalisation to the Bayesian version of the SBM [97, 50, 73, 96] is presented in section 2.2 where conjugate prior distributions for the model parameters are introduced.

2.1 Generation of the Stochastic Block Model

A graph or network $G = (V,E)$ consists of a set V of N vertices or nodes and a set of (directed) edges E connecting the vertices. Throughout this thesis, the terms graph and network are used synonymously. The edges connecting the vertices are given by an

adjacency matrix $\boldsymbol{A}$. If there is an edge from vertex i to vertex j, it is $A_{ij} = 1$. If there is no edge from vertex i to vertex j, it is $A_{ij} = 0$.
If the network is directed, A_{ij} and A_{ji} have to be considered differently. Thus, it is possible that $A_{ij} = 0$ but $A_{ji} = 1$ holds, so that $A_{ij} \neq A_{ji}$ follows. In the case of weighted graphs, it holds that $A_{ij} = w_{ij}$, $w_{ij} \in (0,1,2,\dots)$, if there is an edge from i to j. It was remarked in [123] that in [97] dyads, (A_{ij}, A_{ji}), were used to model directed and undirected edges contrary to the edge notation A_{ij} or A_{ji} used above, which was introduced in [35]. In this paper we will consider directed graphs unless otherwise stated.
The following Stochastic Block Model (SBM) was introduced in [97, 35, 84] as an algorithm for generating graphs. We assume that $\boldsymbol{A}$ was generated by the SBM.
The SBM assigns the vertices V of the graph depending on their connection probability patterns to clusters.
The SBM consists of K clusters. To each vertex i, the SBM assigns a unique cluster membership. A vertex belongs to cluster k with probability π_k with $\sum_{k=1}^{K} \pi_k = 1$.
The cluster membership is given by the random variable $\boldsymbol{Z}_i \in \mathbb{R}^{1\times K}$, with $Z_{ik} = 1$ if i is an element of cluster k and $Z_{ik} = 0$ otherwise. $\boldsymbol{Z}$ is the $N \times K$ cluster indicator matrix with matrix rows $\boldsymbol{Z}_i$ for $i \in \{1,\dots,N\}$. This cluster indicator matrix is also called *cluster partition matrix*. An edge exists within each cluster k with the probability θ_{kk} and between the clusters k and l with the probability θ_{kl}. So, the **SBM** is generated in the following way [50, 116]:

(i) Roll a k – sided dice which has probability $p(i \in k | Z_{ik} = 1) = \pi_k$ for side k for each vertex i, to determine the unique cluster membership of the vertex.
(iia) Flip a coin for each pair of vertices. With probability $\theta_{kl} = p(A_{ij} \,|\, Z_{ik}Z_{jl} = 1)$ there is an edge from vertex i to j with $i \in k$ and $j \in l$ and with probability $1 - \theta_{kl}$ there is no edge.

So, the SBM consists of two distributions, the distribution of the cluster assignments:

$$p(\boldsymbol{Z}|\boldsymbol{\pi}) = \prod_{i=1}^{N}\prod_{k=1}^{K} \pi_k^{Z_{ik}}, \tag{2.1}$$

and the distribution for the edge existence dependent on the cluster assignments, $\boldsymbol{Z}$, and probabilities for the edge existence, $\boldsymbol{\theta}$:

$$p(\boldsymbol{A}|\boldsymbol{Z},\boldsymbol{\theta}) = \prod_{i\neq j}^{N}\prod_{k,l}^{K} \left(\theta_{kl}^{A_{ij}}(1-\theta_{kl})^{(1-A_{ij})}\right)^{Z_{ik}Z_{jl}}. \tag{2.2}$$

Therefore we can sum up the joint probability distribution of the SBM by:

$$\begin{aligned} p(\boldsymbol{A},\boldsymbol{Z},\boldsymbol{\pi},\boldsymbol{\theta}) &= p(\boldsymbol{A}|\boldsymbol{Z},\boldsymbol{\theta})\, p(\boldsymbol{Z}|\boldsymbol{\pi}) \\ &= \prod_{i\neq j}^{N}\prod_{k,l}^{K} \left(\theta_{kl}^{A_{ij}}(1-\theta_{kl})^{(1-A_{ij})}\right)^{Z_{ik}Z_{jl}} \prod_{i=1}^{N}\prod_{k=1}^{K} \pi_k^{Z_{ik}}. \end{aligned} \tag{2.3}$$

The results of the clustering are easily interpretable. The prediction of new edges with this model follows naturally from the estimated parameters. Variants of the SBM for directed and weighted graphs exist [84, 8, 116]. For example it is possible to replace the Bernoulli distribution in (12.2) with a Poisson distribution [84]:

$$f(A_{ij};\lambda_{kl}) = \frac{\lambda_{kl}^{A_{ij}}}{A_{ij}!}\exp(-\lambda_{kl}). \tag{2.4}$$

Then we can replace **(iia)** with:
(iib) Draw a realization from $f(\cdot;\lambda_{kl})$ for the edge A_{ij} from vertex i to vertex j, with $i \in k$ and $j \in l$. Then, the joint probability distribution for directed and weighted graphs is then:

$$p(\boldsymbol{A},\boldsymbol{Z},\boldsymbol{\pi},\boldsymbol{\lambda}) = \prod_{i\neq j}^{N}\prod_{k,l}^{K} f(A_{ij};\lambda_{kl})^{Z_{ik}Z_{jl}} \prod_{i=1}^{N}\prod_{k=1}^{K} \pi_k^{Z_{ik}}. \tag{2.5}$$

Using the Poisson distribution also works for unweighted graphs. In the following, we call this SBM [84] the Poisson SBM contrary to the Bernoulli SBM of [108, 35]. We sum up that the Poisson SBM can model all important networks types: directed, undirected, weighted and unweighted networks. It is also possible to use other distributions than the Poisson distribution for $f(\cdot;\lambda_{kl})$, like the Gaussian distribution [84]. Henceforward, we will call $f(\cdot;\cdot)$ the *edge existence distribution*.

Modelling Loops or Self edges A *loop* or self–edge is an edge which connects the same edge with itself. We will see in chapter 10, that the self–edges follow from the construction of earthquake networks to model direct after shocks of earthquakes [1]. The SBM we reviewed above does not allow the use of self–edges. For a way to add self–edges to the SBM we refer to [123].
We discuss in chapter 10 why we do not use the self edges of the earthquake network. Self-edges are not used in e.g. [97, 35, 50, 84, 73, 40, 74].

2.2 The Bayesian Stochastic Block Model

The Bayesian version of the Bernoulli SBM was proposed in [108, 97]. For the Bernoulli SBM we follow the version of [72, 71, 73].
It is possible to set *prior distributions* over the parameters of the SBM which yields the *Bayesian SBM* [97]. Prior distributions are required for two important inference algorithms of the SBM: Gibbs sampling [97] and the Variational Bayesian Expectation Maximisation Algorithm (VBEM) [50, 73]. An inference algorithm for the SBM is used to infer the unknown SBM for a given network. Throughout this thesis we will use the VBEM algorithm for the Poisson SBM, and in some instances the Bernoulli SBM, as the basis for our contributions in this thesis. We will provide this inference algorithm in detail in chapter 4.
A Dirichlet distribution, $\mathrm{Dir}(\boldsymbol{\pi};\boldsymbol{\delta}^0)$, was set as the conjugate prior distribution to the parameters $\boldsymbol{\pi}$ of the multinomial distribution and a Beta distribution, $\mathrm{Beta}(\theta_{kl};\alpha^0_{kl},\beta^0_{kl})$, was set as the conjugate prior distribution on the edge existence parameters, θ_{kl}, of the Bernoulli distribution [97]. The parameters of the conjugate prior distributions, $\boldsymbol{\delta}^0$,$\boldsymbol{\alpha}^0$ and $\boldsymbol{\beta}^0$ are called the *hyperparameters* [50, 72, 73].
Therefore the prior distributions for the Bernoulli SBM are given by [73]:

$$\boldsymbol{\pi} \sim p\left(\boldsymbol{\pi} \,|\, \boldsymbol{\delta}^0 = (\delta^0_1,\ldots,\delta^0_K)\right) = \mathrm{Dir}(\boldsymbol{\pi};\boldsymbol{\delta}^0), \tag{2.6}$$

$$\theta_{kl} \sim p\left(\theta_{kl} \,|\, \eta^0_{kl},\zeta^0_{kl}\right) = \mathrm{Beta}(\theta_{kl};\eta^0_{kl},\zeta^0_{kl}),\ \forall k,l. \tag{2.7}$$

To find conjugate prior distributions for the Bayesian Poisson SBM, we use the fact that the Gamma distribution is the conjugate prior distribution of the Poisson distribution [19]. Conjugate prior Gamma distributions for the Poisson edge existence distributions

were also used in the case of the Infinite Relational Model (IRM), which is closely related to the SBM, in [96] and in the implementation of the Bayesian Poisson SBM in [7].
The conjugate Gamma distribution for the edge existence rates $\lambda_{kl}\ \forall(k,l) \in \{1,\dots,K\}^2$, is now given by:

$$\lambda_{kl} \sim p(\lambda_{kl} \mid \alpha^0_{kl}, \beta^0_{kl}) = \text{Gamma}\left(\lambda_{kl}; \alpha^0_{kl}, \beta^0_{kl}\right),\ \forall k,l. \tag{2.8}$$

Dependent on the prior distributions we write the joint distribution of the Bayesian SBM as

$$p(\boldsymbol{A}, \boldsymbol{Z}, \boldsymbol{\pi}, \boldsymbol{\lambda} \mid \boldsymbol{\delta}^0, \boldsymbol{\alpha}^0, \boldsymbol{\beta}^0, K) = \prod_{i \neq j}^{N} \prod_{k,l}^{K} f(A_{ij}; \lambda_{kl})^{Z_{ik} Z_{jl}} \prod_{i=1}^{N} \prod_{k=1}^{K} \pi_k^{Z_{ik}} \prod_{k,l=1}^{K} p(\lambda_{kl}) p(\boldsymbol{\pi}). \tag{2.9}$$

From eqn. (2.9) we see that the joint distribution of the Bayesian Poisson SBM is dependent on the hyperparameters. This observation leads us to the task of finding suitable hyperparameters.

Choice of the Hyperparameters The discussion of the choice of hyperparameters for the edge existence probabilities, $\boldsymbol{\theta}$, or rates, $\boldsymbol{\lambda}$, is postponed to chapter 5 where we will link that choice to the approach of our new BlockVB++ algorithm. Our BlockVB++ algorithm uses our newly introduced *informative adaptive hyperparameters*.
We will see in chapter 5 and in the numerical tests in chapter 10 that the choice of hyperparameters is of huge importance for the quality of the results for the inference of the SBM.

2.3 The restricted Stochastic Block Model

In this section we shortly review the restricted Stochastic Block Model (restricted SBM) introduced in [50]. The restricted SBM was proposed to model communities or modules of vertices which are more densely connected to vertices in the same module than to to other vertices in the network. Alternatively, it can also model dis-assortative connected clusters, where the vertices of one cluster are sparsely linked to each other but more densely linked to vertices in other clusters [71]. We will refer to the VBEM algorithm for inference of the restricted SBM presented in [50] in the later chapters, so we present the model here.
The restricted SBM is a special case of the undirected Bernoulli SBM. It generates undirected simple (unweighted) network, which can be extended to the directed case. There are N vertices and K clusters like in the Bernoulli SBM, but contrary to the Bernoulli SBM, there are only two connection probabilities: the intra–cluster edge existence probability, θ_{in} and the edge existence probability between clusters, θ_{out}.
Due to its design, the restricted SBM cannot model hubs which often arise in real world networks [73]. The design of the restricted SBM leads to a lower order of the computational costs per iteration for VBEM inference than for the SBM [50, 35, 73].

2.4 The Erdös–Rényi graph

The famous *Erdös–Rényi graph (ER–graph)* presented in [98], can be retrieved as a special case of the Bernoulli SBM for $K = 1$ cluster, one edge connection probability, θ, and N vertices [35]. Obviously, the ER–graph is a limited model which gave rise to more complex models in the past which can capture important the properties of real world networks better [35]. For a further discussion of the relation of the SBM to the ER–graph we refer to [35].

Chapter 3

Review of the EM and Variational EM algorithms

In this section we will review deterministic inference algorithms based on the two step *Expectation Maximisation (EM)* algorithm with regard to the Stochastic Block Model (SBM). We have seen in chapter 2 how to generate a network with a SBM. Now, we will review the theoretical background for addressing the opposite task of inferring a SBM of a given network.
First we will introduce some general terminology and then we will explain the general principles of the most important deterministic algorithms for inference of the SBM. A review of EM based methods for inference of the SBM was also given e.g. in [50, 125, 71]. A source for a general overview of EM based algorithms is provided in [19].
We start with the statement of the likelihood of the SBM [35, 84]. We will see that marginalisation of the likelihood is only computationally feasible for very small networks [35, 50]. Then we review in section 3.2 why the EM algorithm is not a solution for inference of the SBM either [35]. This realisation will lead us to the topic of network interdependency which prevents the application of the EM algorithm for the SBM [35, 40].
Thus, a *Variational EM* algorithm based on [62, 55] was proposed in [35] for the SBM, making deterministic inference of the SBM for larger networks possible. We will review the general principles of the VEM algorithm in section 3.3.
Finally, building upon the VEM algorithm, we will review the general theory of the *Variational Bayesian EM* algorithm [15, 14] in section 3.4. The VBEM algorithm provides a Bayesian framework for the inference of the posterior distributions over the model parameters and leads itself naturally to a model selection criterion based on the approximation of the marginal likelihood [15]. We will see in chapter 6 that this criterion can also be used for inference of the SBM and to determine the unknown number of clusters of the SBM [50, 73].
The VBEM inference will be the base for all our new inference algorithms we will introduce in this thesis.

3.1 Likelihood of the Stochastic Block Model

The likelihood of the (undirected) Bernoulli SBM was introduced in [35] using the framework of [36]. In this framework, the unknown optimal cluster assignments (also called cluster labels), $\boldsymbol{Z}$ are the *latent* or *hidden* variables. The adjacency matrix $\boldsymbol{A}$ is the given, *incomplete–* or *known data*. The set $(\boldsymbol{A},\boldsymbol{Z})$ is then called the *complete data set*. We use the general formulation of the *complete data log–likelihood*, proposed in [84] for directed networks and the distribution $f(\lambda_{kl};A_{ij})$ for possibly valued edges (see chapter 2). This complete data log–likelihood is given by [84]:

$$\ln p(\boldsymbol{A},\boldsymbol{Z}) = \sum_i \sum_k Z_{ik} \ln \pi_k + \sum_{i\neq j} \sum_{k,l} Z_{ik} Z_{jl} \ln f\left(\lambda_{kl};A_{ij}\right). \tag{3.1}$$

It is based on the complete data log–likelihood of the undirected Bernoulli SBM proposed in [35]. It was shown in [35] and [84] that the formulation of the complete data log–likelihood in eqn. (3.1) holds because of

$$\ln p(\boldsymbol{A},\boldsymbol{Z}) = \ln p(\boldsymbol{Z}) + \ln p(\boldsymbol{A}|\boldsymbol{Z}) \tag{3.2}$$

where

$$\ln p(\boldsymbol{Z}) = \sum_i \sum_k Z_{ik} \ln \pi_k, \tag{3.3}$$

$$\ln p(\boldsymbol{A}|\boldsymbol{Z}) = \sum_{i\neq j} \sum_{k,l} Z_{ik} Z_{jl} \ln f\left(\lambda_{kl};A_{ij}\right). \tag{3.4}$$

Now, as a first idea for inference, it was considered in [35] to calculate the likelihood of the observed data by marginalisation:

$$\ln p(\boldsymbol{A}) = \sum_{\boldsymbol{Z}} \ln p(\boldsymbol{Z},\boldsymbol{A}). \tag{3.5}$$

Unfortunately this summation over all possible cluster assignments leads to high computational costs which render this summation intractable save for very small networks [35, 50, 84]. The summation over all possible cluster assignments has K^N terms [50, 84], where K is the number of clusters and N the number of vertices of the network.

3.2 The Expectation Maximisation (EM) algorithm

The classical Expectation Maximisation (EM) algorithm [36] was shortly considered in [35] to avoid the direct calculation of the intractable likelihood of the observed data, $p(\boldsymbol{A})$, of the SBM.
The EM algorithm is used to find the model parameters $\boldsymbol{\vartheta}$ which maximise the likelihood $p(\boldsymbol{A}|\boldsymbol{\vartheta})$ without the need for the marginalisation of eqn. 3.5 [19]. We will see below that the EM algorithm also is intractable in case of the SBM [35, 40].
Now, we shortly review the general principles of the classical **EM algorithm** following [19].
(i) As a first step of the EM algorithm, the model parameters are initialised with, $\boldsymbol{\vartheta}^{(old)}$.

(ii) Then the EM algorithm continues with the *Expectation step (E–step)*, where the *posterior distribution*, $p(\boldsymbol{Z}|\boldsymbol{A},\boldsymbol{\vartheta}^{(old)})$ is calculated.

The calculation of the conditional distribution $p(\boldsymbol{Z}|\boldsymbol{A}, \boldsymbol{\vartheta}^{(old)})$ involves the summation over all possible cluster assignments in the case of the SBM and is therefore computationally intractable as the likelihood [35, 40]. Moreover, it will be shown below following [35, 40], that a simplification of the term $p(\boldsymbol{Z}|\boldsymbol{A}, \boldsymbol{\vartheta}^{(old)})$ is not possible for the SBM, because of network interdependency. So, the EM algorithm is not a viable choice for inference of the SBM.
Nevertheless we state the rest of the EM algorithm because a two step inference procedure will also be introduced below for the VEM and VBEM algorithm, which are viable algorithms for the SBM.

(iii) In the *Maximisation Step (M-step)*, the expectation of the complete-data likelihood with respect to the posterior distribution, which is given by

$$Q(\boldsymbol{\vartheta}, \boldsymbol{\vartheta}^{(old)}) = \sum_{\boldsymbol{Z}} p(\boldsymbol{Z}|\boldsymbol{A}, \boldsymbol{\vartheta}^{old}) \ln p(\boldsymbol{A}, \boldsymbol{Z}|\boldsymbol{\vartheta}), \tag{3.6}$$

is maximised to find an update for the model parameters, $\boldsymbol{\vartheta}^{(old)}$:

$$\boldsymbol{\vartheta}^{(new)} = \arg\max_{\{\boldsymbol{\vartheta}\}} Q(\boldsymbol{\vartheta}, \boldsymbol{\vartheta}^{old}). \tag{3.7}$$

(iv) After the M–step, the convergence of the log–likelihood is checked indirectly by calculating the difference between $\boldsymbol{\vartheta}^{(old)}$ and $\boldsymbol{\vartheta}^{(new)}$. If no convergence was reached, the model parameters are updated, e.g. $\boldsymbol{\vartheta}^{(old)} \longleftarrow \boldsymbol{\vartheta}^{(new)}$, and the inference continues in step **(ii)**.

It was remarked in [93], that the EM algorithm converges to a local optimum for most models. For a proof that the EM algorithm indeed maximises the observed likelihood, we refer to e.g. [93, 19, 71].
We stated the classical EM algorithm to motivate the introduction of variational methods in sections 3.3 and 3.4, where a solution to the intractability of the posterior likelihood, $p(\boldsymbol{Z}|\boldsymbol{A}, \boldsymbol{\vartheta})$, will be introduced.

Network Interdependency The following counterexample was proposed in [35], where it was shown that the likelihood of the Bernoulli SBM depends on the whole network and cannot be simplified with respect to neighbourhoods of the vertices.
The neighbourhood, E_i, of a vertex, i, are all edges and non–existing edges (non–edges) with respect to i.The set of edges in a network is denoted by, E.
An undirected Bernoulli SBM with $K^\star = 2$ clusters and cluster connection probabilities

$$\boldsymbol{\theta} = \begin{pmatrix} \theta_{11} & \theta_{12} \\ \theta_{21} & \theta_{22} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & \mu \end{pmatrix}, \tag{3.8}$$

where $0 < \mu < 1$, was considered in [35] for this counter–example.
It is assumed that a network with $N = 3$ vertices, i, j, k was generated by this SBM and that $A_{ij} = 1$ and $A_{jk} = 1$ holds.
From these edge connections it is drawn the conclusion that i, j have to be in the same cluster because there are no edges connecting the clusters because of $\theta_{21} = \theta_{12} = 0$.

With the same reasoning it can be seen that vertices i and k have to be in the same cluster.
We can infer from the existence of the edge $A_{jk} = 1$ that vertex i could be in both clusters k_1 and k_2 with probability greater than zero, so it holds that $\mathbb{P}(Z_{i1} = 1|E_i, A_{jk}) > 0$ and $\mathbb{P}(Z_{i1} = 1|E_i, A_{jk}) > 0$. If on the other hand it is assumed that $A_{jk} = 0$ holds with the other assumptions being the same, we see that $\mathbb{P}(Z_{i1} = 1|E_i, A_{jk}) = 0$. Therefore the information of the existence of an edge between vertices j and k cannot be left out for the cluster assignment of vertex i.
This observations shows, that all edges and non–edges have to be considered for the cluster assignment, $\mathbb{P}(Z_{iq} = 1|E)$, and not only the neighbourhood of vertex i, $\mathbb{P}(Z_{iq}|E_i)$, which makes a simplification with respect to the neighbourhoods impossible [35].
This counter example applies to the undirected Bernoulli SBM. We will also treat a weighted version of the SBM [84] (see chapter 2).
It was shown for the SBM in general in [40], that the cluster assignment of each vertex depends on the cluster assignments of all other vertices of the network. Therefore, the graph which describes the dependency of the hidden cluster assignments of the vertices, $\boldsymbol{Z}_i \in \{1, \ldots, N\}$ forms a fully connected graph or clique [40].
It was emphasised in [40], that this dependency applies independently from the actual existence of edges of the observed graph. So, the claim of the impossibility of simplification of the SBM with respect to the neighbourhood of the vertices applies in any case.

3.3 Variational Expectation Maximisation

In section 3.2, it was explained that the EM algorithm is computationally unviable for the inference of the SBM due to the computational intractability of $p(\boldsymbol{Z}|\boldsymbol{A})$. A solution for this problem was proposed in [35], where the variational EM strategy of [62] and [55] was applied to the Bernoulli SBM.
We now explain the general principle of the *Variational Expectation Maximisation (VEM)* algorithm following [19, 71, 35]. Again, we denote the observed variables as $\boldsymbol{A}$, the hidden variables as $\boldsymbol{Z}$ and the unknown model parameters as $\boldsymbol{\vartheta}$.
With the help of the VEM algorithm we want to optimise the computationally intractable log–likelihood which was marginalised over the hidden variables, $\boldsymbol{Z}$,

$$\ln\left(p(\boldsymbol{A}|\boldsymbol{\vartheta})\right) = \ln\left(\sum_{\boldsymbol{Z}} p(\boldsymbol{A}, \boldsymbol{Z}|\boldsymbol{\vartheta})\right). \tag{3.9}$$

To start the inference, which is otherwise intractable as we have seen in section 3.2, an arbitrary variational distribution over the hidden variables $\boldsymbol{Z}$, $q(\boldsymbol{Z})$, to approximate $p(\boldsymbol{Z}|\boldsymbol{A})$ is introduced. Dependent on the variational distribution, $q(\boldsymbol{Z})$, the following decomposition of the likelihood dependent can be made:

$$\ln\left(p(\boldsymbol{A}|\boldsymbol{\vartheta})\right) = \mathscr{L}(q(\boldsymbol{Z}), \boldsymbol{\vartheta}) + \mathrm{KL}\left(q(\cdot)\,||\,p(\cdot|\boldsymbol{A})\right) \tag{3.10}$$

with

$$\mathscr{L}(q, \boldsymbol{\vartheta}) = \sum_{\boldsymbol{Z}} \ln\left(\frac{p(\boldsymbol{A}, \boldsymbol{Z}|\boldsymbol{\vartheta})}{q(\boldsymbol{Z})}\right), \tag{3.11}$$

$$\mathrm{KL}(q\,||\,p) = -\sum_{\boldsymbol{Z}} q(\boldsymbol{Z}) \ln\left(\frac{p(\boldsymbol{Z}|\boldsymbol{A}, \boldsymbol{\vartheta})}{q(\boldsymbol{Z})}\right), \tag{3.12}$$

where KL denotes the Kullback–Leibler divergence. For background information of the KL-divergence see e.g. [19]. We see that the decomposition for $\ln(p(\boldsymbol{A}|\boldsymbol{\vartheta}))$ in eqn. 3.10 holds by calculating

$$\begin{aligned}
\mathscr{L}(q,\boldsymbol{\vartheta}) &= \sum_{\boldsymbol{Z}} q(\boldsymbol{Z}) \ln p\big(\boldsymbol{A},\boldsymbol{Z}|\boldsymbol{\vartheta}\big) - \sum_{\boldsymbol{Z}} q(\boldsymbol{Z}) \ln q(\boldsymbol{Z}) && (3.13)\\
&= \sum_{\boldsymbol{Z}} q(\boldsymbol{Z}) \left[\ln p(\boldsymbol{Z}|\boldsymbol{A},\boldsymbol{\vartheta}) - \ln q(\boldsymbol{Z})\right] + \ln p\left(\boldsymbol{A}|\boldsymbol{\vartheta}\right) && (3.14)\\
&= \sum_{\boldsymbol{Z}} q(\boldsymbol{Z}) \left[\ln\left(\frac{p(\boldsymbol{Z}|\boldsymbol{A},\boldsymbol{\vartheta})}{q(\boldsymbol{Z})}\right)\right] + \ln p(\boldsymbol{A}|\boldsymbol{\vartheta}) && (3.15)\\
&= -\mathrm{KL}(q||p) + \ln p(\boldsymbol{A}|\boldsymbol{\vartheta}), && (3.16)
\end{aligned}$$

where we used that

$$\ln p(\boldsymbol{A}|\boldsymbol{\vartheta}) = \ln p(\boldsymbol{Z},\boldsymbol{A}|\boldsymbol{\vartheta}) - \ln p(\boldsymbol{Z}|\boldsymbol{A},\boldsymbol{\vartheta}), \tag{3.17}$$

holds. It follows from eqn. (3.10) that

$$\ln p(\boldsymbol{A}|\boldsymbol{\vartheta}) \geq \mathscr{L}(q,\boldsymbol{\vartheta}), \tag{3.18}$$

because $KL(q||p) \geq 0$ and $KL(q\,||\,p) = 0$ is equivalent to $q(\boldsymbol{Z}) = p(\boldsymbol{Z}|\boldsymbol{A},\boldsymbol{\vartheta})$. This shows that $\mathscr{L}(q,\boldsymbol{\vartheta})$ is a lower bound of the likelihood [19, 71].
The factorisation of the variational distribution with respect to the hidden variables, $q(\boldsymbol{Z}) = \prod_i q_i(\boldsymbol{Z}_i)$, also called *mean–field–assumption* [55], is used to achieve a tractable form of $\mathscr{L}(q,\boldsymbol{\vartheta})$ at the price of an approximation [19, 35, 71]. In [35] it is assumed in the case of the SBM, that

$$q(\boldsymbol{Z}) = \prod_{i=1}^{N} \mathscr{M}(\boldsymbol{Z}_i;\boldsymbol{Q}_i) = \prod_{i=1}^{N} \boldsymbol{Q}_i^{\boldsymbol{Z}_i} \tag{3.19}$$

where $\mathscr{M}(\boldsymbol{Z}_i;\boldsymbol{Q}_i)$ is the Multinomial distribution.
Now, the Likelihood is optimised dependent on $q(\boldsymbol{Z})$ and the model parameters $\boldsymbol{\vartheta}$ in a two step algorithm which is reminiscent of the EM–algorithm of section 3.2.
This two step algorithm [35] consists of initialising the start values for the hidden variables, $\boldsymbol{Z} \equiv \boldsymbol{Z}_0$, and dependent on those start values, point estimates for the model parameters are optimised in the following *M–step*

$$\{\boldsymbol{\vartheta}^{(t+1)}\} = \arg\max_{\{\boldsymbol{\vartheta}\}} \mathscr{L}\left(q^{(t)}(\boldsymbol{Z}),\boldsymbol{\vartheta}^{(t)}\right). \tag{3.20}$$

After the M–step, $q(\boldsymbol{Z})$ is optimised in the *E–step*, dependent on the update parameters, $\boldsymbol{\vartheta}^{(t+1)}$:

$$\{q^{(t+1)}(\boldsymbol{Z})\} = \arg\max_{\{q(\boldsymbol{Z})\}} \mathscr{L}(q^{(t)}(\boldsymbol{Z}),\boldsymbol{\vartheta}^{(t+1)}). \tag{3.21}$$

The M–and the E–step are repeated until convergence of $\mathscr{L}(q,\boldsymbol{\vartheta})$. We sum up the VEM algorithm in 1.
The VEM algorithm is a popular algorithm for inference of the SBM and was used e.g. in [35, 84, 123, 40, 75, 76, 79]. It was shown under some mild assumptions in [25], that VEM inference for the SBM is consistent. The VEM algorithm converges to a local optimum in the case of the SBM [25].

Algorithm 1: Variational Expectation Maximisation (VEM) algorithm [35, 71]

```
/* Initialisation */
```
Initialise start values $\boldsymbol{Z}_0$
```
/* Main Loop */
```
repeat
```
    /* M-step */
```
$\boldsymbol{\vartheta}^{(t+1)} \longleftarrow \arg\max_{\{\boldsymbol{\vartheta}\}} \mathscr{L}\left(q^{(t)}(\boldsymbol{Z}), \boldsymbol{\vartheta}^{(t)}\right)$
```
    /* E-step */
```
$q^{(t+1)}(\boldsymbol{Z}) \longleftarrow \arg\max_{\{q(\boldsymbol{Z})\}} \mathscr{L}(q^{(t)}(\boldsymbol{Z}), \boldsymbol{\vartheta}^{(t+1)})$

until *convergence of* $\mathscr{L}(\cdot)$

The VEM algorithm is dependent on start values. Therefore, these start values have to be initialised either randomly or with another algorithm. In the case of the SBM, a random initialisation of start values is not recommended in the literature [84, 75, 76] and returns poor results in most cases. We will mention and introduce important algorithms for the initialisation of start values, like the k-means or Ascending Hierarchical Clustering (AHC) algorithm, which are also used for VEM inference of the SBM in chapter 7.
We have also seen, that the VEM algorithm returns point estimates for the model parameters which are susceptible to outliers [15].
Algorithms which return point estimates are called *frequentist* algorithms in the literature contrary to *nonparametric* or *Bayesian* algorithms, where prior distributions are set over the parameters and the full posterior distribution over the parameters is approximated [15].
We will review the non–parametric version of the VEM algorithm, the Variational Bayesian Expectation Maximisation in section 3.4. We will see that the nonparametric VBEM algorithm provides additional features and advantages compared to the VEM algorithm. The derivation of the VBEM algorithm will also lead to a model selection criterion for the SBM.

3.4 Variational Bayesian EM

In section 3.3, we saw that the VEM algorithm is a tractable, deterministic inference algorithm for the SBM. Nevertheless, we already mentioned some shortcomings of the VEM algorithm. It returns only point estimates of the parameters, is dependent on other algorithms for the initialisation of start values and needs a separate model selection criterion.
The algorithmic framework of the Variational Bayesian Expectation Maximisation (VBEM) algorithm [15] will provide us with answers to these issues with the VEM algorithm. The 'Variational Bayes(ian)' (VB) algorithmic framework was introduced by [15].
The VBEM framework will be the base for all our newly developed inference algorithms for the SBM we will present in this thesis. We will use it for our derivation of all our propositions for the VBEM inference algorithm for the Poisson SBM in section 4.
For the following presentation of the general VBEM algorithm, we follow [19, 71]. The

VBEM algorithm was designed for the approximation of the intractable marginalised log–likelihood given by

$$\ln p(\boldsymbol{A}) = \ln\left(\sum_{\boldsymbol{Z}}\int p(\boldsymbol{A},\boldsymbol{Z},\boldsymbol{\vartheta})\mathrm{d}\boldsymbol{\vartheta}\right). \tag{3.22}$$

This time, it is marginalised not only over the latent variables $\boldsymbol{Z}$ like for the VEM algorithm, but also over the model parameters $\boldsymbol{\vartheta}$.
An arbitrary variational distribution of the latent variables and model parameters, $q(\boldsymbol{Z},\boldsymbol{\vartheta})$, is introduced, which allows the decomposition of the log–likelihood, $\ln p(\boldsymbol{A})$, into the sum of

$$\mathscr{L}(q) = \sum_{\boldsymbol{Z}}\int q(\boldsymbol{Z},\boldsymbol{\vartheta})\ln\left(\frac{p(\boldsymbol{A},\boldsymbol{Z},\boldsymbol{\vartheta})}{q(\boldsymbol{Z},\boldsymbol{\vartheta})}\right)\mathrm{d}\boldsymbol{\vartheta}. \tag{3.23}$$

and

$$\mathrm{KL}(q(\cdot)\,||\,p(\cdot|\boldsymbol{A})) = -\sum_{\boldsymbol{Z}}\int q(\boldsymbol{Z},\boldsymbol{\vartheta})\ln\left(\frac{p(\boldsymbol{Z},\boldsymbol{\vartheta}\,|\boldsymbol{A})}{q(\boldsymbol{Z},\boldsymbol{\vartheta})}\right)\mathrm{d}\boldsymbol{\vartheta}. \tag{3.24}$$

Combining eqn. (3.35) and (3.36) leads to

$$\ln p(\boldsymbol{A}) = \mathscr{L}(q(\boldsymbol{Z},\boldsymbol{\vartheta})) + \mathrm{KL}\left(q(\cdot)||p(\cdot|\boldsymbol{A})\right). \tag{3.25}$$

The proof of the decomposition in eqn. (3.25) is analogous to those for the VEM algorithm. Like in section 3.3, it can be seen that $\ln p(\boldsymbol{A}) \geq \mathscr{L}\left((q(\boldsymbol{Z},\boldsymbol{\vartheta}))\right)$ holds by using that

$$q(\boldsymbol{Z},\boldsymbol{\vartheta}) = p(\boldsymbol{Z},\boldsymbol{\vartheta}|\boldsymbol{A}) \Leftrightarrow \mathrm{KL}(q||p) = 0 \tag{3.26}$$

and $\mathrm{KL}(q||p) \geq 0$. Therefore, $\mathscr{L}(q)$ is a lower bound dependent on the variational distribution q. Optimisation of the variational lower bound, $\mathscr{L}(\cdot)$, with respect to $q(\boldsymbol{Z},\boldsymbol{\vartheta})$ is intractable.Therefore, it is assumed that

$$q(\boldsymbol{Z},\boldsymbol{\vartheta}) = q_{\boldsymbol{Z}}(\boldsymbol{Z})q_{\boldsymbol{\vartheta}}(\boldsymbol{\vartheta}), \tag{3.27}$$

which makes the inference tractable [15]. This assumption which is often called the *mean–field–assumption* [55],'[...] makes q approximate but tractable.'[15].

3.4.1 Model Selection criterion

The mean–field–assumption also allows the following re–formulation of the variational lower bound $\mathscr{L}(q)$ with respect to $q(\boldsymbol{\vartheta})$ which shows that the number of model parameters is penalised [15]. We use that $p(\boldsymbol{A},\boldsymbol{Z},\boldsymbol{\vartheta}) = p(\boldsymbol{A},\boldsymbol{Z}|\boldsymbol{\vartheta})p(\boldsymbol{\vartheta})$ holds by the product rule of probability, this yields

$$\mathscr{L}(q) = \sum_{\boldsymbol{Z}}\int q_{\boldsymbol{Z}}(\boldsymbol{Z})q_{\boldsymbol{\vartheta}}(\boldsymbol{\vartheta})\ln\left(\frac{p(\boldsymbol{A},\boldsymbol{Z}|\boldsymbol{\vartheta})p(\boldsymbol{\vartheta})}{q_{\boldsymbol{Z}}(\boldsymbol{Z})q_{\boldsymbol{\vartheta}}(\boldsymbol{\vartheta})}\right)\mathrm{d}\boldsymbol{\vartheta} \tag{3.28}$$

$$= \sum_{\boldsymbol{Z}}\int q(\boldsymbol{Z})q(\boldsymbol{\vartheta})\ln\left(\frac{p(\boldsymbol{A},\boldsymbol{Z}|\boldsymbol{\vartheta})}{q(\boldsymbol{Z})q(\boldsymbol{\vartheta})}\right)\mathrm{d}\boldsymbol{\vartheta} + \int q(\boldsymbol{\vartheta})\ln\left(\frac{p(\boldsymbol{\vartheta})}{q(\boldsymbol{\vartheta})}\right)\mathrm{d}\boldsymbol{\vartheta} \tag{3.29}$$

$$= \sum_{\boldsymbol{Z}}\int q(\boldsymbol{Z})q(\boldsymbol{\vartheta})\ln\left(\frac{p(\boldsymbol{A},\boldsymbol{Z}|\boldsymbol{\vartheta})}{q(\boldsymbol{Z})q(\boldsymbol{\vartheta})}\right)\mathrm{d}\boldsymbol{\vartheta} - \mathrm{KL}\left(q(\boldsymbol{\vartheta})||p(\boldsymbol{\vartheta})\right). \tag{3.30}$$

If the number of parameters grows, the KL–divergence in eqn.(3.30) increases which lowers the variational bound, $\mathscr{L}(q)$ [15]. Therefore, the converged variational lower bound $\mathscr{L}(q)$ can be applied as a model selection criterion which follows from the VBEM framework. We refer to section 6.2, where we discuss the application of $\mathscr{L}(q)$ as a model selection criterion for the SBM.

3.4.2 Free form optimisation

For the following derivations of the Variational Bayesian EM inference algorithm we follow [19, 71]. The mean–field factorisation allows for a reformulation of $\mathscr{L}(\cdot)$ which leads to an inference algorithm with respect to the factors $q_\cdot(\cdot)$ of the variational distribution $q(\cdot)$.
For the sake of a concise notation, we set $\boldsymbol{\Theta} = (\boldsymbol{Z}, \boldsymbol{\vartheta})$. Consequently, the factorisation of eqn. (3.27) can be rewritten as $q(\boldsymbol{\Theta}) = \prod_i q_i(\boldsymbol{\Theta}_i)$. VBEM is a free form optimisation framework where we do not need to specify the functional form of the variational distributions in advance [15, 19]. The collection of all hidden variables except the hidden variable $\boldsymbol{\Theta}_a$, is denoted by $\boldsymbol{\Theta}^{\setminus a}$. The following derivations are the same if we use summation or integration of hidden variables.
Using this notation, $\mathscr{L}(q)$ is rewritten in the following way:

$$\mathscr{L}(q) = \int q(\boldsymbol{\Theta}) \ln\left(\frac{p(\boldsymbol{A},\boldsymbol{\Theta})}{q(\boldsymbol{\Theta})}\right) \mathrm{d}\boldsymbol{\Theta} \tag{3.31}$$

$$= \int \left(\prod_i q_i(\boldsymbol{\Theta}_i)\right) \left(\ln p(\boldsymbol{A},\boldsymbol{\Theta}) - \sum_i \ln q_i(\boldsymbol{\Theta}_i)\right) \mathrm{d}\boldsymbol{\Theta} \tag{3.32}$$

$$= \int \left(\prod_i q_i(\boldsymbol{\Theta}_i)\right) \ln p(\boldsymbol{A},\boldsymbol{\Theta}) \mathrm{d}\boldsymbol{\Theta} - \int \sum_i \ln q_i(\boldsymbol{\Theta}_i) \mathrm{d}\boldsymbol{\Theta} \tag{3.33}$$

$$= \int q_a(\boldsymbol{\Theta}_a) \left(\int \prod_{i\neq a} q_a(\boldsymbol{\Theta}_a) \ln p(\boldsymbol{A},\boldsymbol{\Theta}) \mathrm{d}\boldsymbol{\Theta}^{\setminus a}\right) \mathrm{d}\boldsymbol{\Theta}_a \tag{3.34}$$

$$- \int q_a(\boldsymbol{\Theta}_a) \mathrm{d}\boldsymbol{\Theta}_a - \sum_{i\neq a} \int q_i(\boldsymbol{\Theta}_i) \ln q_i(\boldsymbol{\Theta}_i) \mathrm{d}\boldsymbol{\Theta}_i. \tag{3.35}$$

We define

$$\int \prod_{i\neq a} q_a(\boldsymbol{\Theta}_a) \ln p(\boldsymbol{A},\boldsymbol{\Theta}) \mathrm{d}\boldsymbol{\Theta}_a = \mathbb{E}_{\boldsymbol{\Theta}^{\setminus a}} \left(\ln p(\boldsymbol{A},\boldsymbol{\Theta})\right) + \text{const} \equiv \ln \tilde{p}(\boldsymbol{A},\boldsymbol{\Theta}_a). \tag{3.36}$$

Combining eqn. (3.35) with eqn. (3.36) yields

$$\mathscr{L}(q) = \int q_a(\boldsymbol{\Theta}_a) \left(\ln \tilde{p}(\boldsymbol{A},\boldsymbol{\Theta}_a) - \text{const}\right) \mathrm{d}\boldsymbol{\Theta}_a - \int q_a(\boldsymbol{\Theta}_a) \ln q_a(\boldsymbol{\Theta}_a) \mathrm{d}\boldsymbol{\Theta}_a \tag{3.37}$$

$$- \sum_{i\neq a} \int q_i(\boldsymbol{\Theta}_i) \ln q_i(\boldsymbol{\Theta}_i) \mathrm{d}\boldsymbol{\Theta}_i \tag{3.38}$$

$$= -\mathrm{KL}\left(q_a(\cdot) \,||\, \tilde{p}(\cdot)\right) + \sum_{i\neq a} \mathscr{H}\left(q_i\right) - \text{const}. \tag{3.39}$$

It was used in eqn. (3.39), that the entropy of $q_i(\cdot)$ is given by

$$\mathscr{H}(q_i) = \int q_i(\boldsymbol{\Theta}_i) \ln q_i(\boldsymbol{\Theta}_i) \mathrm{d}\boldsymbol{\Theta}_i. \tag{3.40}$$

The KL–divergence in eqn. (3.39) being equal to zero is now equivalent to $q_a(\boldsymbol{\Theta}_a) = \tilde{p}(\boldsymbol{A}, \boldsymbol{\Theta}_a)$. If this equality holds, an optimum of the variational lower bound $\mathscr{L}(q)$ with respect to the factor $q_a(\boldsymbol{\Theta}_a)$ is reached and it follows from eqn. (3.39) that the optimal variational distribution $q_a^\star(\boldsymbol{\Theta}_a)$ is given by

$$\ln q_a^\star(\boldsymbol{\Theta}_a) = \ln \tilde{p}(\boldsymbol{A}, \boldsymbol{\Theta}_a) = \mathbb{E}_{\boldsymbol{\Theta}^{\setminus a}}(\ln p(\boldsymbol{A}, \boldsymbol{\Theta})) + \text{const.} \tag{3.41}$$

The constant is calculated by normalisation of $q_a^\star(\boldsymbol{A}, \boldsymbol{\Theta})$ which yields:

$$q_a^\star(\boldsymbol{\Theta}_a) = \frac{\exp\left(\int \prod_{i \neq a} q_a(\boldsymbol{\Theta}_a) \ln p(\boldsymbol{A}, \boldsymbol{\Theta}) \mathrm{d}\boldsymbol{\Theta}^{\setminus a}\right)}{\int \exp\left(\int \prod_{i \neq a} q_a(\boldsymbol{\Theta}_a) \ln p(\boldsymbol{A}, \boldsymbol{\Theta}) \mathrm{d}\boldsymbol{\Theta}^{\setminus a}\right) \mathrm{d}\boldsymbol{\Theta}_a}. \tag{3.42}$$

The variational bound, $\mathscr{L}(\cdot)$ can be optimised with respect to each factor distribution, $q_i(\cdot)$, of the the variational distribution $q(\cdot)$.The inference scheme is done factorwise because each factor, $q_a(\cdot)$, depends on the other factors $q_i(\cdot), i \neq a$ [19]. This gives rise for the need of an algorithm with an EM–like inference scheme analogous to those in sections 3.2 and 3.3 for the EM–and VEM algorithm. A global optimum with respect to all factors $q(\boldsymbol{\Theta}) = \prod_i q_i(\boldsymbol{\Theta}_i)$ is not necessarily reached.
In this VBEM algorithm, it is also distinguished between the optimisation with respect to the latent variables, $q_i(\boldsymbol{Z}_i)$, in the Expectation step (E–step) and the optimisation of the variational, factors dependent on the model parameters, $q_i(\boldsymbol{\vartheta}_i)$, in the Maximisation step (M–step). Thus we have $q(\boldsymbol{\Theta}) = q(\boldsymbol{Z}, \boldsymbol{\vartheta}) = q_{\boldsymbol{Z}}(\boldsymbol{Z}) q_{\boldsymbol{\vartheta}}(\boldsymbol{\vartheta}) = \prod_i q_{\boldsymbol{Z}_i}(\boldsymbol{Z}_i) \prod_j q_{\boldsymbol{\vartheta}_j}(\boldsymbol{\vartheta})$ for separate variational distributions dependent on the latent variables and the model parameters.
The VBEM algorithm starts with the initialisation of the factors $q_i(\cdot)$. After the initialisation of the factors $q_i(\cdot)$, an EM–like algorithm can be applied to successively optimise the lower bound $\mathscr{L}(\cdot)$ with respect to each factor. Then the variational bound $\mathscr{L}(\cdot)$ is optimised with respect to each factor $q_i(\boldsymbol{\Theta}_i)$ in the E–and M–step.
The variational bound, $\mathscr{L}(\cdot)$, is convex with respect to each factor $q_i(\cdot)$ which leads to the convergence of the variational bound, $\mathscr{L}(\cdot)$, for any initialisation of the algorithm [19]. For a general discussion of convex optimisation [19] referred to [22]. We sum up the VBEM algorithm in 2.

3.4.3 Conjugate Prior Distributions

The VBEM algorithm allows the use of conjugate prior distributions for the model parameters [15]. This can be seen using the update equations (3.41) for the optimal variational distribution, $q_a^\star(\boldsymbol{\vartheta}_a)$ [15].
We recall that two distributions are conjugate if a prior distribution, $p(\boldsymbol{\Theta})$, can be chosen so that

$$q(\boldsymbol{\Theta}) \propto f(\boldsymbol{\Theta}) p(\boldsymbol{\Theta}) \tag{3.43}$$

holds [15]. In this case $f(\boldsymbol{\Theta})$ and $q(\boldsymbol{\Theta})$ are from the same family of distributions, for example the exponential family. In many practical relevant cases $f(\cdot), p(\cdot)$ and $q(\cdot)$ are from the exponential family. In the case of the Poisson or Bernoulli Stochastic Block Model, this applies to all distributions [123].
The inclusion of conjugate prior distributions for the model parameters now follows

from eqn. (3.42) in the following way [15]:

$$\ln q_a^\star(\boldsymbol{\Theta}_a) = \mathbb{E}_{\boldsymbol{\Theta}^{\setminus a}}[\ln p(\boldsymbol{A},\boldsymbol{\Theta})] + \text{const} \tag{3.44}$$

$$\Rightarrow q_a^\star(\boldsymbol{\Theta}_a) \propto \exp\left(\mathbb{E}_{\boldsymbol{\Theta}^{\setminus a}}[\ln p(\boldsymbol{A},\boldsymbol{\Theta})]\right). \tag{3.45}$$

In the derivation above, we mentioned that one can distinguish between the latent variables, $\boldsymbol{Z}$, which can be for example the hidden cluster assignments of a SBM, and the model parameters $\boldsymbol{\vartheta}$. Prior distributions are set for the model parameters in the Bayesian inference setting which leads to following reformulation of eqn. (3.45) above:

$$q_a^\star(\boldsymbol{\Theta}_a) \propto \exp\left(\mathbb{E}_{\boldsymbol{\vartheta}^{\setminus a}}[\ln p(\boldsymbol{A},\boldsymbol{Z}\,|\,\boldsymbol{\vartheta})\,p(\boldsymbol{\vartheta})]\right) \tag{3.46}$$

$$= \exp\left(\mathbb{E}_{\boldsymbol{\vartheta}^{\setminus a}}[\ln p(\boldsymbol{A},\boldsymbol{Z}\,|\,\boldsymbol{\vartheta})]\right)\exp\left(\mathbb{E}_{\boldsymbol{\vartheta}^{\setminus a}}[\ln p(\boldsymbol{\vartheta}_a)]\right) \tag{3.47}$$

$$\propto \exp\left(\mathbb{E}_{\boldsymbol{\vartheta}^{\setminus a}}[\ln p(\boldsymbol{A},\boldsymbol{Z}\,|\,\boldsymbol{\vartheta})]\right)p(\boldsymbol{\vartheta}_a). \tag{3.48}$$

If we now set $f(\boldsymbol{\vartheta}_a) \equiv \exp\left(\mathbb{E}_{\boldsymbol{\vartheta}^{\setminus a}}[\ln p(\boldsymbol{A},\boldsymbol{Z}\,|\,\boldsymbol{\vartheta})]\right)$ it follows from eqn. (3.48) that

$$q_a^\star(\boldsymbol{\vartheta}_a^\star) \propto f(\boldsymbol{\vartheta}_a)p(\boldsymbol{\vartheta}_a), \tag{3.49}$$

and thus $p(\boldsymbol{\vartheta}_a)$ is the *conjugate prior distribution* to $f(\boldsymbol{\vartheta}_a)$. In chapter 2, conjugate prior distributions for the model parameters of the Bayesian SBM were set. Therefore, the VBEM algorithm for conjugate distributions allows the inference for the Bayesian SBM.

We will present our derivation of the VBEM algorithm for inference of the Bayesian Poisson SBM together with all propositions in section 4. This derivation is based on the general VBEM framework we provided in this section.

Algorithm 2: Variational Bayesian Expectation Maximisation (VBEM) algorithm

```
/* Initialisation */
```
Initialise start values $q_{\boldsymbol{Z}_i}(\boldsymbol{Z}_i)\forall i$
```
/* Main Loop */
```
repeat
  ```
  /* M-step */
  ```
  $q_{\boldsymbol{\vartheta}_a}^{(t+1)}(\boldsymbol{\vartheta}_a) \longleftarrow \exp\left(\sum_{\boldsymbol{Z}}\int\prod_i q_{\boldsymbol{Z}_i}^{(t)}(\boldsymbol{Z}_i)\prod_{j\neq a}q_{\boldsymbol{\vartheta}_j}^{(t)}(\boldsymbol{\vartheta}_j)\ln p(\boldsymbol{A},\boldsymbol{Z},\boldsymbol{\vartheta})\mathrm{d}\boldsymbol{\vartheta}^{\setminus a}\right)\ \forall a$

  Normalise $q_{\boldsymbol{\vartheta}_a}^{(t+1)}(\boldsymbol{\vartheta}_a)\ \forall a$
  ```
  /* E-step */
  ```
  $q_{\boldsymbol{Z}_a}^{(t+1)}(\boldsymbol{Z}_a) \longleftarrow \exp\left(\sum_{\boldsymbol{Z}^{\setminus a}}\int\prod_{i\neq a} q_{\boldsymbol{Z}_i}^{(t)}(\boldsymbol{Z}_i)\prod_j q_{\boldsymbol{\vartheta}_j}^{(t+1)}(\boldsymbol{\vartheta}_j)\ln p(\boldsymbol{A},\boldsymbol{Z},\boldsymbol{\vartheta})\mathrm{d}\boldsymbol{\vartheta}\right)\ \forall a$

  Normalise $q_{\boldsymbol{Z}_a}^{(t+1)}(\boldsymbol{Z}_a)\ \forall a$

until *convergence of* $\mathscr{L}(\cdot)$

Chapter 4

Variational Bayesian EM for the Poisson Stochastic Block Model

In chapter 2, we explained how to generate a graph with different types of the Stochastic Block Model (SBM). Now, we want to solve the inverse problem of inferring the best cluster assignment of the vertices of a given graph according to a Poisson SBM. In chapter 2 we found that we can cover all important types of networks with a directed Poisson SBM. We take the case of the directed Bayesian Poisson SBM as an exemplary 'workhorse model' for the new inference algorithms we will present for the SBM in the following chapters.
We assume that the network is given by its adjacency matrix, $\boldsymbol{A}$, without any prior knowledge of the now unknown (hidden) number of clusters K, the cluster assignment, $\boldsymbol{Z}$ or the model parameters $\boldsymbol{\vartheta} = (\boldsymbol{\lambda}, \boldsymbol{\pi})$. This scenario is common for the task of clustering [35, 50, 10, 84, 125, 123, 73, 76, 79] and we will also face it for the clustering of earthquake networks. The aim of all methods we will present below is to estimate the optimal number of clusters K, the unknown cluster assignment $\boldsymbol{Z}$ and the model parameters $\boldsymbol{\vartheta}$ of the SBM.

We discussed the general Variational Bayesian Expectation Maximisation (VBEM) framework of [19, 71] and [15, 14] in section 3.4. The use of the VBEM framework was proposed for the estimation of the restricted undirected Bernoulli SBM (see chapter 2) in [50]. The restricted SBM models is restricted to two different kinds of links: inter– and intra cluster connections (chapter 2). The use of the converged variational bound (converged free energy) as a model selection criterion for the restricted SBM was also proposed in [50].
A VBEM based inference algorithm and model selection criterion for the undirected Bernoulli SBM of [97], based on the free form optimisation reviewed in section 3.4, was introduced in [71, 73]. The SBM of [97] models the probabilities for the existence of edges dependent on the clusters.
The frequentist Variational Expectation Maximisation (VEM) framework of the same SBM was proposed in [35], and a VEM algorithm for the Poisson SBM and its variants in [84].
The VBEM inference for the weighted SBM was presented from a general point of view in [8]. There was an accompanying software package [7] of [8], where the VBEM batch algorithm for the Poisson SBM was implemented among others. We will com-

pare our following approach of VBEM inference of the Poisson SBM with this implementation in the numerical test section (chapter 10). There, we will see that our different algorithmic approach outperforms the implementation of [7].
The derivation in [8] was restricted to distributions from the exponential family. Contrary to [8], we will use the algorithmic framework of the more general free form optimisation of [19, 71] for our derivations, which we reviewed in section 3.4.2, where we do not make this restriction.

We now present our derivation of the Variational Bayesian EM algorithm for inference of the directed Bayesian Poisson SBM. We will present all necessary propositions and update equations for VBEM inference of the Poisson SBM. We will use the VBEM frame work to provide the free energy model selection criterion for the Poisson SBM.
We will use the Variational Bayesian framework to develop our new BlockVB algorithm which can be restricted to subsets of the vertices of the network in section 4.2.
We will show in chapter 5 that we can use the Bayesian features of the Variational Bayesian framework to develop our original fully Bayesian BlockVB++ algorithm, which does not use a second algorithm like K–means or Spectral Clustering to initialise the start values.

4.1 Propositions and Inference Algorithm

The network is given by the adjacency matrix, $\mathbf{A}$. We do not assume any prior knowledge about the number of clusters, the cluster assignment of vertices or model parameters. We want to infer the optimal number of clusters $K^\star$. For each vertex i, we want to calculate the optimal latent cluster membership (latent variable), $\boldsymbol{Z}_i$.
For a fixed number of clusters K, which is specified in advance, we want to infer the optimal posterior likelihood $p(\boldsymbol{Z},\boldsymbol{\lambda},\boldsymbol{\pi},K|\boldsymbol{A})$ of the Poisson SBM.
The approach for this optimisation with the VBEM framework is to approximate the posterior likelihood with an arbitrary tractable distribution $q(\boldsymbol{Z},\boldsymbol{\lambda},\boldsymbol{\pi}|K)$. To achieve a tractable distribution, we use the mean field assumption for the variational distribution $q(\boldsymbol{Z},\boldsymbol{\vartheta}) = q(\boldsymbol{\vartheta})q(\boldsymbol{Z}) = q(\boldsymbol{\pi})\prod_{k,l}^{K} q(\lambda_{k,l})\prod_{i=1}^{N} q(\boldsymbol{Z}_i)$. This assumption makes the VBEM inference approximate [15]. It is the only assumption necessary for this kind of VBEM inference [19].
We follow [50] for a different derivation than in section 3.4.2 of the variational bound of the marginal log-likelihood, $\ln\, p(\boldsymbol{A}|K)$, based on Jensen's inequality.We obtain a variational upper bound of the negative log likelihood, $-\ln\, p(\boldsymbol{A}|K)$, also called free energy F [50, 38], by using Jensen's inequality on the negative marginal log–likelihood:

$$\begin{aligned}
-\ln\, p(\boldsymbol{A}|K) &= -\ln \sum_{Z} \int p(\boldsymbol{A},\boldsymbol{Z},\boldsymbol{\vartheta}|K)\mathrm{d}\boldsymbol{\vartheta} \\
&= -\ln \sum_{Z} \int \frac{p(\boldsymbol{A},\boldsymbol{Z},\boldsymbol{\vartheta}|K)}{q(\boldsymbol{Z})q(\boldsymbol{\vartheta})} q(\boldsymbol{Z})q(\boldsymbol{\vartheta})\mathrm{d}\boldsymbol{\vartheta} \\
&\leq -\sum_{Z} \int \ln\left(\frac{p(\boldsymbol{A},\boldsymbol{Z},\boldsymbol{\vartheta}|K)}{q(\boldsymbol{Z},\boldsymbol{\vartheta})}\right) q(\boldsymbol{Z},\boldsymbol{\vartheta})\mathrm{d}\boldsymbol{\vartheta} \\
&\equiv F\left[q(\boldsymbol{Z},\boldsymbol{\vartheta})\right]. \qquad (4.1)
\end{aligned}$$

Therefore, a lower free energy is a better approximation of the marginal likelihood. We recall, that in section 3.4.2 a lower variational bound was derived.

Following the VBEM framework of section 3.4.2, the negative log–likelihood $-\ln\ p(\boldsymbol{A}|K)$ can as well be decomposed into

$$-\ln\ p(\boldsymbol{A}|K) = F\left[q(\boldsymbol{Z},\boldsymbol{\vartheta})\right] - \mathrm{KL}\left(q(\cdot)\,||\,p(\cdot|K)\right), \tag{4.2}$$

where

$$-\mathrm{KL}\left(q(\cdot)\,||\,p(\cdot|\boldsymbol{A})\right) = \sum_{\boldsymbol{Z}}\int q(\boldsymbol{Z},\boldsymbol{\vartheta})\ln\left(\frac{p(\boldsymbol{A},\boldsymbol{Z},\boldsymbol{\vartheta}|K)}{q(\boldsymbol{Z})q(\boldsymbol{\vartheta})}\right)\mathrm{d}\boldsymbol{\vartheta} \tag{4.3}$$

is the negative Kullback–Leibler–divergence (KL–divergence). So, minimising the free energy with respect to $q(\cdot)$ is the same as minimising the negative KL–divergence between the posterior distribution $p(\cdot\,|\boldsymbol{A})$ and the approximating distribution $q(\cdot)$. Equality is reached, if $q(\cdot) = p(\cdot\,|\boldsymbol{A})$. Therefore we can now apply the VBEM two step algorithm of section 3.4.2.
We provide the free energy criterion or converged free energy for the directed Poisson SBM without self–edges in our Proposition 1, below. The properties and applications of the free energy model selection criterion for the Poisson SBM are discussed in depth in chapter 6. There, other model selection criteria for the Poisson SBM are also discussed and compared to the free energy criterion.

Proposition 1 (Converged Free Energy)**.** *The free energy after convergence (negative Integrated Likelihood variational bound (ILvb)) for the directed Poisson Stochastic Block Model for K clusters, is given by:*

$$\begin{aligned} F[q(\boldsymbol{Z},\boldsymbol{\vartheta})] = &\sum_{k,l}^{K}\ln\left(\frac{\beta_{kl}^{\alpha_{kl}}\Gamma(\alpha_{kl}^0)}{\beta_{kl}^{0\,\alpha_{kl}^0}\Gamma(\alpha_{kl})}\right) + \sum_{i=1}^{N}\sum_{k=1}^{K}Q_{ik}\ln Q_{ik} \\ &+\ln\left(\frac{\Gamma\left(\sum_{x=1}^{K}\delta_x\right)\prod_{x=1}^{K}\Gamma(\delta_x^0)}{\Gamma\left(\sum_{x=1}^{K}\delta_x^0\right)\prod_{x=1}^{K}\Gamma(\delta_x)}\right), \end{aligned} \tag{4.4}$$

where $\boldsymbol{Q}$ is the cluster partition matrix, $\boldsymbol{A}$ the adjacency matrix, $\boldsymbol{\vartheta} = (\boldsymbol{\lambda},\boldsymbol{\pi})$ the model parameters and $(\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\delta},\boldsymbol{\alpha}^0,\boldsymbol{\beta}^0,\boldsymbol{\delta}^0)$ thehyperparameters of the variational- and prior distributions of the model parameters.

Proof. See appendix B. □

The VBEM algorithm for the Poisson SBM of chapter 2, consists now of the two following steps [35, 50, 73]:

Maximisation step In the Maximisation Step (M-Step), the free energy F is optimised with respect to the distributions of the parameters $q(\boldsymbol{\vartheta})$, with the latent variables of the cluster memberships $q(\boldsymbol{Z})$ held fixed:

$$\{q^{(t+1)}(\boldsymbol{\vartheta})\} = \underset{\{q(\boldsymbol{\vartheta})\}}{\arg\min}\, F[q^{(t)}(\boldsymbol{Z}), q^{(t)}(\boldsymbol{\vartheta})]. \tag{4.5}$$

We present the update equations for the hyperparameters $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ of the variational distribution $q(\boldsymbol{\lambda})$ in the next Proposition 2.

Proposition 2. *The optimisation of the upper variational bound (free energy) for $q(\lambda_{kl})$ for all $k,l = 1,\ldots,K$ shows, that $q(\lambda_{kl})$ has the functional form of a Gamma$(\lambda_{kl};\alpha_{kl},\beta_{kl})$ distribution. It has the same functional form as the prior distribution $p(\lambda^0_{kl}) = Gamma(\lambda_{kl};\alpha^0_{kl},\beta^0_{kl})$. The update equations of the hyperparameters α_{kl} and β_{kl} for all $k,l = 1,\ldots,K$ are calculated according to:*

$$\alpha_{kl} = \sum_{i\neq j}^{N} Q_{ik}Q_{jl}A_{ij} + \alpha^0_{kl}, \tag{4.6}$$

$$\beta_{kl} = \sum_{i\neq j}^{N} Q_{ik}Q_{jl} + \beta^0_{kl}. \tag{4.7}$$

Proof. See appendix B. □

The functional form of the variational distribution $q(\boldsymbol{\pi})$ and the update equations for thehyperparameters are the same as in [73] and we can use the following proposition of [73].

Proposition 3 ([73])**.** *The optimisation of the upper bound (free energy) with respect to $q(\boldsymbol{\pi})$ produces a distribution with the same functional form as the prior $p(\boldsymbol{\pi})$*

$$q(\boldsymbol{\pi}) = \mathrm{Dir}(\boldsymbol{\pi};\boldsymbol{\delta}) \tag{4.8}$$

where

$$\delta_k = \sum_{i=1}^{K} Q_{ik} + \delta^0_k. \tag{4.9}$$

Expectation step In the Expectation step (E-step), F is optimised with respect to the distributions $q(\boldsymbol{Z})$ of the latent variables where the distributions of the model parameters $q(\boldsymbol{\vartheta})$ are held fixed:

$$\{q^{(t+1)}(\boldsymbol{Z})\} = \arg\min_{\{q(\boldsymbol{Z})\}} F[q^{(t)}(\boldsymbol{Z}), q^{(t+1)}(\boldsymbol{\vartheta})]. \tag{4.10}$$

We provide the optimal distributions $q(\boldsymbol{Z}_i)\forall i$ and the update equations of $\mathbb{E}[Z_{ik}]$, $\forall(i,v) \in \{1,\ldots,N\}\times\{1,\ldots,K\}$ in the following Proposition 4.

Proposition 4. *The optimisation of the free energy (upper variational bound) with respect to $q(\boldsymbol{Z}_i)$ $\forall i = 1,\ldots,N$, $q^\star(\boldsymbol{Z}_i) = \arg\min_{q(\boldsymbol{Z}_i)} F[q(\boldsymbol{Z}), q(\boldsymbol{\vartheta})]$, shows that $q^\star(\boldsymbol{Z}_i)$ has the functional form of a multinomial distribution:*

$$q^\star(\boldsymbol{Z}_i) = \mathscr{M}(\boldsymbol{Z}_i; 1, \boldsymbol{Q}_i = \{Q_{i1},\ldots,Q_{iK}\}). \tag{4.11}$$

The update equation for $\mathbb{E}[Z_{ik}] = Q_{ik}$, $\forall(i,v) \in \{1,\ldots,N\}\times\{1,\ldots,K\}$ is given by

$$Q^\star_{av} \propto \exp\Big(\sum_{\substack{i=1\\ i\neq a}}^{N}\sum_{k=1}^{K} A_{ai}Q_{ik}C_{vk} + \sum_{\substack{i=1\\ i\neq a}}^{N}\sum_{k=1}^{K} A_{ia}Q_{ik}C_{kv} - \sum_{\substack{i=1\\ i\neq j}}^{N}\sum_{k=1}^{K} Q_{ik}D_{vk} + G_v\Big), \tag{4.12}$$

$$\tag{4.13}$$

where $\mathbb{E}_{\boldsymbol{\lambda}}[\log \lambda_{vk}] = \psi(\alpha_{vk}) - \log(\beta_{vk}) = C_{vk}$, $\mathbb{E}_{\boldsymbol{\lambda}}[\lambda_{vk}] + \mathbb{E}_{\boldsymbol{\lambda}}[\lambda_{kv}] = \frac{\alpha_{vk}}{\beta_{vk}} + \frac{\alpha_{kv}}{\beta_{kv}} = D_{vk}$, $\mathbb{E}_{\boldsymbol{\pi}}[\boldsymbol{\pi}_v] = \psi(\delta_v) - \psi\left(\sum\left(\sum_{l=1}^{K} \delta_l\right)\right) = G_v$ *and* $\psi(\cdot)$ *is the Digamma function.*

Proof. See appendix B. □

We calculate $F[q^{(t+1)}(\boldsymbol{Z}), q^{(t+1)}(\boldsymbol{\vartheta})]$ dependent on the distributions of the latent cluster assignments $q^{(t+1)}(\boldsymbol{Z})$ and the model parameters $q^{(t+1)}(\boldsymbol{\vartheta})$ which were returned by the M–and E–Step. The VBEM algorithm has converged, if

$$F[q^{(t)}(\boldsymbol{Z}), q^{(t)}(\boldsymbol{\vartheta})] - F[q^{(t+1)}(\boldsymbol{Z}), q^{(t+1)}(\boldsymbol{\vartheta})] < T \tag{4.14}$$

holds, where T is a threshold or a maximum of iterations is reached.
We sum up the batch VBEM algorithm for the Poisson SBM in algorithm 3.

Algorithm 3: VBEM batch algorithm for the Poisson SBM

Data: adjacency matrix $\boldsymbol{A}$, fixed number of clusters K, prior parameters $\left(\boldsymbol{\alpha}^0, \boldsymbol{\beta}^0, \boldsymbol{\delta}^0\right)$, maximum number of iterations

`/* Initialisation */`

Calculate start values $\boldsymbol{Q}^0$ with the AHC or other basic clustering algorithm
Calculate parameter updates according to Propositions 2 and 3

`/* Main Loop */`

repeat

  `/* E-step:update all matrix entries of` $\boldsymbol{Q}$ `*/`

  repeat

    Calculate $Q_{iv}\ \forall\ (i,v) \in \{1,\ldots,N\} \times \{1,\ldots,K\}$ according to eqn. 4.12 in Proposition 4

  until $\boldsymbol{Q}$ *converges or maximum number of iterations*

  `/* M-step */`

  Update the model parameters $\boldsymbol{\lambda}$ and $\boldsymbol{\pi}$ according to Propositions 2 and 3

until *convergence of* $F[q(\cdot)]$ *or maximum number of iterations*

Complexity of the Batch Algorithm In the worst case, where each matrix entry of $\boldsymbol{A}$ is greater than zero, the complexity of one iteration of the VBEM batch algorithm for the Poisson SBM is $\mathcal{O}\left(K^2 N\right)$. In most cases though, the adjacency matrix of the network is sparse.
To achieve a more precise measure of the order of the computational costs of our batch algorithm which accounts for sparsity of $\boldsymbol{A}$, we follow [29] and estimate the complexity dependent on the number of edges, E, of the network.
To calculate E for weighted networks, we define the indicator function for the existence of an edge between the vertices i, j to be

$$\mathbb{I}_{A_{ij}} = \begin{cases} 1, & \text{if } A_{ij} > 0 \\ 0, & \text{else} \end{cases}. \tag{4.15}$$

It follows that $E = \sum_{j=1}^{N} \sum_{i=1}^{N} A_{ij}$. We have computational costs of $\mathcal{O}\left(K^2 E\right)$ for one iteration of the VBEM batch algorithm 3.
These costs are the same as for one iteration of the VEM algorithm of [84] for the frequentist version of the Poisson SBM.

The computational costs of the VEM algorithm of [35] and the VBEM algorithm of [73] for the simple (unweighted and undirected) Bernoulli SBM are also $\mathcal{O}\left(K^2E\right)$.
The order of the computational costs is the same for directed and undirected networks. In practice though, even if the order of the computational costs is the same, the computation time for undirected networks is always lower than for the same network with directed edges, because in the undirected case, we only have to take the upper (or lower) triangular matrix of the adjacency matrix, $\boldsymbol{A}$, into account.

Convergence to local optima and dependence on start values The VBEM (and VEM) batch algorithm for the (restricted) Bernoulli SBM converges only to a local optimum and thus are dependent on the start values [35, 50, 25, 73]. The same was found in [84] for the VEM batch algorithm for the Poisson SBM.
The convergence to different local optima, dependent on the start values, of VBEM inference was also noted in the case of mixture distributions in [111].
We also found in numerical tests, that the VBEM algorithm for the Poisson SBM converged to different local optima of the free energy for different start values. These local optima were sometimes far away from the global optimum according to the known ground truth.
We will propose remedies to this behaviour of the batch VBEM algorithm in chapter 8. We will also see in section 8.2.2 that different types of local optima can occur for VBEM inference of the SBM. There, we will classify these optima as favourable or unfavourable.
So, it is advisable for the application of the VBEM batch algorithm, Algorithm 3, to initialise it with different start values and then to choose the best result [50, 73]. One extreme example is the application of the VBEM framework to time–series analysis where the algorithm was initialised for several thousand restarts in [81]. Nevertheless, even initialisation of the VBEM batch algorithm with a wide range of different start values gives no guarantee for finding a global optimum of the free energy or a value near to it. We will give an example in chapter 10, that the initialisation of the VBEM batch algorithm for the Poisson SBM with many different start values is even not enough to reach certain values of the free energy. There, we will show that with our new inference algorithms, we will propose chapter 8, far better values of the free energy criterion were reached.

To determine the best result of all tests for different numbers of clusters, we need a model selection criterion. We have also seen in section 3.4 that the free energy after convergence (Proposition 1) can be used as a model selection criterion. The application and properties of the converged free energy and other important model selection criteria will be discussed in section 6.

4.2 VBEM Subset Algorithm: BlockVB

We prensented the VBEM batch algorithm for the inference of the Poisson SBM at the beginning of chapter 4. We will review the batch algorithm (algorithm 6) used in the literature [35, 50, 84, 73] to apply the variational bound of the VBEM algorithm and other criteria for model selection of the SBM in chapter 6. In this algorithmic framework, all vertices of the network are optimised in one iteration. All these algorithms are batch algorithms.
To prepare our Blockloading and Blockloading++ algorithms in chapter 8, we now

propose our original *BlockVB* algorithm for the Poisson SBM which is optimised for the application to subsets of vertices of the network. We developed the BlockVB algorithm out of the VBEM batch algorithm 3. Our BlockVB algorithm for the Poisson SBM is able to restrict the VBEM inference to subsets of the vertices of the network in a computationally efficient way.
There have also been proposed different approaches for variational subset based methods for model based clustering like VEM based online clustering [125, 123] or Stochastic Variational Inference (SVI) [42, 39]. For a more detailed review of important existing subset based (variational) methods for clustering we refer to chapter 7.
We denote the set of vertices which are considered for optimisation with I. We call this set the *active vertices*. Let $|I|$ be the number of vertices in I.
To summarise the computational costs in concise way, we denote the number of edges leaving or pointing to vertices in I by

$$E_I = \sum_{i \in I} \mathbb{I}_{A_{ij}} + \sum_{\substack{i \in I \\ j \notin I}} \mathbb{I}_{A_{ji}}, \tag{4.16}$$

where $\mathbb{I}$ is the indicator function introduced in section 4. Accordingly, the number of all vertices in the network is given by E (see section 4).
We assume that we have an existing start partition matrix, $\boldsymbol{Q}^{(start)}$, of the cluster assignments from previous calculations. We initialise start values, $\boldsymbol{Q}^I$, for the active vertices in I. To save computational time, we only want to calculate updates in the M– and E–Step which depend on the active vertices.
In the M–step of algorithm 3, we had to calculate the updates

$$\alpha_{kl} = S_{\alpha_{kl}} + \alpha^0_{kl}, \tag{4.17}$$

$$\beta_{kl} = S_{\beta_{kl}} + \beta^0_{kl}, \tag{4.18}$$

$$\delta_k = S_{\delta_k} + \delta^0_k, \tag{4.19}$$

for all $\forall k,l$, where

$$S_{\alpha_{kl}} = \sum_{i \neq j}^{N} Q_{ik} Q_{jl} A_{ij}, \tag{4.20}$$

$$S_{\beta_{kl}} = \sum_{i \neq j}^{N} Q_{ik} Q_{jl}, \tag{4.21}$$

$$S_{\delta_k} = \sum_{i=1}^{N} Q_{ik}. \tag{4.22}$$

The computational costs of these formulas are $\mathscr{O}\left(K^2 E\right)$. If we restrict the updates in the M–step to the set I we can save computational costs by calculating:

$$S^I_{\alpha_{kl}} = \sum_{\substack{i=1 \\ i \neq j}}^{N} \sum_{\substack{j \in I \\ i \neq j}} Q_{ik} Q_{jl} A_{ij} + \sum_{\substack{i \in I \\ i \neq j}} \sum_{\substack{j \notin I \\ j=1}}^{N} Q_{ik} Q_{jl} A_{ij}, \tag{4.23}$$

$$S^I_{\beta_{kl}} = \sum_{\substack{i=1 \\ i \neq j}}^{N} \sum_{\substack{j \in I \\ i \neq j}} Q_{ik} Q_{jl} + \sum_{\substack{i \in I \\ i \neq j}} \sum_{\substack{j \notin I \\ j=1}}^{N} Q_{ik} Q_{jl}, \tag{4.24}$$

$$S^I_{\delta_k} = \sum_{i \in I} Q_{ik}. \tag{4.25}$$

for all k,l. Using these formulas, we set $S^{(old)}_{\alpha_{kl}} = S^I_{\alpha_{kl}}$, $S^{(old)}_{\beta_{kl}} = S^I_{\beta_{kl}}$ and $S^{(old)}_{\delta_k} = S^I_{\delta_k}$. Then we calculate $S^I_{\alpha_{kl}}$, $S^I_{\beta_{kl}}$ and $S^I_{\delta_k}$. Now we can update the model parameters for all k,l:

$$\alpha_{kl} = S_{\alpha_{kl}} - S^{(old)}_{\alpha_{kl}} + S^I_{\alpha_{kl}} + \alpha^0_{kl}, \tag{4.26}$$

$$\beta_{kl} = S_{\beta_{kl}} - S^{(old)}_{\beta_{kl}} + S^I_{\beta_{kl}} + \beta^0_{kl}, \tag{4.27}$$

$$\delta_k = S_{\delta_k} - S^{(old)}_{\delta_k} + S^I_{\delta_k} + \delta^0_k. \tag{4.28}$$

Our adjusted update equations for the M–step have computational cost of $\mathcal{O}(K^2 E_I)$ compared with the original cost of $\mathcal{O}(K^2 E)$ of the batch algorithm 4, if $S_{\alpha_{kl}}$ and $S_{\beta_{kl}}$ are given. Thus, we save computational costs of $\mathcal{O}\left(K^2(E-E_I)\right)$ with our BlockVB M–step. In the E–step, we only have to update the cluster assignments for the active vertices, $Q_{iv}, (i,v) \in I \times \{1,\ldots,K\}$ according to proposition 4. We have computational costs of $\mathcal{O}\left(K^2 E_I\right)$ for all vertices in I whereas the computational costs for the update of the cluster assignment of all vertices in the network are $\mathcal{O}\left(K^2 E\right)$. Like in the M–step, we save computational costs of $\mathcal{O}\left(K^2(E-E_I)\right)$.
We calculate the free energy, F dependent on the updated parameters and variables and check for convergence of F.
Since we assume like in [29] that the Gamma function $\Gamma(\cdot)$ and the logarithm of the Gamma function can be calculated approximately in constant time, the computational costs of our BlockVB algorithm are dominated by the update equations in the M–and E–step.
We conclude that the computational costs for one iteration of our BlockVB algorithm are $\mathcal{O}\left(K^2 E_I\right)$ compared to $\mathcal{O}\left(K^2 E\right)$ of the batch algorithm 3. We remark that $E_I << E$ holds except for extreme cases. In all our tests we achieved a high increase of speed with our BlockVB algorithm compared to the VBEM algorithm 3 for all vertices. We sum up our BlockVB algorithm in algorithm 4.

We propose the BlockVB algorithm for the directed Bernoulli SBM in appendix A together with all update equations and the free energy model selection criterion.

Algorithm 4: BlockVB algorithm for the Poisson SBM

Data: adjacency matrix $\boldsymbol{A}$, index set of vertices for optimisation, I, cluster partition matrix $\boldsymbol{Q}$, prior parameters $\left(\boldsymbol{\alpha}^0, \boldsymbol{\beta}^0, \boldsymbol{\delta}^0\right)$, maximum number of iterations

`/* Initialisation */`

Calculate $S_{\alpha_{kl}}, S_{\beta_{kl}}, S_{\delta_k}$

Calculate $S^I_{\alpha_{kl}}, S^I_{\beta_{kl}}, S^I_{\delta_k}$

Update the parameters, $(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\delta})$, according to Propositions 2 and 3

`/* Main Loop */`

repeat

  `/* E-step:update all matrix entries of` $\boldsymbol{Q}$ `*/`

  repeat

    foreach $(i, v) \in I \times \{1, \ldots, K\}$ **do**

      Calculate Q_{iv} according to eqn. 4.12 in Proposition 4

    end

  until $\boldsymbol{Q}_{\cdot i} \forall i \in I$ *converges or maximum number of iterations*

  `/* M-step:update the model parameters */`

  foreach $(k, l) \in \{1, \ldots, K\}^2$ **do**

    `/* Preparation */`

    $S^{(old)}_{\alpha_{kl}} \longleftarrow S^I_{\alpha_{kl}}$

    $S^{(old)}_{\beta_{kl}} \longleftarrow S^I_{\beta_{kl}}$

    Calculate $S^I_{\alpha_{kl}}$ and $S^I_{\beta_{kl}}$

    `/* The Actual Update */`

    $\alpha_{kl} \longleftarrow S_{\alpha_{kl}} - S^{(old)}_{\alpha_{kl}} + S^I_{\alpha_{kl}} + \alpha^0_{kl}$

    $\beta_{kl} \longleftarrow S_{\beta_{kl}} - S^{(old)}_{\beta_{kl}} + S^I_{\beta_{kl}} + \beta^0_{kl}$

  end

  foreach $k \in \{1, \ldots, K\}$ **do**

    $S^{(old)}_{\delta_k} \longleftarrow S^I_{\delta_k}$

    Calculate $S^I_{\delta_k}$

    $\delta_k \longleftarrow S_{\delta_k} - S^{(old)}_{\delta_k} + S^I_{\delta_k} + \delta^0_k$

  end

until *convergence of* $F[q(\cdot)]$ *or maximum number of iterations*

Chapter 5

Fully Bayesian Start Values

We have discussed in sections 3.4 and 4, and seen in the literature review of e.g. [15, 50, 73] that the VBEM algorithm for the SBM converges to a local optimum dependent on the start value initialisations of the cluster assignments of the vertices. Therefore it is recommended in the literature to initialise the VBEM or VEM algorithm with different start values to find the global optimum [50, 84, 73].
The need for an initial cluster assignment of the vertices arises from the EM–like two step design of the VBEM algorithm. It follows from Algorithm 3 in section 4 (Algorithm 4 in section 4.2) and Proposition 4 that the update equations for each vertex in the E–step of the VBEM algorithm depend on the current cluster assignments of the other vertices.
In addition, we have to specify hyperparameters (parameters of the prior distribution) $(\boldsymbol{\alpha}^0, \boldsymbol{\beta}^0)$ for the Gamma or $(\boldsymbol{\zeta}^0, \boldsymbol{\eta}^0)$ for the Beta and $\boldsymbol{\delta}^0$ for the Dirichlet prior distributions for these algorithms. We also mentioned this in chapter 2, when we reviewed the Bayesian Poisson SBM.
There are two possibilities which were proposed in the literature to choose the start cluster partition, $\boldsymbol{Q}^{(start)}$, of the vertices. The first possibility is to use a second clustering algorithm with preferably lower computational costs to calculate the initial cluster assignments. Popular algorithms which were proposed for this task in the literature are the Ascending Hierarchical Clustering (AHC) algorithm based on the Ward distance [89, 35, 84, 71, 74], the K–means algorithm in combination with the AHC algorithm [73] and the spectral clustering algorithm [39, 75, 79]. We will review the AHC algorithm in section 9.1.
Then dependent on $\boldsymbol{Q}^{(start)}$, the updates of the model parameters are performed. This approach is used for the frequentist VEM batch algorithm of the SBM in [35, 84, 75, 79] except with different update equations for the model parameters. We saw during the review of the VEM algorithm in section 3.3, that there are no prior distributions of the parameters in the case of the frequentist VEM algorithm.
A similar strategy was proposed for the application of the VBEM algorithm to the Bernoulli SBM [71, 73] and the Bayesian overlapping SBM (OSBM) [71, 74]: The initial cluster assignments are calculated with a second algorithm, in this case the AHC algorithm, and fixed *non–informative* priors were chosen for the updates of the model parameters. The idea of non–informative priors is that there should be no or a low influence of the prior distributions on the inference algorithm [61, 73]. In [71, 73] non–informative *Jeffreys* priors [61] were proposed for the Bayesian Bernoulli SBM. All parameters of the Beta prior distribution of the edge existence probabilities, $\boldsymbol{\theta}$, were

set to $\zeta_{kl}^0 = \frac{1}{2},\ \eta_{kl}^0 = \frac{1}{2}, \forall k,l$ and the hyperparameters of the Dirichlet prior distribution to $\delta_k^0 = \frac{1}{2}, \forall k$ [71, 73].
We call this approach the **mixed approach**, because the start values of the Bayesian VBEM algorithm are initialised with a second algorithm like in the case of the VEM algorithm.

Random cluster assignments and informative priors Another approach, which was proposed in [50] for the restricted Bernoulli SBM (review in chapter 2), is to use a randomly initialised start partition matrix $\boldsymbol{Q}^{(start)}$ combined with fixed informative hyperparameters (see definition below) $\boldsymbol{\zeta}^0, \boldsymbol{\eta}^0$ for the Beta prior distributions of the edge existence distributions.
The start cluster assignments of the vertices, are generated uniformly at random with $\sum_{k=1}^K Q_{ik} = 1\ \forall k$ and $Q_{ik} \in [0,1]$. In the software package vbmod [49], which accompanied [50], uniform Dirichlet prior distributions are used with $\delta_k^0 = 1, \forall k$. To model the assumption that there is a higher probability for the existence of an edge within the clusters than between the clusters, the hyperparameters were set to $\zeta_{kk}^0 = 2, \eta_{kk}^0 = 1\ \forall k$ and $\zeta_{kl}^0 = 1, \eta_{kl}^0 = 2\ \forall k \neq l$ in [50].
In the algorithmic structure of the VBEM inference of vbmod, the model parameters are initialised with the hyperparameters. Then the E–step for the update of the cluster assignments is performed at the beginning of the main loop of the algorithm. Afterwards, the M–step for the update of the model parameters follows. The hyperparameters remain fixed throughout the runtime of the vbmod algorithm.
In this algorithmic approach, the hyperparameters influence the inference process and '[...] act as pseudocounts that augment observed edge counts and occupation numbers.' [50].
We remark that there are two general strategies for choosing the hyperparameters for the restricted Bernoulli SBM due to the model design: The edge connections within the clusters can be more dense or less dense then the edge connections between the clusters and the hyperparameters can be chosen accordingly.
We note, that we have to consider additional edge connection profiles in the case of the SBM, like disassortative connection profiles and hub clusters which can occur together with assortative clusters with different edge existence probabilities in the same network. Therefore, the informative hyperparameters of the restricted SBM are too limited for the SBM. We also found during preliminary numerical tests, that such a choice of priors is unsuitable for the Gamma prior distribution of the Bayesian Poisson SBM.
Because the combination of informative priors with random cluster assignments of the vertices is applied without a second, normally non–Bayesian, algorithm, we call this approach the **fully Bayesian** approach contrary to the mixed approach which was outlined above.
From now on, we will call the hyperparameters of informative prior distributions **informative hyperparameters**. We note this in the following definition.

Definition 1 (Informative Hyperparameters)**.** *Hyperparamters of informative prior distributions are called informative hyperparameters.*

Hyperparameters for Gibbs sampling For the Gibbs sampling algorithm for inference of the SBM, proposed in [97], conjugate prior distributions for the model parameters were also used. The ideas for the choice of the hyperparameters of these

prior distributions are also interesting for VBEM inference. So, we now review the choice of the informative hyperparameters for the conjugate prior Beta distributions on the edge existence probabilities $\theta_{kl}, \forall k,l$. For the case that within cluster edge connections are expected to be dense and between cluster connections to be sparse, $\zeta_{kk} = \frac{N^2}{10K^2}, \eta_{kk} = \frac{N^2}{40K^2}, \forall k$ and $\eta_{kl} = \frac{N^2}{40K^2}, \zeta_{kl} = \frac{N^2}{10K^2}$ are recommended in [97] as the hyperparameters of the Beta prior distributions. It is also noted that the number of vertices per cluster should not differ too much for the use of these informative prior distributions [97].
For the hyperparameters of the Dirichlet prior distributions on the parameters of the sizes of the clusters, π_k, hyperparameters $\delta_k^0 = 100K, \forall k$ were recommended in [97]. It was remarked in [97] that uniform hyperparameters of $\delta_k^0 = 1, \forall k$ tend to lead to unequal sizes of the clusters.
Another fully Bayesian approach with a randomly initialised fuzzy start partition matrix and informative priors for the dynamic Bernoulli SBM of [122], was used for a simulated annealing Gibbs sampler in [122]. Like in [97, 50], the informative hyperparameters of the conjugate Beta distributions of the edge existence probabilities, $\boldsymbol{\theta}$, were optimised for networks where the within cluster connection probability $\theta_{kk}, \forall k$, is expected to be higher than the inter cluster connection probability, $\theta_{kl}, \forall k \neq l$. The authors of [122] found that for these undirected example networks, hyperparameters of $\boldsymbol{\zeta}_{kk}^0$, $\forall k \in \{1, \ldots, K\}$ equal to the number of edges in the network and $\boldsymbol{\eta}_{kl}^0 = 10$; $\forall k \neq l$ yielded the best results. They also noted, that only the hyperparameters of $\boldsymbol{\zeta}^0$ (and less $\boldsymbol{\eta}^0$) are important for the inference process and the other hyperparameters can be hold fixed and non–informative or set to low values which are not meant to influence the inference process. So, the hyperparameters of the Dirichlet prior distributions were set to $\delta_k^0 = 1, \forall k$ in [122].

Hyperparameters for Gamma Priors So far we have only reviewed the hyperparameters for the Bayesian Bernoulli SBM. Now we review choices for the Gamma prior distributions of the Bayesian Poisson SBM. It is remarked in [96], that 'By construction, the Gamma distribution is informative.', [96]. The authors of [96] stated that the hyperparameters of the Gamma prior distributions for their Infinite Poisson mixture model (IPM) should have a low influence on the outcome of their Gibbs sampler. Therefore, following the literature, they proposed hyperparameters of $\alpha_{kl} = \beta_{kl} = 0.1$, $\forall k,l$ for the Gamma prior distributions on the rate parameters, $\boldsymbol{\lambda}$.
The IPM model is the weighted version of the Infinite Relational Model (IRM) [67]. The IPM is closely related to the Bayesian Poisson SBM except that the cluster assignments are determined by the Chinese Restaurant Process (CRP) instead of the Multinomial distribution.
In the implementation of the VBEM batch algorithm for the Poisson SBM in the WSBM software package [7] (downloadable at http://tuvalu.santafe.edu/∼aaronc/wsbm/), which accompanied [8], hyperparameters of $\alpha_{kl}^0 = 0, \beta_{kl}^0 = 0.001, \forall k,l$ were the default hyperparameters. In addition, it was recommended by the authors of the WSBM package to run the algorithm with different values for hyperparameters to determine their influence. Other hyperparameters suggested in the comments of the source code were uniform priors $\alpha_{kl} = \beta_{kl} = 0, \forall k,l$ or $\alpha_{kl} = 1, \beta_{kl} = 0.001, \forall k,l$.

Our Approach Contrary to the existing approaches above, we now present our findings concerning the choice of the hyperparameters for our VBEM inference of the SBM.
We remark that the hyperparameters of all the work presented above remain fixed throughout the inference process. So, if we use informative priors, the information or bias which should result from the informative priors has to be known and set up in advance. For large and complex networks which are a combination of different cluster connection profiles like densely linked clusters, hub–like clusters, sparsely connected clusters and so on, it is difficult to come up with an informative prior in advance. Otherwise we have to employ several restarts to try different hyperparameters which would lead to increased computational costs.
For our BlockVB algorithm in section 4.2, we proposed to limit the inference to subsets of vertices of networks with the cluster assignments of the other vertices held fixed.
We first discuss our choice for the Gamma prior distributions of the Bayesian Poisson SBM. Our findings presented for the Gamma prior distributions can also be transferred to case of the Bayesian Bernoulli SBM which will be discussed below.

Neutral Gamma Priors The choice of hyperparameters for the Gamma prior distribution which have a low influence on the conjugate posterior distribution was discussed in depth in [69]. The author of [69] introduced the concept of neutral hyperparameters for the conjugate Gamma and Beta distributions. These neutral hyperparameters for the conjugate Gamma distribution are given by $\alpha_{kl} = \frac{1}{3}, \forall k,l$ and $\beta_{kl} = 0, \forall k,l$. Neutral hyperparameters '[...] lead to posterior distributions with approximately 50 per cent probability that the true value is either smaller or larger than the maximum likelihood estimate.' [69].
If we choose, $\beta_{kl} = 0, \forall k,l$, we get an *improper* prior distribution [69], which leads to undefined expressions in the case of empty clusters of the VBEM algorithm (see chapter 6 for a discussion of these empty clusters). So, we have to add a tiny value to $\beta_{kl} = 0$ to cover the case of empty clusters.
We also noted during numerical tests of earthquake networks that a value for the $\beta_{kl}, \forall k,l$ parameters near to zero penalises the emergence of small clusters and thus has influence in the case of VBEM inference for the Bayesian Poisson SBM. Therefore we choose $\beta_{kl} = 0.01, \forall k,l$ as values for the hyperparameters, instead of values closer to zero or $\beta_{kl} = 0, \forall k,l$. This choice led to good results in all numerical tests.
We note that our hyperparameters are 'nearly neutral' when judged by the criteria which led to the choice of the hyperparameters proposed in [69]. For now we settle on the choice of $\alpha_{kl} = \frac{1}{3}, \forall k,l$ and $\beta_{kl} = 0.01, \forall k,l$ for the nearly neutral hyperparameters of the conjugate Gamma distributions of the Bayesian Poisson SBM.

We use the uniform Dirichlet prior distributions, $\delta_k^0 = 1, \forall k$ like in the implementation of vbmod [49, 50]. We also tried Jeffreys Dirichlet hyperparameters used for the Bernoulli SBM in [73], which did not bring any notable difference in preliminary numerical tests.
Now, that we have found hyperparameters with low impact on the inference process of the VBEM algorithm for the Poisson SBM, we have acquired two options presented above for the initialisation of the VBEM algorithm for the Bayesian Poisson SBM: The mixed approach with a second helper algorithm for the initialisations of the cluster assignments of the vertices and uninformative priors and on the other hand the random

initialisations of the start cluster assignments with informative priors.
Both approaches work for earthquake networks as we will see in the numerical tests in section 10.7. There, we will note that the mixed approach did not allow us to use every option for the selection of the active cluster in our algorithmic framework of the Blockloading algorithm 8 equally well. Astoundingly, random initialisations and nearly neutral hyperparameters can miss some network structures in the case of simple networks generated with the Poisson SBM. This applies to networks with approximately equally sized clusters and the same within and inter cluster connection rates or probabilities. This is also true for the mixed approach, although to a smaller extend.
To find a remedy for this challenge we searched for a way to combine random initialisations of the initial cluster assignments of the vertices with informative priors for the Poisson and Bernoulli SBM. We present a solution to this challenge in the next section which will also lead to improved results.

5.1 Adaptive Informative Priors: BlockVB++ Algorithm

Above, we reviewed several approaches for informative priors of a special case of the Bernoulli SBM. This special case is nearly similar to the problematic case presented above. Thus we take the existing work as a starting point.
We found that the specific choice of informative priors presented by [122] did not work for the (directed) Poisson SBM and directed Bernoulli SBM, though. Contrary to the examples of [97, 50, 122] presented above, we need prior distributions for weighted and directed networks in case of the Poisson SBM. We have to consider that the informative priors of [97, 50, 122], we reviewed above, were proposed for the special case of the SBM that the within cluster connections are more dense than the connections between the clusters.
Our Blockloading algorithm makes use of our BlockVB algorithm which limits the inference to subsets of the network. Thus, we need informative hyperparameters which take into account that the cluster assignments of many vertices remains fixed and the optimisation considers only one existing cluster at once (see chapter 8). The dependence of the VBEM inference process on informative priors is another argument why we select the existing clusters as subsets for the optimisation in our Blockloading algorithm.
In order to find informative priors for our subset based inference of our BlockVB - and Blockloading algorithm, we varied the parameter, α^0_{cc}, of only the active cluster(s) and kept the values of all other hyperparameters as nearly neutral. The idea of varying only α^0_{cc} was inspired by the finding of [122]. We tested these hyperparameters first on a restricted SBM and then also on the normal SBM as well as earthquake networks. We increased the values for the parameter, α^0_{cc}, step by step, and used randomly initialised start cluster assignments.
We found that at a certain threshold value for $\alpha^0_{kk} > T, \forall k$, the batch algorithm for the Bayesian Poisson SBM (Algorithm 3) and our Blockloading algorithm of section 8.1 were able to recover the true ground truth (true cluster assignments of the SBM), which was not possible before with the nearly neutral priors.
This threshold varied for different networks though. Therefore we have still the dependence on the network and fixed priors. Contrary to [122], we did not find a general rule like setting the hyperparameters $\alpha^0_{kk}, \forall k$ equal to the number of edges in the network.

We now introduce our new concept of priors which are automatically calculated de-

pending on the inference process. We took the number of edges in the active cluster(s) $c \in \{1,\dots,K\}$, which are given by $S_{\alpha_{cc}} = \sum_{i \neq j}^{N} Q_{ic} Q_{jc} A_{ij}$ in the case of the Bayesian Poisson SBM, (see the BlockVB algorithm 4), as a starting point to find suitable informative hyperparameters. If we choose $\alpha^0_{cc} = S_{\alpha_{cc}}$, the inference leads to the clustering of all vertices to only one cluster most of the time. Consequently, we found in tests that we can choose the hyperparameter to be a fraction of $S_{\alpha_{cc}}$. We started with at least $\frac{S_{\alpha_{cc}}}{2} = \alpha^0_{cc}, \forall k$ and tested several fractions up to $\frac{S_{\alpha_{cc}}}{S_{\alpha_{cc}}} = 1 = \alpha^0_{cc}$.
In all cases, we set all other hyperparameters of the Gamma prior distribution to nearly neutral hyperparameters. We measured the quality of the results with the free energy after convergence of the Poisson SBM of Proposition 1 in section 4.1, which we applied with nearly neutral hyperparameters $\alpha^0_{kl} = \frac{1}{3}$ and $\beta^0_{kl} = 0.01, \forall k,l \in \{1,\dots,K\}$ and uniform Dirichlet priors.
This approach is a **fully Bayesian approach** with informative hyperparameters and randomly initialised cluster assignments. We noted during experiments that our fully Bayesian approach leads to consistently better results for complicated real world networks (see also section 10.7) than the mixed approach with non informative hyperparameters or randomly initialised cluster assignments and non or neutral informative hyperparameters. We remark that a higher fraction of $S_{\alpha_{kk}}$ as the hyperparameter α^0_{kk} tends to separate the more densely connected vertices more reliably and tends to lead to better results. We found in tests that the best all purpose informative hyperparameters are $\alpha^0_{cc} = \frac{S_{\alpha_{cc}}}{4}$ for the active cluster(s), $c \in \{1,\dots,K\}$, and the new cluster in the Expansion Step. These adaptive informative hyperparameters lead to the best results for all different types of networks we tested and worked for all tested networks.

The BlockVB++ Algorithm We add our **adaptive informative hyperparameters** to our BlockVB algorithm, and start with the initialisation of the parameters before the main loop. We have two sets of hyperparameters for the conjugate Gamma distributions, the nearly neutral hyperparameters $(\boldsymbol{\alpha}^{0,(n)}, \boldsymbol{\beta}^{0,(n)})$ and the adaptive hyperparameters which are calculated depending on the random initial cluster assignments or the vertices, $\boldsymbol{Q}^{(start)}$, as explained above.
We proceed with the E–step as explained for the BlockVB algorithm in section 4, which is now dependent on $\boldsymbol{Q}^{(start)}$. Dependent on $\boldsymbol{Q}^{(start)}$, the model parameters are also randomly initialised at the beginning of the main loop contrary to the approach in [50] or [73].
After the E-step, we have updates of the cluster assignments of the subsets of active vertices, $\boldsymbol{Q}_I$, and thus the active cluster. Depending on the updated values of $\boldsymbol{Q}$, we update our adaptive informative hyperparameters $\alpha_{cc} = \sum_{i \neq j}^{N} Q_{ic} Q_{jc} A_{ij}$ in the newly introduced prior parameter update step (P–step). Dependent on the updated adaptive informative hyperparameters and the other hyperparameters, the M–step follows. We call this new algorithm, which builds upon the BlockVB algorithm, the ***BlockVB++*** algorithm.

We remark that it follows from the definition of our adaptive informative priors, that they depend on the edges connecting the vertices of the current active clusters. Therefore, our adaptive informative priors adjust themselves automatically to different within cluster edge connectivity parameters, $\lambda_{kk}, \forall k \in \{1,\dots,K\}$. There is no need to perform a separate optimisation of the hyperparameters of the prior distributions like it was dis-

cussed in [16] for the VBEM algorithm. The variational bound is also unaffected by our adaptive informative priors contrary to the prior optimisation proposed in [16].
Superficially, one could guess that our adaptive informative hyperparameters mainly apply to SBMs where the within cluster connections are more dense then the between cluster connections. So, we tested our Blockloading for networks with hub clusters, clusters which were more densely connected within than between the clusters and vice versa and clusters with irregular connection patterns all in the same network. Our Blockloading(++) algorithm together with the BlockVB++ algorithm 5 was able to identify all clusters correctly for all these cases.
We noted in the numerical tests that for the mixed approach or nearly neutral hyperparameters and random start values, more iterations for the BlockVB algorithm and the E–step of the BlockVB algorithm (see section 4.2 and Algorithm 4) were necessary to achieve good results than with our adaptive informative hyperparameters. Our BlockVB++ algorithm achieved its best result with one or two iterations in the E–step and five to ten iterations of the whole BlockVB++ algorithm contrary to the BlockVB algorithm with only nearly neutral hyperparameters which needed double the number of iterations to achieve their best results. In addition our BlockVB++ algorithm with informative hyperparameters returned noticeably better results with lower computational costs.
As mentioned above, we found that all cluster selection methods were equally viable for earthquake networks with our BlockVB++ algorithm contrary to the BlockVB algorithm.
So we have found all purpose informative hyperparameters which let us use the advantage and flexibility of randomly initialised start cluster assignments of the SBM together with the Bayesian algorithmic framework with our new P–step for our adaptive hyperparameters. This lead to noticeably improved results and computational speed due to fewer iterations as we will see section 10.7.1.

Hyperparameters for the Bayesian Bernoulli SBM We found that the findings for the informative and adaptive informative hyperparameters transfer to Bayesian Bernoulli SBM. Here we need hyperparameters for the conjugate Beta prior distributions. For non-informative hyperparameters we followed [73] and choose Jeffreys non-informative hyperparameters [61], $\zeta^0_{kl} = \eta^0_{kl} = \frac{1}{2}, \forall k,l$. We found that the advantages of our adaptive informative hyperparameters also transfer to case of the Bayesian Bernoulli SBM.
We present the BlockVB++ variant for the Bayesian Bernoulli SBM and its informative priors in appendix A.

Algorithm 5: BlockVB++ algorithm with adaptive informative priors for the Poisson SBM

Data: adjacency matrix $\boldsymbol{A}$, index set of vertices for optimisation, I, active clusters, cluster partition matrix $\boldsymbol{Q}$, neutral hyperparameters $\left(\boldsymbol{\alpha}^{0,(n)}, \boldsymbol{\beta}^{0,(n)}, \boldsymbol{\delta}^{0}\right)$, maximum number of iterations

`/* Initialisation */`

Initialise random cluster assignments for the active vertices, $\boldsymbol{Q}_I$

Calculate $S_{\alpha_{kl}}, S_{\beta_{kl}}, S_{\delta_k}$

Calculate $S^I_{\alpha_{kl}}, S^I_{\beta_{kl}}, S^I_{\delta_k}$

Set fraction of adaptive priors, $f_p \longleftarrow 4$

Initialise adaptive informative hyperparameters with $\alpha^0_{cc} = \frac{S_{\alpha_{cc}}}{f_p}, \forall c \in \{1,\ldots,K\}$

Update the parameters, $(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\delta})$, according to Propositions 2 and 3 dependent on $\left(\boldsymbol{\alpha}^{0,(n)}, \boldsymbol{\beta}^{0,(n)}, \boldsymbol{\delta}^{0}\right)$ and α^0_{cc}

`/* Main Loop */`

repeat

`/* E-step: update of cluster assignments */`

Calculate $Q_{iv}, \forall i \in I$ according to eqn. 4.12 in Proposition 4

`/* Preparation of P-and M-step */`

$S^{(old)}_{\alpha_{kl}} \longleftarrow S^I_{\alpha_{kl}}, \forall k,l$

$S^{(old)}_{\beta_{kl}} \longleftarrow S^I_{\beta_{kl}} \forall k,l$

Calculate $S^I_{\alpha_{kl}}$ and $S^I_{\beta_{kl}}$ for all k,l

`/* P-step: update of adaptive informative hyperparameters */`

$\alpha^0_{cc} = \frac{S_{\alpha_{cc}} - S^{(old)}_{\alpha_{cc}} + S^I_{\alpha_{cc}}}{f_p}$

`/* M-step:update the model parameters */`

$\alpha_{kl} \longleftarrow S_{\alpha_{kl}} - S^{(old)}_{\alpha_{kl}} + S^I_{\alpha_{kl}} + \alpha^0_{kl}$ for all k,l

$\beta_{kl} \longleftarrow S_{\beta_{kl}} - S^{(old)}_{\beta_{kl}} + S^I_{\beta_{kl}} + \beta^0_{kl}$ for all k,l

$S^{(old)}_{\delta_k} \longleftarrow S^I_{\delta_k}, \forall k$

Calculate $S^I_{\delta_k}, \forall k$

$\delta_k \longleftarrow S_{\delta_k} - S^{(old)}_{\delta_k} + S^I_{\delta_k} + \delta^0_k, \forall k$

until *convergence of* $F\left[q(\cdot)\right]$ *or maximum number of iterations*

Chapter 6

Model Selection Criteria for the Stochastic Block Model

We address the task of model selection for the SBM where we want to find the optimal combination of the number of clusters, parameters and cluster assignments of the vertices. The efficient identification of the optimal number of clusters of the SBM with respect to a model selection criterion was one of our objectives for the introduction of our divisive Blockloading algorithms in chapter 8. We will see in chapter 8, that the choice of the model selection criterion is of importance for our Blockloading algorithm and its variants.
The values of the inferred cluster assignment, the model parameters and the number of clusters influence the value of the model selection criterion.
In the context of the Stochastic Block Model (SBM) model selection is the task of finding the optimal number clusters, $K^\star$ of the SBM. The VBEM algorithm in chapter 4 is initialised with a fixed number of clusters K.
The optimal (true) number of clusters K^* is unknown.
We now review the general application of such a model selection criterion together with the batch algorithm which was the only way to find the optimal model in the context of the SBM for several years [35, 50, 123, 84, 73, 74].
Then we will review and present the most important and popular model selection criteria for the SBM. We begin this review in section 6.2 with the explication why the converged free energy of the Poisson SBM we introduced in Proposition 1 can be used as a model selection criterion. We continue with the presentation of our new ICL_{ex} model selection criterion for the Bayesian Poisson SBM in section 6.3. The ICL_{ex} was first proposed for the Bernoulli SBM in [29] and for the Poisson version among others of the bipartite Latent Block Model [43] in [120]. To the best of our knowledge the ICL_{ex} for Poisson SBM was not proposed yet.
Finally, we review the Integrated Classification Likelihood criterion in section 6.4. The ICL was proposed in [18] and adopted to the Bernoulli SBM in [35] and the Poisson SBM in [84]. We discuss the properties of each model selection criterion in the respective section.

Akaike and Bayesian Information Criterion Two important model selection criteria, the Akaike Information Criterion (AIC) [11, 19] and the Bayesian Information Criterion (BIC) [106, 19] cannot be applied to the SBM because both criteria depend

on the likelihood of the observed data, $p(\boldsymbol{A} \mid \boldsymbol{\vartheta})$ [84, 71, 73]. It was already mentioned in section 3.2, that calculation of the log–likelihood of the observed data by marginalisation, $\ln p(\boldsymbol{A} \mid \boldsymbol{\vartheta}) = \sum_{\boldsymbol{Z}} \ln p(\boldsymbol{A}, \boldsymbol{Z} \mid \boldsymbol{\vartheta})$, leads to a summation over K^N terms, which is the number of all possible cluster assignments, where K is the number of clusters and N the number of vertices of the network [84]. Likewise, all other model selection criteria which rely on calculation of the term $\ln p(\boldsymbol{A} \mid \boldsymbol{\vartheta})$ are not tractable in the case of the SBM [84].

Empty Clusters and the VBEM algorithm The VBEM algorithm does not assign any vertices to unneeded clusters and reassigns vertices from redundant clusters to other clusters during the optimisation process [14, 31, 50]. These redundant clusters are returned as empty clusters and can be discarded after convergence of the algorithm. If vertices were assigned to such a redundant cluster as start values (see also section 4), the VBEM algorithm actively removes these vertices from the superfluous clusters during the inference process, dependent on these start values [14, 31, 50].
This feature was used in the context of the variational Bayesian EM estimation of mixture distributions in [14, 31, 111, 86]. It was also noted to hold for VBEM inference of the restricted SBM in [50]. We confirmed during tests, that this feature of VBEM inference also works for the Bernoulli and Poisson SBM.
An algorithm for model–selection of mixture distributions was designed around this feature of VBEM inference in [86]. We remark that using this feature of VBEM exclusively for model selection of the SBM increases the computational costs because at the start of the inference vertices are assigned to clusters which might be found to be redundant during the inference.
Using this feature for model selection of the SBM is also highly dependent on the choice of start values for the cluster assignment of the vertices. Therefore, such a model selection procedure does not work well for bigger and more complex networks and we will see that there are better algorithms for model selection of the SBM in chapter 8. Nevertheless this feature of the VBEM algorithm supports other ways of model selection we will present below.

6.1 Model Selection for Batch Algorithms with fixed number of clusters

The method to use the following model selection criteria together with a batch algorithm for the SBM is always the same (e.g. [35, 50, 123, 84, 73, 74]): The batch algorithm is run for every number of clusters, K, which we guess that might be optimal, for different initialisations of the start values. Then we choose the result of all these runs, which yields the best value of our model selection criterion. We need several initialisations of the algorithm because of the possibility of local optima (see section 4). We sum up this procedure in Algorithm 6.
We mentioned in the introduction, that Algorithm 6 is computationally expensive, especially if high numbers of clusters are involved. In chapter 8, we will discuss our Blockloading framework and other methods which are able to avoid these costly re–initialisations with different cluster sizes.
We denote with $K_{\max}$ the highest number of clusters the batch algorithm is initialised with during the search for an optimum. Under the assumption that the model selec-

tion criterion can be calculated in linear time, the order of the computational costs for the VBEM batch algorithm for the undirected Bernoulli SBM was found to be of $\mathcal{O}(K_{\max}^3 E)$ [29]. Under the same assumption, we found the order of the computational costs for Algorithm 6 together with our Poisson VBEM batch algorithm also to be $\mathcal{O}(K_{\max}^3 E)$. The same is true for VEM version of the batch algorithm for the Poisson or Bernoulli SBM because these algorithms are of the same order as the two algorithms above as discussed in section 4, [35, 84, 30].

Algorithm 6: Model Selection for the SBM with fixed number of clusters (Batch algorithm)

```
/* Initialisation */
```
Choose number of restarts, R, with different start values and set of number of clusters for inference, $k \in \{K_1, \ldots, K_n\}$
for $r = 1$ **to** R **do**
  foreach $k \in \{K_1, \ldots, K_n\}$ **do**
    Run the batch algorithm 3 with k clusters
    Calculate the model selection criterion, $\mathscr{F}_{rk}$, for the result of the run r
  end
end
$\mathscr{F}^{\star}_{r^{\star}K^{\star}} \longleftarrow \min_{\substack{r\in\{1,\ldots,R\}\\ k\in\{K_1,\ldots,K_n\}}} \mathscr{F}_{rk}$
Result: Optimal number of clusters, $K^{\star}$, and value of model selection criterion, $\mathscr{F}^{\star}_{r^{\star}K^{\star}}$

6.2 Model Selection with the converged free energy or ILvb

We have already seen in section 3.4, that the variational bound, $\mathscr{L}(\cdot)$, of the the marginal log–likelihood, $\ln p(\boldsymbol{A}\,|\,K)$, penalises the model complexity [15]. We presented the variational bound or free energy, F, for the Bayesian Poisson SBM in chapter 4.

The application of the converged variational bound for model selection of the restricted Bernoulli SBM was proposed in [50]. It was taken as an upper bound of the negative log–likelihood, $-\ln p(\boldsymbol{A}\,|\,K)$, and called *free energy* because this method originates from statistical physics [38]. The motivation which lead to the use of the converged free energy for model selection of the restricted Bernoulli SBM was that the optimisation problem to find the optimal number of clusters, $K^{\star}$, can be stated as [50]:

$$K^{\star} = \arg\max_{K} p(K\,|\,\boldsymbol{A}). \tag{6.1}$$

The VBEM framework allows the optimisation of the variational bound of

$$-\ln p(\boldsymbol{A}\,|\,K) = -\ln \sum_{\boldsymbol{Z}} \int p(\boldsymbol{A}, \boldsymbol{\vartheta}, \boldsymbol{Z}\,|\,K)\mathrm{d}\boldsymbol{\vartheta}. \tag{6.2}$$

Now, it was used in [50], that by Bayes formula (see e.g. [19]) it follows that

$$p(K\,|\,\boldsymbol{A}) = \frac{p(\boldsymbol{A}\,|\,K)p(K)}{p(\boldsymbol{A})} \tag{6.3}$$

$$\propto p(\boldsymbol{A}\,|\,K)p(K). \tag{6.4}$$

If there is no known prior information, $p(K)$, about the number of clusters, it is assumed that $p(K)$ is a uniform distribution. So, all numbers of clusters, K_i, $i = K_{i_1}, \ldots, K_{i_r}$ are equally likely. Thus, it follows from eqn. (6.4) that $p(K\,|\,\boldsymbol{A}) \propto p(\boldsymbol{A}\,|\,K)$ [50].
Therefore optimising the likelihood (or evidence) $p(\boldsymbol{A}\,|\,K)$ is equivalent to optimising the posterior $p(K\,|\,\boldsymbol{A})$, if there is no prior information about the number of clusters available or used [50]. The need for the VBEM method to approximate the marginalised log–likelihood, $\ln p(\boldsymbol{A}\,|\,K)$ was explained in section 3.4.
It is noted in [50], that this justification for model selection is a general principle which follows from the theory presented in [66]. This framework is known as *Bayes factors* in the literature [66] and was originally proposed in [60].
Using the converged variational bound of the integrated log–likelihoood after convergence of the VBEM algorithm for model selection was also proposed for the Bernoulli SBM in [71, 73] and for the overlapping SBM in [74]. This model selection criterion was called *Integrated Likelihood variational Bayes (ILvB)* in [73].
It was added in [24, 73], that the quality of the approximation of the integrated log–likelihood, which is given by the Kullback–Leibler divergence (see [19] and section 3.3), is unknown in practice.
The ILvB (converged free energy) is a a *non–asymptotic* model selection criterion in contrast to the Integrated Classification Likelihood (ICL) criterion [18], which was also adapted to the SBM in [35, 84]. We review the ICL criterion for the SBM in section 6.4.
We repeat (see section 3.4), that no Hessian matrix is needed for the converged free energy in contrast to other model selection criteria [15].
We have seen above, that the free energy was stated for the restricted ([50]) the (normal) Bernoulli SBM [71, 73] and the overlapping SBM [74]. All these SBM have in common that they model undirected and unweighted networks.
We have already stated the converged free energy of the Poisson SBM in Proposition 1 as the variational bound after convergence of the VBEM algorithm 3 for the inference of the Bayesian Poisson SBM. In this section in addition to section 3.4, the explanation why it can be used as a model selection criterion was added.
We remark, that in our VBEM algorithm for the Poisson SBM of section 4, the free energy is also the objective function, which is optimised during the inference process of the VBEM algorithm. The other model selection criteria differ from the objective function.
We discussed suitable choices for the hyperparameters of the prior distributions of the Bayesian Poisson SBM which also influence the value of the converged free energy in chapter 5. Care has to be taken for the choice of the hyperparameters for the free energy of the Poisson SBM because the gamma prior distributions of the parameters $\boldsymbol{\lambda}$ are informative by construction [96].

6.3 Exact Integrated Classification Likelihood Criterion

In this chapter we derive another model selection for the Bayesian Poisson SBM, the ICL_{ex}. The ICL_{ex} was proposed in [29] for the Bayesian Bernoulli SBM.

The ICL_{ex} criterion is also the basis for the greedyICL algorithm for optimisation of the ICL_{ex} for a given graph in [29]. The greedyICL algorithm uses greedy heuristics introduced in [95] and [21], [29]. We will review the greedyICL algorithm in chapter 7.

Contrary to the free energy (ILvb) model selection criterion or the ICL criterion we review below, the ICL_{ex} is not an approximation and only assumes factorised prior distributions. Nevertheless we will now see that the free energy - and ICL criterion are closely related [29].

We have seen in chapter 4 that the converged free energy (ILvb) is used for the fuzzy cluster assignments returned by the updates in proposition 4. We recall that fuzzy assignments of the vertices give the probability $Q_{ik} \in [0,1]$ of the cluster assignment of vertex i to cluster k. It holds that $\sum_{k=1}^{K} Q_{ik} = 1$.

We can also use the free energy (ILvb) criterion for a hard clustering of the vertices, where each vertex is assigned to exactly one cluster with probability one. In this case, the entropy term, $\sum_{i=1}^{N}\sum_{x=1}^{K} Q_{ix} \ln Q_{ix}$, of eqn. (4.4) is equal to zero. The ILvb with the entropy term equal to zero is similar to the exact Integrated Classification Likelihood Criterion (ICL_{ex}), which was introduced in [29].

The derivation of the ICL_{ex} differs from the derivation of the free energy (ILvb). This was shown in [29] for the Bernoulli SBM.

We also show this in our proof of the ICL_{ex} for the Poisson SBM in Proposition 5 below.

The free energy (ILvb) model selection criterion is an approximation of the negative marginal log-likelihood, $-\ln p(\boldsymbol{A}|K)$. On the contrary, the ICL_{ex} is no approximation with a variational bound, but an analytical model selection criterion for the SBM [29]. It takes the cluster indicator matrix $\boldsymbol{Z}$, which is a hard cluster assignment, and the number of clusters K as the input.

We present the ICL_{ex} in Proposition 5. The ICL_{ex} model selection criterion of the Poisson version of the Latent Block Model (LBM) [43] for bipartite networks was stated in [120] without a proof.

Like the Variational Bayesian approximation in chapter 4, the ICL_{ex} builds on the Bayesian framework of the Poisson SBM [73, 96], we reviewed in section 2.2. We use the same prior distribution as in section 2.2 or chapter 4.

Proposition 5. *Let the cluster indicator matrix $\boldsymbol{Z}$, the number of clusters K and the adjacency matrix $\boldsymbol{A}$ be given. Under the assumption that the factorisation $p(\boldsymbol{\lambda},\boldsymbol{\pi}) = p(\boldsymbol{\lambda})p(\boldsymbol{\pi})$ holds, the ICL_{ex} of the Poisson Stochastic Block Model of eqn. 2.5 is given by*

$$ICL_{ex}[\boldsymbol{Z},K] = \ln p(\boldsymbol{A},\boldsymbol{Z}\,|\,K) = \sum_{k,l}^{K} \ln\left(\frac{\beta_{kl}^{\alpha_{kl}}\Gamma(\alpha_{kl}^0)}{\beta_{kl}^{0\,\alpha_{kl}^0}\Gamma(\alpha_{kl})}\frac{1}{(A_{ij}!)^{\sum_{i\neq j}^{N} Z_{ik}Z_{jl}}}\right) + \ln\left(\frac{\Gamma\left(\sum_{k=1}^{K}\delta_k\right)\prod_{k=1}^{K}\Gamma(\delta_k^0)}{\Gamma\left(\sum_{k=1}^{K}\delta_k^0\right)\prod_{k=1}^{K}\Gamma(\delta_k)}\right), \tag{6.5}$$

where $\Gamma(\cdot)$ is the Gamma function. The parameters α_{kl},β_{kl} and δ_k for all $k,l = 1,\ldots,K$ are calculated according to $\alpha_{kl} \equiv \sum_{i\neq j} Z_{ik}Z_{jl}A_{ij} + \alpha_{kl}^0\ \forall k,l$, $\beta_{kl} \equiv \sum_{i\neq j} Z_{ik}Z_{jl} + \beta_{kl}^0\ \forall k,l$ and $\delta_k \equiv \sum_{i=1}^{N} Z_{ik} + \delta_k^0\ \forall k$. The parameters $\alpha_{kl}^0,\beta_{kl}^0$ and δ_k^0 are the hyperparameters.

Proof. We assume factorized prior distributions $p(\boldsymbol{\lambda}) = \prod_{k,l}^{K} \text{Gamma}(\lambda_{kl};\alpha_{kl}^0,\beta_{kl}^0)$ and $p(\boldsymbol{\pi}) = \text{Dir}(\boldsymbol{\pi};\boldsymbol{\delta}^0 = (\delta_1^0,\ldots,\delta_k^0))$. Thus, the integrated complete data log likeli-

hood can written in the following way [18, 29]:

$$\ln p(\boldsymbol{A},\boldsymbol{Z}|K) = \ln\left(\int_{\boldsymbol{\pi},\boldsymbol{\lambda}} p(\boldsymbol{A},\boldsymbol{Z},\boldsymbol{\pi}|K)\mathrm{d}\boldsymbol{\pi}\mathrm{d}\boldsymbol{\lambda}\right) \tag{6.6}$$

$$= \ln\left(\int_{\boldsymbol{\lambda}} p(\boldsymbol{A}|\boldsymbol{Z},\boldsymbol{\lambda},K)p(\boldsymbol{\lambda}|K)\mathrm{d}\boldsymbol{\lambda}\int_{\boldsymbol{\pi}} p(\boldsymbol{Z}|\boldsymbol{\pi},K)p(\boldsymbol{\pi}|K)\mathrm{d}\boldsymbol{\pi}\right) \tag{6.7}$$

$$= \ln p(\boldsymbol{A}|\boldsymbol{Z},K) + \ln p(\boldsymbol{Z}|K). \tag{6.8}$$

In the following, we use the abbreviations

$$\alpha_{kl} \equiv \sum_{i\neq j} Z_{ik}Z_{jl}A_{ij} + \alpha^0_{kl}\ \forall k,l, \tag{6.9}$$

$$\beta_{kl} \equiv \sum_{i\neq j} Z_{ik}Z_{jl} + \beta^0_{kl}\ \forall k,l, \tag{6.10}$$

$$\delta_k \equiv \sum_{i=1}^{N} Z_{ik} + \delta^0_k\ \forall k. \tag{6.11}$$

The term $p(\boldsymbol{A}|\boldsymbol{Z},K)$ is together with eqn. (2.5):

$$p(\boldsymbol{A}|\boldsymbol{Z},K) = \int_{\boldsymbol{\lambda}} p(\boldsymbol{A}|\boldsymbol{Z},\boldsymbol{\lambda},K)p(\boldsymbol{\lambda}|K)\mathrm{d}\boldsymbol{\lambda} \tag{6.12}$$

$$= \int_{\boldsymbol{\lambda}} \left(\prod_{k,l}^{K} \frac{\lambda_{kl}^{\sum_{i\neq j}^{N} Z_{ik}Z_{jl}A_{ij}}}{(A_{ij}!)^{\sum_{i\neq j}^{N} Z_{ik}Z_{jl}}} \exp\left(-\lambda_{kl}\left(\sum_{i\neq j}^{N} Z_{ik}Z_{jl}\right)\right)\right)$$

$$\times \prod_{k,l}^{K} \frac{{\beta^0_{kl}}^{\alpha^0_{kl}}}{\Gamma(\alpha^0_{kl})}\lambda_{kl}^{\alpha^0_{kl}-1}e^{-\beta^0_{kl}\lambda_{kl}}\mathrm{d}\boldsymbol{\lambda} \tag{6.13}$$

$$= \int_{\boldsymbol{\lambda}} \prod_{k,l}^{K} \frac{{\beta^0_{kl}}^{\alpha^0_{kl}}}{\Gamma(\alpha^0_{kl})}\lambda_{kl}^{\sum_{i\neq j}^{N} Z_{ik}Z_{jl}A_{ij}+\alpha^0_{kl}-1} \frac{1}{(A_{ij}!)^{\sum_{i\neq j}^{N} Z_{ik}Z_{jl}}}$$

$$\exp\left(-\left(\sum_{i\neq j} Z_{ik}Z_{jl} + \beta^0_{kl}\right)\lambda_{kl}\right)\mathrm{d}\boldsymbol{\lambda} \tag{6.14}$$

$$= \prod_{k,l}^{K} \frac{\Gamma(\alpha_{kl})}{\Gamma(\alpha^0_{kl})} \frac{{\beta^0_{kl}}^{\alpha^0_{kl}}}{\beta_{kl}^{\alpha_{kl}}} \frac{1}{(A_{ij}!)^{\sum_{i\neq j}^{N} Z_{ik}Z_{jl}}} \tag{6.15}$$

In [29] it was shown that the term $p(\boldsymbol{Z}|K)$ is:

$$p(\boldsymbol{Z}|K) = \int_{\boldsymbol{\pi}} p(\boldsymbol{Z}|\boldsymbol{\pi},K)p(\boldsymbol{\pi}|K)\mathrm{d}\boldsymbol{\pi} \tag{6.16}$$

$$= \int_{\boldsymbol{\pi}} \left(\prod_{k=1}^{K} \pi_k^{\sum_{i=1}^{K} Z_{ik}}\right) \frac{1}{C(\boldsymbol{\delta}^0)} \prod_{k=1}^{K} \pi_k^{\delta^0_k-1}\mathrm{d}\boldsymbol{\pi} \tag{6.17}$$

$$= \frac{C(\boldsymbol{\delta})}{C(\boldsymbol{\delta}^0)} \int_{\boldsymbol{\pi}} \mathrm{Dir}(\boldsymbol{\pi};\boldsymbol{\delta})\mathrm{d}\boldsymbol{\delta} \tag{6.18}$$

$$= \frac{C(\boldsymbol{\delta})}{C(\boldsymbol{\delta}^0)}, \tag{6.19}$$

where we used the abbreviation $C(\boldsymbol{\delta}) = \frac{\prod_{k=1}^{K}\Gamma(\delta_k)}{\Gamma(\sum_{l=1}^{K}\delta_k)}$ for $\boldsymbol{\delta} \in \mathbb{R}^K$. We take the negative logarithm of Eqn. (6.15) and eqn. (6.19) together with eqn. (6.8), this yields eqn. (6.5). □

Every time we will use the free energy or ILvb criterion in this thesis one can also use the ICL_{ex}. Of course fuzzy cluster assignments, $\boldsymbol{Q}$, have to be transformed to hard clustering, $\boldsymbol{Z}$. We describe this transformation in chapter 8. This comes at a slightly increased computational cost.
Non-informative priors of [61] are used for the ICL_{ex} of the Bernoulli SBM in [29]. The discussion of low influence hyperparameters for the prior distributions over the model parameters in chapter 5 also applies to ICL_{ex} of the Poisson SBM.
It is remarked in [29], that a Stirling approximation of the Gamma function of the ICL_{ex} of the Bernoulli SBM shows that the ICL_{ex} penalises model complexity. The same is true for the Gamma function used of our ICL_{ex} criterion for the Poisson SBM in eqn. 6.5.

6.4 Integrated Classification Likelihood Criterion

The Integrated Classification Likelihood (ICL) criterion was introduced in [18] for the model selection of mixture distributions. The ICL criterion was shown to be less likely to overestimate the number of clusters, K, in mixture inference problems than the Bayesian Information Criterion (BIC) [18]. The ICL was adapted to the Bernoulli SBM in [35] to determine the optimal number of clusters with a VEM batch algorithm. Building upon [35], the ICL criterion for the directed Poisson SBM was proposed in [84]. We follow the presentation of [35], [18], [84] and [29].

We recall that the free energy is an approximation of the marginal log-likelihood (integrated observed-data likelihood), $p(\boldsymbol{A}|K)$ [73]. The principle idea of the ICL for the SBM is to approximate the integrated likelihood of the complete data, $(\boldsymbol{A},\boldsymbol{Z})$, (integrated classification likelihood) which is given by [18, 35]:

$$p(\boldsymbol{A},\boldsymbol{Z}|K) = \int p(\boldsymbol{A},\boldsymbol{Z}|\boldsymbol{\vartheta},K)p(\boldsymbol{\vartheta}|K)\mathrm{d}\,\boldsymbol{\vartheta}. \tag{6.20}$$

The term $p(\boldsymbol{\vartheta}|K)$ in eqn. 6.20 is the prior distribution over the model parameters $\boldsymbol{\vartheta}$. We have seen above that this likelihood is intractable and has to be approximated.
The derivation of of the ICL is based on a Lemma of [18]: If we assume that $p(\boldsymbol{\lambda},\boldsymbol{\pi}|K) = p(\boldsymbol{\lambda}|K)p(\boldsymbol{\pi}|K)$ holds, it follows that $\ln p(\boldsymbol{A},\boldsymbol{Z}|K) = \ln p(\boldsymbol{Z}|K) + \ln p(\boldsymbol{A}|\boldsymbol{Z},K)$. The proof of this Lemma was repeated in equations 6.6 to 6.8 in Proposition 5.
To find a tractable expression for the term $\ln p(\boldsymbol{Z}|K)$, a Dirichlet prior distribution is set for $p(\boldsymbol{\pi}|K)$, with Jeffreys priors, $\delta_k^0 = \frac{1}{2}\ \forall k$, and the same calculation as in the equations 6.16 to 6.19 of the proof of Proposition 5 is done [18, 35]. Thus, the ICL is a Bayesian model selection criterion [35].Then the Stirling approximation in the limit of large N, where N is the number of vertices of the network, is used. This leads to [35, 84]:

$$\ln p(\boldsymbol{Z}|K) \approx \max_{\boldsymbol{\pi}} p(\boldsymbol{Z}|\boldsymbol{\pi},K) - \frac{(K-1)}{2}\ln(N). \tag{6.21}$$

This expression was derived in [35] in the case of the Bernoulli SBM but is the same for other versions of the SBM with the Multinomial distribution for the cluster assignments like e.g. the Poisson SBM of [84].
For the term $\ln p(\boldsymbol{A}|\boldsymbol{Z},K)$, a BIC-like Laplace approximation is used [18, 35] which leads to [18, 35, 29]

$$\ln p(\boldsymbol{A}|\boldsymbol{Z},K) \approx \max_{\boldsymbol{\lambda}} p(\boldsymbol{A}|\boldsymbol{Z},\boldsymbol{\lambda},K) - \frac{K^2}{2}\ln(N(N-1)). \tag{6.22}$$

For a detailed derivation of these two approximations we refer to [18] and [35].

The ICL is used with the frequentist VEM point estimators of the model parameters in [35] and [84]. These VEM estimators, $\hat{\boldsymbol{\vartheta}} = (\hat{\boldsymbol{\pi}}, \hat{\boldsymbol{\lambda}})$, for the model parameters $\boldsymbol{\vartheta} = (\boldsymbol{\pi}, \boldsymbol{\lambda})$ of the Poisson SBM are given by [84]:

$$\hat{\pi}_k = \frac{1}{N}\sum_{i=1}^{N} Q_{ik} \; \forall k, \tag{6.23}$$

and

$$\hat{\lambda}_{kl} = \frac{\sum_{i\neq j} A_{ij} Q_{ik} Q_{jl}}{\sum_{i\neq j} Q_{ik} Q_{jl}} \; \forall k,l, \tag{6.24}$$

where $\mathbf{Q}$ is the estimated cluster assignment returned by the chosen inference algorithm and $\mathbf{A}$ the adjacency matrix of the network. We define $f(A_{ij},\lambda_{kl}) = \frac{\lambda_{kl}^{A_{ij}}}{A_{ij}!}\exp(-\lambda_{kl})$ to be the Poisson distribution. It was remarked in [18] that also Bayesian or other parameter estimates can be used instead of the of the VEM estimators stated above.

The combination of eqn. 6.21 and 6.22 leads to the ICL criterion for the directed Poisson SBM which is now given by [84, 29]:

$$ICL(\boldsymbol{Q},K) = \max_{\boldsymbol{\vartheta}} \ln p\left(\boldsymbol{A},\boldsymbol{Q}|\hat{\boldsymbol{\vartheta}},K\right) - \frac{1}{2}K^2\ln(N(N-1)) - \frac{K-1}{2}\ln(N) \tag{6.25}$$

$$= \sum_i\sum_k Q_{ik}\ln\hat{\pi}_k + \sum_{i\neq j}\sum_{k,l} Q_{ik}Q_{jl}\ln f(A_{ij},\hat{\lambda}_{kl}) - \frac{1}{2}K^2\ln(N(N-1)) - \frac{K-1}{2}\ln(N). \tag{6.26}$$

The term $-\frac{1}{2}K^2\ln(N(N-1)) - \frac{K-1}{2}\ln(N)$ is a penalty for model complexity and similar to term used in the BIC [84]. The ICL is an asymptotic model selection criterion in contrast to the converged free energy in section 6.2 or the ICL_{ex} in section 6.3. It was remarked in [84], that the ICL could underestimate the number of clusters for small synthetic (generated with the SBM) graphs with $N = 50$ vertices but worked almost always for synthetic graphs with more than $N = 100$ vertices.
The same observation was found in [73] where the converged free energy/ILvb of the Bernoulli SBM was compared to the ICL for the Bernoulli SBM in tests of small graphs with $N = 50$ vertices which were generated by the Bernoulli SBM. The converged free energy/ILvb outperformed the ICL in those tests in [73]. It was remarked in [30] based on [18, 84] that '[...] analyses based on the ICL tend to miss certain important structures present in the data because of the use of (asymptotic) approximations.' [30].

This result is explained in [73] by the different approaches of the two model selection criteria. It is remarked that '[...] ILvb approximates the marginal likelihood, which is known to focus on density estimation only.' [73] and that the 'ICL approximates the integrated *complete-data* likelihood and is known to focus on cluster analysis view since it favours well- separated clusters.' [73].

The ICL is a widely used model selection criterion for the SBM [35, 124, 84, 123] and is also used in the software package Wmixnet [75] and its successor the Blockmodels package [79] which is a state of the art implementation of a split-merge VEM algorithm for different versions of the SBM.
We also remark that we do not need any hyperparameters of the priors of the model parameters, $\boldsymbol{\lambda}$, for the ICL criterion for the Poisson SBM, if we use the frequentist estimators given in eqn. 6.24. This is an advantage when compared to the free energy or ICL_{ex} criterion for the Bayesian Poisson SBM which both rely on Bayesian estimators for the parameters. As mentioned above, we discuss the choice of these hyperparameters for the Gamma prior distributions of $\boldsymbol{\lambda}$ in chapter 5. We have to be careful with the choice of these hyperparameters of the priors in the case of the free energy or the ICL_{ex} because the Gamma distribution is informative by construction [96].

Chapter 7

Review of Split–Merge concepts for clustering

The idea of restricting the optimisation process of a clustering algorithm to subsets of the vertices of the network is a recurrent one in the literature. This restriction to subsets is often combined with a split–merge (or split–join) algorithm where a model selection criterion or cost function is optimised through splitting and merging as well as optimising subsets of the network. Now, we will review selected past work where such algorithms were proposed.

Based on [56] and [123], we first introduce some terminology to distinguish different types of divisive and agglomerative algorithms. For the sake of convenience we also recall important facts about split, merge and subset algorithms from the introduction of the thesis and chapter 1.

The following discussion of algorithms assumes that the optimal number of clusters is unknown, that the cluster assignment is *exclusive* and that the optimal assignment of the vertices to the unknown number of clusters is hidden. Therefore these algorithms are exclusive, unsupervised classification (clustering) algorithms [56].

If the inference process starts with all vertices in one cluster and there is a split proposal to split an existing cluster into two or more new clusters to improve a model selection criterion, this algorithm is called a *divisive* algorithm [56]. The inference of a a divisive algorithm stops if no further improvement of the model selection is achieved with a split or merge step or the desired maximum number of clusters was reached. Such a procedure was used for example for the bisecting K-means [109] or X-means algorithm [99]. It can also be considered to split several clusters simultaneously [99]. Of course a divisive algorithmic scheme can be applied to an existing partition of more than one cluster (e.g. [109, 57, 75, 118]).

The opposite strategy consists of considering each vertex as a separate cluster and then to start to merge clusters to optimise a model selection criterion. Such an approach is called *agglomerative* clustering. Again, the agglomerative algorithm can also be applied to any more than two clusters [29, 121].

Both, agglomerative and divisive clustering algorithms are *hierarchical* clustering algorithms [56]. The inference process of hierarchical clustering algorithms can be visualised by a tree like structure called *dendrogram* [123]. The root of the dendrogram is the starting cluster of the divisive algorithm to which all vertices are assigned. The leafs of the dendrogram are the single vertex clusters or the initial cluster partition be-

fore the application of the merge steps of the agglomerative algorithm.
Agglomerative and divisive clustering can be combined into a split–merge hierarchical algorithm [32, 57, 79].
It was noted in [99] that we also need to specify a criterion to choose the cluster or clusters we want to split. The same is true when we select vertices or clusters to merge in the agglomerative case. This choice is often performed randomly in the case of merge steps [57, 89] but there also exist algorithms with more sophisticated criteria [103].
Split and merge algorithms focus naturally on subsets of the data. There are also algorithms which optimise subsets of the data and keep the number of clusters fixed like the Stochastic Variational Inference (SVI) algorithm [48, 41].
We limit the following concise review of split, merge and subset based methods to the general properties of the chosen inference algorithm and focus on the split and merge algorithm of these methods.

7.1 Variational Subset Based Inference for Clustering

7.1.1 Online clustering for variational algorithms

The computational limitations of the variational batch algorithm for inference of the SBM for large networks were addressed with an online clustering strategy for the Classification EM [124] and the VEM algorithm, [125, 123]. Online strategies are used to calculate the optimal cluster assignments of newly added vertices of a growing network, where the existing cluster assignment of the old (existing) vertices stays fixed. Once a cluster assignment of a vertex is calculated it stays the same until the end of the inference process [124, 125, 123].
Online clustering can also be used to considerably lower the computational costs of the inference process of a fixed network. To apply online clustering to a fixed network, the inference process is started with a low number vertices and then the remaining vertices are added vertex by vertex to the network and their cluster assignments are calculated according to the online clustering algorithm. It was shown in [125] that such a procedure can lead to a considerable computational speed up of the inference process when compared to the variational batch algorithm. This speed up comes at the cost of a lower quality of the results for some networks [125]. The online algorithm is initialised with a fixed number of clusters before the start of the inference process like the batch algorithm [125]. The variational algorithm is also notable for limiting the inference process to subsets of the network. We remark is that the online algorithms in [124, 125, 123] are frequentist algorithms contrary to our Bayesian BlockVB and BlockVB++ algorithms.

7.1.2 Stochastic Variational Inference

To improve upon the variational batch algorithm, the concept of Stochastic Variational Inference (SVI) [48], which is based on Stochastic Optimisation [100], was proposed first for the Mixed-Membership Stochastic Block Model (MMSB) [42, 41] and then for the Bernoulli SBM [39]. The idea of SVI is to limit the inference to subsets of the data and to optimise a noisy approximation of the variational bound which provably converges to a local optimum of non-convex optimisation problems [48], which is the case for the MMSB or SBM [41] and to the optimum in case of convex optimisation problems [100, 48].
It was shown with numerical examples in [41] that the use of SVI for the inference on

the MMSB lead to a much faster convergence and better quality of the results than the application of the batch approach. In the case of the Bernoulli SBM in [39], it was also shown to have a much faster convergence than the traditional batch algorithm without SVI for very large synthetic networks, although it did not necessarily lead to a better quality of the results for large networks. It was concluded in [39] regarding the quality of the results of SVI, that: 'Our experiments make it clear that our stochastic algorithm is able to perform at least as well as the traditional batch algorithm.' [39].
The application of the SVI inference to the SBM in [39] still depends on the Spectral Clustering (or a second algorithm for the start cluster initialisation) together with an online version of K-means to initialise the start values for the cluster assignments of the vertices, to achieve its speed of convergence. It was noted that the SVI inference algorithm '[...], instead of initializing our methods entirely randomly, which tended to yield bad results, we instead initialize using a simpler algorithm run for a brief time (spectral clustering from Section 2)' [39].
The SVI algorithm is only applied to a fixed number of clusters in [39] and does not reuse cluster assignments previously determined for lower number of clusters. So the main critic of the batch approach of wasting computational resources to find the optimal number of clusters, also applies to the SVI algorithm for the SBM.
There is also additional user input needed to determine the step size of the SVI algorithm and the user defined size of the subsets [48, 41, 39]. Different schemes were proposed to sample these vertices for SVI inference of the Bernoulli SBM in [39]. There, most of the time, a subset of vertices and all edges of each vertex of a fixed user defined size is sampled uniformly at random from the network. The size of these subsets, called 'batch size', is shown to strongly influence the speed of convergence and quality of the results in [39]. There is also an influence of the step size of the SVI inference on quality and computational speed in [39]. This step size has also to be determined to an extent by trial and error [41, 39].

7.2 Divisive Algorithms

We now review several exemplary divisive algorithms like bisecting K-means ([109]) or X-means [99] which use the K-means algorithm for inference. The bisecting K-means algorithm in the version of [109] (or [110]) is an example for a divisive algorithm with a model selection criterion and splits according to the model selection criterion. A bisecting algorithm which uses a model selection and simultaneous splits of the clusters together with K-means inference was proposed in [99]. Both algorithms, the bisecting K-means and the X-means algorithm are notable because they combine a divisive scheme with subsequent refinements of the cluster assignments with the batch (normal) K-means algorithms.
The classic K–means algorithm [83, 80] (see also [19]) is a batch algorithm which is run for a fixed number of clusters, K. There are several versions of the K–means algorithm, when we refer to the K-means algorithm we mean the popular approach of [80] in the version of [19] adopted to networks in e.g. [89].
It is also of interest for us, because it was applied to initialise start values for the cluster assignments of the vertices in model based clustering e.g. in [89, 73, 29].
The **basic K-means algorithm** consists of the random choice of K data points (vertices in the case of networks) called centroids. Then the following two step algorithm is applied until convergence [99, 19]:

(i) Assign each data point to the centroid with the smallest distance (e.g. Euclidean distance). Each centroid corresponds to a cluster.

(ii) Re-calculate the new centroids of each cluster depending on the data points which were assigned to it in **(i)**.

Convergence of K-means is reached if there is no change in step **(i)**. The inference of the K-means algorithm has an EM–like structure which consists of the E–step and the M–step [19]. K-means only converges to a local optimum and is recommended to be run for different start values as explained for the batch algorithm (see chapter 4) [99]. Therefore we face a comparable problem as for variational methods of possibly many 'bad' local optima.

7.2.1 Bisecting K–means

This review of the bisecting K–means algorithm is based on [109, 110].
Split and join (merge) steps of the clusters to refine an existing cluster assignment returned by the algorithm K-means were proposed in [32] at the beginning of the 1990s. These split and join steps were also used in [70] together with incremental update of the centroids. The application of the K-means algorithm restricted to randomly sampled subsets as a refinement of the existing cluster assignments was proposed in [23].
The maximum number of clusters $K_{\max}$ is chosen in advance. At the beginning of the algorithm all vertices or data points are assigned to one cluster. Then this cluster is split into two new clusters with the help of the K-means algorithm. The biggest cluster of all existing clusters is split with the help of the K-means algorithm. Then the following **bisecting K-means algorithm** is applied [110, 109]:

(i) Choose the cluster to split.
(ii) Split the chosen cluster into two new clusters with the help of the K-means algorithm.
(iii) Repeat step **(ii)** a user defined number of times and pick the best of these splits according to a cluster similarity criterion.
(iv) Steps **(i), (ii)** and **(iii)** are iterated until the chosen number of clusters, $K_{\max}$, is reached.

The bisecting K-means is a good example for a divisive algorithm. We note that step **(ii)** uses several trial splits and chooses the best one according to model selection criterion.
The strategy for choosing the cluster to split was also discussed in [110]. There it was concluded that choosing the largest cluster, the cluster with the smallest cluster similarity or a combination of both methods did not lead to any significant differences of the results in several tests. Consequently, the largest cluster was always chosen in the tests to split.
For optional refinement of the cluster assignment, $\mathbf{Q}_{\mathbf{bisect}}$, returned by the bisecting K-means algorithm it is proposed in [110] to apply the normal K-means algorithms to $\mathbf{Q}_{\mathbf{bisect}}$. It was noted in [110] that a refinement for the algorithm was also proposed in [32, 23, 70].

The bisecting K-means algorithm was shown in [110] to outperform agglomerative algorithms of e.g. [56] for documents clustering both in speed and quality of the results.

7.2.2 X-means

We review the X-means algorithm, which was introduced in [99], together with its discussion of concepts for split algorithms also presented in [99].
A maximum number of clusters $K_{\max}$ is specified for the X-means algorithm. It is assumed that an existing partition with K clusters exists which was returned by the batch K–means algorithm. Now, K–means is run until convergence in order to split each existing cluster into two new clusters. During all those runs, the applications of the K–means algorithm is restricted to the vertices (data points) of each existing cluster.
After convergence of these runs, the BIC model selection criterion is calculated restricted to the data points of each original (parent) cluster, to decide separately if the newly split cluster (child cluster) should be kept or not.
So, the **X-means algorithm** consists of the following steps [99]:

(i) Run the normal K-means with the existing cluster partition as the input. This is called the improve-params step.

(ii) Split each existing cluster into two new cluster by applying the K-means algorithm restricted to the data points in each cluster. This is called the improve-structure step.

(iii) Check if the maximum number of clusters was reached. Otherwise, continue with **(i)**.

It was remarked in [99], that X-means covers all 2^K possible configurations to split an existing cluster partition.
Moreover it was found in [99] that restricting the splits of each cluster to only two new clusters helps to avoid local optima the K–means algorithm could get trapped in, if the split proposals would involve more than two clusters.
Obviously, X-means determines the number of clusters during the inference process [99] although there is a user defined maximum number of clusters. To accelerate the inference of the improve structure operations it was also recommended to store the clusters which were not split and which did not receive any new data points (vertices) during the K–means in the improve–parameter step. These 'inactive' clusters need not be revisited in the subsequent iterations of the algorithm.
We remark, that the improve-structure operations (step **(ii)**) are not an option for a split algorithm for inference of the Stochastic Block Model, because the inference and model selection is restricted to subsets of the network which would be a violation of network interdependency of the counter example of [35, 101]. We reviewed network interdependency of the SBM in section 3.2.
Two other possible split operations were considered in [99].
The first alternative was to split one existing cluster and then to check if the split proposal improved the model selection criterion like e.g. in the bisecting K-means algorithm. For such an algorithm, which would be the bisecting K-means we reviewed in section 7.2.1, there would be $\mathcal{O}(K_{\max})$ splitting operations where $K_{\max}$ is the maximum

number of cluster during the inference process [99]. Such an algorithm would demand a strategy for choosing the next cluster to split [99] but no strategy to choose the next clusters was given in [99]. The other idea was to split half (or another percentage) of the parent clusters and then to decide with respect to the model selection criterion whether to keep all splits or to keep none of them. This idea was dismissed with the argument that some optimal splits might be missed due to a large number of none optimal splits [99]. There would be again the need for a criterion or heuristic to choose the clusters to split.

7.2.3 Discussion of bisecting K-means and X-means

We remark, that the X-means algorithm and the bisecting K-means algorithm both refine the existing cluster assignments in their refinements and do not limit the refinement to subsets of the data. The X-means algorithms applies the refinement (improveparams) of the existing cluster assignments in succession with the split step (improvestructure).
Bisecting K-means and X-means have both a user defined maximum number of clusters and do not solely rely on the model selection to determine the optimal number of clusters. In the case of bisecting K-means, the bisecting steps concentrate solely on the biggest cluster. We note, that this might be very suboptimal with respect to the model selection criterion because all other clusters are not considered for bisection in the bisecting K-means algorithm.
This is especially problematic, if the biggest cluster according to the true clustering result (ground truth) should not be split but rather one of the smaller clusters. Examples of such situations can be easily constructed for larger networks. In such an example, more than half of the data points or vertices could be assigned to one constructed cluster and the remaining vertices to several smaller clusters.
We will see in the discussion of our own Blockloading algorithm in chapter 8, that the favourable bisections steps should not only concentrate on the biggest cluster because important structure of the clustering results could be missed.
We will introduce a procedure for cluster selection which avoids such situations, when we introduce our Blockloading algorithm in chapter 8. Our procedure will ensure that every existing cluster is considered for bisection and optimisation and not only the biggest or otherwise chosen cluster.
In the section 7.4.1, we will review Split–merge algorithm for Gibbs sampling to the Infinite Relational Model (IRM) where split and merge moves are randomly mixed in succession.

7.3 Agglomerative Algorithms

A good example of an agglomerative algorithm is the **Ascending Hierarchical Clustering (AHC) algorithm** in the version of [89]. The AHC algorithm is used for initialising the cluster assignments of the vertices for variational EM inference, for example in [84], [73], [29]. We will also use the the AHC algorithm to initialise start values of the VBEM batch algorithm for the Poisson SBM and our BlockVB algorithm of section 4.2. We will provide a detailed review of the AHC algorithm together with a description how to use it for start value initialisation for variational EM inference of the SBM in section 9.1. There we will see that the AHC algorithm first assigns each

vertex to a cluster and than starts merging the nearest clusters according to the Ward distance.
We continue this section with a review of the greedyICL algorithm of [29] which combines an agglomerative approach with greedy optimisation also proposed in [95].

7.3.1 Greedy Inference and agglomerative Clustering: greedyICL

An algorithm which is not based on variational methods but lead to a significant improvement of the speed and quality of the inference of the Bernoulli SBM, compared to the variational batch algorithm, was the GreedyICL algorithm of [29]. In addition, a new model selection criterion for the Bernoulli SBM, the exact Integrated Complete Likelihood criterion (ICL_{ex}), was proposed, which is a non-asymptotic and analytical model selection criterion for the SBM [29]. The GreedyICL algorithm is based on greedy schemes of [95, 21, 29]. The greedyICL algorithm is of interest as a benchmark algorithm because it was shown to outperform some of the established variational batch algorithms for the Bernoulli SBM ([50],[35]) among other inference algorithms for the SBM.
The greedy inference is combined with an agglomerative clustering algorithm [56, 123].

The GreedyICL algorithm places each vertex in a cluster of its own or uses a cluster assignment of the vertices with far more clusters than the expected optimal number of clusters. The initial cluster assignment is either done randomly or with a second fast algorithm like K-means which is only run for some iterations [29]. Then the algorithm selects a vertex at random and merges it with each other existing cluster in search of a lower ICL_{ex}. The cluster assignment with the optimal value of the ICL_{ex} of all these merge moves is then chosen as the new cluster for this vertex. If otherwise no lower value of the model selection criterion, ICL_{ex}, could be found the vertex keeps its assignment and is stored as converged. If a cluster has no vertices left after a merge move, it is deleted. The merge moves of the greedy inference continue until no improvement with a merge move of any non-converged vertex was possible.
If there was any improvement by a greedy merge move, the set of converged vertices is emptied and another greedy iteration starts for all vertices of the network.
The greedy algorithm is converged, if there was no improvement for any vertex in the iteration of the greedy algorithm.
It is recommended to apply an agglomerative algorithm to the cluster partition, $\mathbf{Q}^{(\mathbf{greedy})}$, which was returned by the greedy algorithm [29]. Therefore merge moves are applied to the clusters of the cluster partition, $\mathbf{Q}^{(\mathbf{greedy})}$, to improve the model selection criterion ICL_{ex} once more.
It was noted that the quality of the results and the speed of convergence of the greedyICL algorithm are better if an algorithm like K-means is used for the start cluster assignments. Moreover, we saw above that divisive clustering outperformed agglomerative clustering in the case of documents clustering algorithms [109].

7.4 Split-Merge Algorithms

7.4.1 Split-Merge Gibbs sampler

We review in a concise way the general procedure of the Restricted split-merge Gibbs (RGSM) sampler which was proposed in the [57] and the Sequentially-Allocated Merge-Split (SAMS) sampler of [33]. The example mixture model used in [57] is closely related to the SBM. The RGSM Split–Merge sampler is the basis for state of the art inference of the Infinite Relational Model (IRM) [67, 68, 46, 92, 105, 12]. The inference of the IRM which was proposed in [67] is based on the restricted split–merge Gibbs sampler of the latent cluster assignments proposed [57] to infer the otherwise intractable likelihood. The intractability of the likelihood is analogous to the challenge of network interdependency [35, 101] we discussed in section 3.2.
Gibbs sampling belongs to the Markov Chain Monte Carlo (MCMC) methods [19]. Its use was also proposed for the inference of small SBMs (up to 200 vertices) in [97, 35, 73].
Gibbs sampling theoretically converges to the true statistics of the posterior [62, 97]. In practice, it can be more difficult to register the convergence of Gibbs sampling when compared the variational algorithms [62]. Convergence of Gibbs sampling is also slower than of variational algorithms [62]. The basic version of Gibbs sampling inference consists of sampling one hidden cluster assignment and holding the other fixed [97, 92]. For bigger or complex networks, the basic Gibbs Sampler often stays in bad local optima, regardless of the number of iterations of the sampling [57]. The result is a suboptimal clustering of the vertices. Thus the size and complexity of networks (or mixtures) traditional Gibbs sampling can handle with good results is limited [57].
This situation is familiar from a perspective of a VBEM or VEM user where we also have to deal with the convergence to a local optimum.
As a solution to these shortcomings of the traditional Gibbs sampler, the Restricted Gibbs split-merge (RGSM) sampler was introduced in [57]. The idea of the RGSM sampler is to combine split and merge moves in succession with regular Gibbs sampling for the whole cluster partition. The cluster wise inference proposed in the RGSM helps to avoid bad local optima and facilitates the identification of additional clusters [57].
We will now review the general principles of this split-merge sampler. The general procedure of the **split-merge Gibbs sampler** of [57, 33] in the version of [46] is:

(i) Draw two vertices (data points), i_1, i_2, uniformly at random from the set of all vertices (data points) $\{1,\ldots,N\}$.

(ii) If $i_1, i_2 \in k^{(old)}$ are assigned to the same cluster, $k^{(old)}$, form two new clusters, $k_1^{(new)}, k_2^{(new)}$ and assign i_1 to $k_1^{(new)}$ and i_2 to $k_2^{(new)}$. Then randomly (or according to a chosen probability) assign all remaining vertices of $k^{(old)}$ to the new clusters $k_1^{(new)}$ and $k_2^{(new)}$.

(iii) If otherwise $i_1 \in k_1^{(old)}$ and $i_2 \in k_2^{(old)}$ holds, e.g. the two vertices are assigned to two different clusters, merge the two clusters $k_1^{(old)}, k_2^{(old)}$ into a new cluster $k^{(new)}$.

(iv) The split or merge move in **(ii)** or **(iii)** is accepted according to the acceptance probability. This can be the Metropolis-Hastings acceptance probability [57, 33].

After each split-merge step of the RGSM sampler of [57] or the Sequentially-Allocated Merge-Split (SAMS) sampler of [33] it is recommended in [57, 33] to run the normal Gibbs sampler for the whole existing cluster partition. These runs of the traditional Gibbs sampler are comparable to the refinement proposed for e.g. the bisecting K-means or the X-means algorithm we reviewed in section 7.2. It was noted in [57] that the runs of the normal Gibbs sampler increased the quality of the results.
In [57], the vertices were randomly assigned to the new clusters $k_1^{(new)}, k_2^{(new)}$ in step **(ii)**. Gibbs sampling restricted to the vertices of the new clusters $k_1^{(new)}, k_2^{(new)}$ in step **(ii)**, was used for further improvement because of these random assignments.
As an improvement of the computational efficiency of the split merge sampler, the vertices were assigned with the allocation probability in [33] to the new clusters in step **(ii)**. Then the restricted Gibbs sampler is applied to the vertices in the two new clusters, $k_1^{(new)}, k_2^{(new)}$.
The RGSM sampler of [57] was shown in numerical tests to outperform the traditional Gibbs sampler with respect to the quality of the results and could achieve results previously unattainable by the traditional Gibbs sampler. It was also remarked in [57] that there is potential for improvement of the split-merger Gibbs sampler because of the random choice of the clusters to split or merge. It was also noted that the randomly chosen vertices i_1 and i_2 in step **(ii)** cannot change their cluster during the split step. This task was addressed for example in [113].
The split-merge Gibbs sampler for mixture distributions is a field of ongoing research and improvement [59, 26, 113]. In the next section we finally see how the idea of split-merge procedures was applied for Variational EM inference of the SBM.

7.4.2 Variational Split–Merge algorithms for the Stochastic Block Model

We have seen in the previous sections that split–merge algorithms outperformed batch algorithms for clustering and greatly improved the overall performance of different clustering algorithms. To the best of our knowledge there exist only two algorithms which employ Variational EM (VEM) or Variational Bayesian EM (VBEM) methods in combination with split–merge moves for inference of the SBM: There are the Wmixnet [75, 76] (later: Blockmodels [79]) software package which builds on the VEM algorithm and our own Blockloading algorithm [118, 116] which uses the Variational Bayesian EM based BlockVB and BlockVB algorithm (Algorithm 4).
We will also present in chapter 8 the successor algorithm of our Blockloading algorithm which we called the Blockloading++ algorithm a greatly enhanced divisive split and refine algorithm.

Review of the Wmixnet package We now review the Wmixnet software package with special regard to the split–merge options Wmixnet offers to further optimise an existing partition. Our review is based on the software documentation in [75, 76]. Unfortunately, no pseudocode or in depth documentation is given in [75, 76] about the split–merge features of that package.
The Wmixnet package uses the Variational EM algorithm of [35, 84] which was explained in section 3.3) for inference of the SBM. It was explained in section 3.3, that the VEM algorithm is a frequentist algorithm which returns point estimates for the model

parameters. The VEM algorithm depends on start values for the cluster partition The default clustering algorithm for start values is the Absolute Eigenvalues Spectral Clustering algorithm [102]. This Spectral Clustering algorithm was shown to be consistent for the undirected Bernoulli SBM in [102]. The ICL criterion for the SBM of [35, 84], which was reviewed in section 6.4, is used for model selection. Clustering algorithms for directed and undirected as well as weighted Stochastic Block Models are offered, among others for the frequentist Bernoulli and Poisson SBM.
Wmixnet first calculates cluster partitions for different numbers of clusters with a VEM batch algorithm like in [35, 84]. In addition, a split–merge option for these returned partitions is offered to further improve the value of the ICL model selection criterion. So the Wmixnet package combines the batch algorithm with an additional split–merge option. The optimal number of clusters, $K^\star$, together with the optimal value of the model selection criterion, is by default calculated with the batch algorithm for model selection which was reviewed in Algorithm 6. The split option is called the the *ascend* mode where existing clusters are split and the merge option, called *descend* mode where existing clusters are merged to further improve the ICL criterion. We deduced from the documentation of Wmixnet [75] that the ascend mode and the descend mode are performed separately.
We conclude that the Wmixnet package is by default no full–fledged divisive algorithm but can considered as a batch algorithm with split–merge option.

The Blockloading algorithm Our Blockloading algorithm[118, 116] is a fully divisive algorithm where all vertices are assigned to the same cluster and the inference, based on our optimised VBEM subset algorithm, BlockVB, we presented in Algorithm 4 in chapter 4, consists of alternating between an Expansion Step where a cluster is split to optimise the free energy model selection criterion (section 4.1) and the Refinement Step where a cluster of the existing partition is optimised with respect to the existing partition. So the expansion and the refinement of the partition are applied in succession for the same cluster contrary to Wmixnet. In the Blockloading algorithm, we proposed several criteria for choosing the cluster to expand or refine.
The Blockloading algorithm was first developed for the Bayesian Bernoulli SBM [118] and later adopted and expanded for the Bayesian Poisson SBM [116]. Contrary to Wmixnet, the Blockloading algorithm does not apply the batch algorithm and the model selection for the batch algorithm, given in Algorithm 6, at all. All vertices are assigned to the same cluster at the beginning of each inference step. Thus, the existing partition also provides most of the start values for further the next Expansion and Refinement Steps. So, start values are only needed for a subset of vertices of the network. We will present the Blockloading in detail in chapter 8 and postpone the further discussion of the motivation and the advantages of our Blockloading algorithm to this chapter. We will also present a greatly improved Blockloading++ algorithmic framework in chapter 8.

Review of the Blockmodels package The successor of the Wmixnet package is the Blockmodels package [79] which uses optimised C++ code and the Armadillo package for matrix computations and other time critical parts of the inference. It can be downloaded from https://cran.r-project.org/web/packages/blockmodels/index.html.
Like Wmixnet, it is based on the VEM inference algorithm and the ICL model selection for different Stochastic Block Models. According to the documentation of Blockmodels [79, 77], Blockmodels uses user defined input of the number of clusters. In addition,

the user can also define the number of split and merge steps in the ascend and descend mode.
The default number split steps in the ascend mode is two times the number of clusters, K, at the beginning of the inference. The Blockmodels package still offers the Absolute Eigenvalues Spectral Clustering algorithm for the initialisation of the start values. According to [79], Blockmodels also offers the option to re-use existing cluster partitions of lower numbers of clusters as start values for the inference of cluster partitions with a higher number of clusters. This is a feature which we already proposed in [118, 116] in the context of variational inference of the SBM, and which we identified as one of the strengths of our Blockloading algorithm.
The ascend and descend modes are applied separately en bloc in the Blockmodels package [79]. We verified this from reading the state messages during the inference process when we ran the program. So, first the cluster partition is expanded for the number of user specified iterations dependent on the current number of clusters in the ascend mode and afterwards dependent on the number of clusters which then exists, the merge steps are performed. It is remarked in [79], that 'oversplitting' might occur during the ascend mode which means that some splits of clusters are non-optimal and some clusters should be merged again afterwards in the descend mode. So, the inference of the Blockmodels package accepts sub–optimal (unfavourable (see section 8.2.2) cluster partitions during the inference process. Thus, unfavourable local optima are not avoided.
We will compare our Blockloading(++) algorithm with the Blockmodels package with numerical tests and especially with regard to the clustering of earthquake networks in chapter 10.

7.4.3 Discussion of Merge and Refinement Steps

If the expansion and merge steps are performed separately there exists the danger of *oversplitting* [79] the clusters and consequently overfitting the model. This separate application of split and merge steps means that in a first step there are only split proposals for the existing clusters until the maximum of iterations or convergence of the model selection criterion. Then in a following second step there are only merge steps performed until the maximum number of iterations or there are no clusters left to be merged. It makes sense to employ merge moves to correct for over splitting with the result of a too high non optimal number of clusters.
If we therefore use such an ascend–descend approach we would accept over splitting which means that the cluster partition is expanded to an unfavourable (suboptimal) local optimum. We refer to section 8.2.2 for our definition of favourable and unfavourable local optima. We have already mentioned that it is not easy and sometimes virtually impossible for the inference algorithm (also VEM and VBEM) to escape out of unfavourable local optima.
So, if we accept over splitting, we accept automatically that the inference process has to escape from a suboptimal cluster partition to approach the global optimum again. Contrary to such a separated split-merge approach, we take the different approach of preventing the inference process of moving to such a unfavourable state in the first place.
Our concept includes Expansion and Refinement steps which are applied each to the active cluster in succession in a nested way to keep the inference near the global optimum for a given number of clusters. In fact we want the inference process to be at or near the global optimum for a given number of clusters.

Chapter 8

The Blockloading Algorithms

In this chapter, we propose our Blockloading algorithm which is a divisive clustering algorithm which expands and refines the clusters (blocks) of the Stochastic Block Model (SBM). With our Blockloading algorithm we expand and refine the cluster partition in an integrated, nested way by alternating between expansion and refinement of the cluster partition.
We have reviewed the general principles of divisive and agglomerative algorithms in chapter 7. There we reviewed several algorithms which build upon these ideas, like split, merge and split-merge algorithms.
We proposed the Bayesian BlockVB algorithm (see Algorithm 4) and the BlockVB++ algorithm (see Algorithm 5) which uses our newly introduced informative adaptive priors for an optimised VBEM based optimisation of the free energy (ILvb) model selection criterion of section 6.2 for subsets of the network. The preparation of our BlockVB++ algorithm allows us the fully Bayesian inference of subsets of the vertices of the network we need for our Blockloading algorithm.
We will introduce and discuss several variants and enhancements of the Blockloading, algorithm, we proposed in [118, 116], in the sections below. These efforts will lead us to our greatly improved Blockloading++ algorithmic framework.
Contrary to the existing variational EM inference of the Wmixnet and the Blockmodels package [75, 79] for the SBM, our Blockloading algorithm is a nested algorithm where the refinement and and expansion of the partition are performed in an alternating way. This algorithmic approach greatly reduces the risk of 'over splitting' which is a problem which can occur if split moves and merge or refinement moves are performed separately like in the Blockmodels package [79].
We have seen in the discussion of the Blockmodels package [79] in section 7.4.2 that only a limited number of restarts, based on the current number of clusters is performed. We will show in section 8.2.2 that such a procedure carries the risk of missing clusters and thus important structures of the network. In contrast to limited restarts, our Blockloading algorithm and its derivatives ensure that each identifiable local optimum and each cluster is found before convergence is registered. We achieve this by introducing several different novel procedures to store if a cluster could be split or refined to achieve a better value of a chosen model selection criterion. If a cluster consists of several clusters according to the ground truth or global optimum, the inference process of our Blockloading algorithm concentrates on finding all local optima in the first visit of each existing cluster of the network.
Moreover, a limited number of visits for each cluster cannot assure convergence with

respect to network interdependency of the SBM we discussed in section 3.2 and [35, 40]. By network interdependency the graph of hidden cluster assignments forms a clique (fully connected graph) [40]. Therefore the change of the cluster assignments of any vertices in a cluster can affect the optimal cluster assignments of different vertices in a different cluster [35, 40]. To take this effect into account we have to visit each cluster until there was no change in any cluster to rule out that the cluster partition is suboptimal. We will show that our new algorithmic approach aims to prevent to get stuck in unfavourable local optima which are far away from the global optimum of the model selection criterion.
We start the optimisation with all vertices of the graph in one cluster and calculate the free energy F (ILvb). Therefore our Blockloading algorithm is a divisive algorithm. This cluster is divided into two clusters in search for a lower converged free energy F . We continue to alternately divide and optimise (refine) chosen clusters by using a variant of the VBEM algorithm. This variant for the optimization of the vertices belonging to one cluster is called the BlockVB algorithm.
The BlockVB algorithm for the Bernoulli SBM is given in appendix A and for the Poisson SBM it was introduced in section 4.2 and Algorithm 4.
The merits of starting with all vertices in one cluster and to proceed by expanding and refining clusters are discussed below in section 8.1 and also in section 8.2. One obvious advantage of this bottom up approach is, that the number of clusters according to the model selection criterion is determined automatically.
An at least equally important advantage is, that the restriction of the optimisation to subsets leads to a high improvement of the results because of the improved quality of the start values.

8.1 The Blockloading algorithm

Now, we propose our Blockloading algorithm. The following sections contain revised parts of our technical report [116]. We changed some parts of the following presentation of our Blockloading algorithm in comparison to [116], to allow the reuse of the definitions and concepts for the presentation of our newly added Blockloading++ algorithm. The way our Blockloading algorithm works is unaffected by these changes.
We will discuss the concepts and thoughts which lead to our Blockloading algorithm in section 8.2. We will see there that we can build a divisive inference framework upon our Blockloading algorithm. This discussion will provide us with new ideas we will use to introduce our greatly improved Blockloading++ algorithm.
We start the Blockloading algorithm with the initialisation step. The input for the Blockloading algorithm is the adjacency matrix of the network, $\boldsymbol{A}$, the choice of hyperparameters (see chapter 5) or the initialisation algorithm for the start values, and the number of iterations for the VBEM algorithm (see chapter 4).

Initialisation of Blockloading We calculate the converged free energy, F, for $K = 1$ clusters, i.e. all vertices are assigned to the same cluster, $c^{(ini)}$. So, the start partition matrix, $\boldsymbol{Q}^{(start)}$, is an $N \times 1$–matrix with all entries equal to one. We only need a single M-step of the VBEM batch algorithm (see Algorithm 3) to calculate the model parameters of the SBM for the free energy, F. We use equations (A.4), (A.5), (A.6) together

with eqn. (A.16) for the calculation of F of the Bernoulli SBM and Propositions 2, 3 and 1 for the Poisson SBM.
The result, the converged free energy F for $K = 1$, is stored as the *reference free energy*, $F^{(ref)}$.
Now we check if a lower (better) converged free energy then $F^{(ref)}$ can be achieved for a partition matrix with $K = 2$ clusters. We initialise a $N \times 2$–partition matrix, $\boldsymbol{Q}^{(start)}$, for the VBEM batch algorithm (algorithm 3 in section 4).
There are several algorithms for initialising $\boldsymbol{Q}^{(start)}$. We explain the most widely used of these algorithms like the K-means algorithm in chapter 7 or the Ascending Hierarchical Clustering (AHC) algorithm in chapter 9. Most of these algorithms assign each vertex to exactly one cluster which is called *hard clustering* or *hard assignment* in the literature [84]. We give the definition of hard clustering below.
If we use our fully Bayesian BlockVB++ algorithm (see algorithm 5) instead, we do not need such a second algorithm. We randomly initialise the cluster assignment of the vertices in the $N \times 2$–matrix $\boldsymbol{Q}^{(start)}$, which has normalised matrix rows $\sum_{k=1}^{K} Q_{ik}^{(start)} = 1$.
For our fully Bayesian BlockVB++ algorithm, the hyperparameters or our new adaptive informative hyperparameters play the role of start values (see chapter 5).
Either way, we run our VBEM algorithm with the start values $\boldsymbol{Q}^{(start)}$ for $K = 2$ clusters.
The VBEM algorithm returns the converged free energy $F^{(trial)}$, the cluster partition matrix $\boldsymbol{Q}^{(trial)}$ and the model parameters $\boldsymbol{\vartheta}^{(trial)}$. If $F^{(trial)} < F^{(ref)}$ holds, $\boldsymbol{Q}^{(trial)}$ is stored as the new reference partition matrix $\boldsymbol{Q}^{(ref)}$. Similarly, $F^{(ref)}$ is overwritten by $F^{(trial)}$, the model parameters are updated with $\boldsymbol{\vartheta}^{(trial)}$ and the number of clusters is set to $K^{(ref)} = 2$.
Otherwise, if the split was not accepted, we can try different start values. If this does not improve the reference free energy $F^{(ref)}$, we conclude that the optimal cluster assignment is given by $c^{(ini)}$.

We use the results of the Initialisation step, which returned a cluster assignments of $K = 2$ clusters, as start values for the following refinement and subdivision of the existing clusters. Now, we give a short overview of the main steps of our Blockloading algorithm and explain the steps in detail below. We recall from the introduction of the chapter that our Blockloading algorithm is an integrated algorithm where optimisation and expansion of the cluster partition are performed in succession.
Optimisation in our Blockloading algorithm is restricted to the vertices of one cluster and the assignments of all other vertices are kept fixed. We call the cluster chosen for optimisation the *active cluster*. The vertices which are not in the active cluster play the role of start values.
In the following *main loop* of the algorithm we will pick one of the two clusters and try to optimise the reference partition, $\boldsymbol{Q}^{(ref)}$, with our BLockVB or BlockVB++ algorithm restricted to the vertices in the active cluster where the cluster assignments of the other vertices are kept fixed. We called this procedure the *Refinement step*.
Afterwards, we determine a cluster and try to split it in into two new clusters to improve the reference free energy $F^{(ref)}$ in our *Expansion step*. We remark that the assignment of the vertices which are not in the active cluster stay fixed but play the role of start values for the optimisation. We use either our BlockVB or BlockVB++ algorithm for this optimisation.
At the end of the main loop we check for Convergence of the Blockloading algorithm.

Overview of the algorithm We sum up the main steps of the **Blockloading algorithm**:

Input.–Adjacency matrix $\boldsymbol{A}$, model type
Result–Cluster partition matrix $\boldsymbol{Q}^{(ref)}$, number of clusters $K^{(ref)}$ and parameters $\vartheta^{(ref)}$.
(i) Blockloading Initialisation.

Main Loop.
(ii) Refinement Step.
(iii) Expansion Step.
(iv) Check for Convergence of all clusters.

The Initialisation Step and the overview show, that the Blockloading algorithm is a *divisive* algorithm.
Now, we formalise and describe all steps in the order of occurrence. We start this formalisation by noting the terminology concerning the active cluster in the following definition.

Definition 2 (Active Cluster and Active Vertices)**.** *The cluster which is chosen for optimisation with the inference algorithm is called the active cluster, $c^{(active)}$. The vertices which are assigned to $c^{(active)}$ are called the active vertices. The set of indices of the active vertices is given by $I^{(active)}$.*

To pick a cluster to optimise, we first need to know which vertex belongs to which cluster. The partition matrix $\boldsymbol{Q}^{(ref)}$ which is returned by the VBEM inference algorithm is a *fuzzy partition* which gives the probabilities that a vertex belongs to one of the clusters. We transform the *fuzzy clustering* into a *hard clustering*.

Hard Clustering As explained in Section 4, the partition matrix $Q^{(ref)}$ returns a probability for the cluster assignment of each vertex (fuzzy clustering). In the following paragraphs we use the concept of cluster dependent optimisation of the existing partition matrix. To be able to optimise the vertices in one cluster, we first have to determine which vertices belong to which of the clusters. We transform all vertices i of $\boldsymbol{Q}^{(ref)}$ into a hard clustering by transforming each matrix row $\boldsymbol{Q}_i = (Q_{i1}^{(ref)}, \ldots, Q_{iK}^{(ref)})\ \forall i \in \{1, \ldots N\}$ according to

$$Q_{ij}^{(hard)} = \begin{cases} 1, & \text{if } Q_{ij} = \max\limits_{j \in \{1,\ldots,K\}} Q_{ij} \\ 0, & \text{else} \end{cases}. \tag{8.1}$$

We note that the transformation to hard clustering can be performed incrementally because only the cluster assignments of subsets change during our Blockloading algorithm. Now, we know which vertex i belongs to the active cluster and can identify the vertices in the active cluster, $c^{(active)}$.

We still need a cluster selection method to choose the active cluster. Due to our use of the model based clustering approach with the SBM, we can not only use the information which is contained in the the cluster partition $\boldsymbol{Q}^{(ref)}$ but also the model parameters

$\boldsymbol{\vartheta}^{(ref)}$. We choose the active cluster with the *cluster selection method*. There are several ways to choose the active cluster we will discuss in section 8.3 below. We choose the active cluster each time we want to optimise a subset of the vertices based on the cluster membership. We call this the *Cluster Selection step*. We note this terminology in the following Definition 3.

Definition 3 (Cluster Selection Step)**.** *In the Cluster Selection step we choose the active cluster and thus the active vertices with the help of the cluster selection method.*

Choice of the active cluster As the default cluster selection method, we choose the cluster with the highest number of vertices as the active cluster. We calculate the number of vertices in one cluster, n_k according to:

$$n_k = \sum_{k=1}^{K} Q_{ik}^{(ref)}. \tag{8.2}$$

The active cluster $c^{(active)}$ is therefore $a = \max_{k=1,\ldots,K} n_k$.
Such a cluster selection method was also used in e.g. [109] for the bisecting k-means algorithm. We will discuss in section 8.3, that this choice of the cluster selection method minimises the computational burden because we need fewer updates of the clusters before convergence is reached. We use the method of selecting the biggest cluster only under the assumption, that we have no (or no cheap to calculate) method to infer which clusters are worth optimising.

If we would have a computationally cheap criterion to find out which clusters could be split or refined to improve the reference free energy, $F^{(ref)}$, we could also choose the clusters according to this criterion. We propose the usage of Largest Gaps algorithm [27] or our newly introduced [116] Optimal Gap algorithm in the discussion of the choice of the active cluster in section 8.3.
Thus, we will have indeed a good idea of clusters which hold potential for optimisation. Both algorithms can be calculated in linear time before we use our more precise BlockVB++ algorithm.
For some graphs with a high variance of vertex degrees, we can choose to focus the inference on the most densely connected cluster first. The densely connected clusters can be identified with the help of the model parameters, $\boldsymbol{\vartheta}^{(ref)}$. We propose this approach to exclude sparsely and irregular connected clusters from the optimisation process which otherwise can harm an optimal inference process. We will provide a more the detailed explanation of different cluster selection methods in section 8.3 below.
After this formal treatment of the choice active cluster we now proceed with the description of the main loop.

Main Loop of the Blockloading algorithm The **Main Loop** of Blockloading begins with the Refinement Step of the active cluster, $c^{(active)}$.

Refinement Step In the Refinement step we want to improve the reference free energy, $F^{(ref)}$, by optimising the vertices in the active cluster $c^{(active)}$. We can apply an infinite number of Refinement steps to an existing reference cluster partition, $\boldsymbol{Q}^{(ref)}$. To use our BlockVB (see Algorithm 4) or our fully Bayesian BlockVB++ algorithm

(Algorithm 5), we randomly initialise the cluster assignments of the vertices in $c^{(active)}$. We initialise the cluster assignment of each vertex $i \in c^{(active)}$ in $\boldsymbol{Q}^{(ref)}$ with a normalised $1 \times K^{(ref)}$–random vector. These random start values are inserted into $\boldsymbol{Q}^{(ref)}$ for all vertices in $c^{(active)}$. All vertices $i \notin c^{(active)}$ remain the same as in $\boldsymbol{Q}^{(ref)}$. This yields the start partition matrix, $\boldsymbol{Q}^{(start)}$, for the Refinement step.
We restrict the use of informative hyperparameters for our BlockVB++ algorithm (see Algorithm 5) to the inter cluster connections of the $K^{(ref)}$ clusters and use non–informative or neutral hyperparameters for all other clusters.
We apply our BlockVB algorithm with non–informative or neutral priors (see section 4) for all clusters.
We remark that it is also possible to apply a start value algorithm of chapter 7 or 9 for the start cluster assignment of the vertices in the active cluster. This did not yield any improvement compared to random start values in our tests.
We run our BlockVB algorithm or our BlockVB++ algorithm for all vertices $i \in c^{(active)}$ with the cluster partition matrix $\boldsymbol{Q}^{(start)}$ and $K^{(ref)}$ clusters as the input. BlockVB(++) returns the fuzzy cluster assignment of the vertices $\boldsymbol{Q}^{(trial)}$, the free energy, $F^{(trial)}$ and the model parameters, $\boldsymbol{\vartheta}^{(trial)}$. After the convergence of our BlockVB(++) algorithm we proceed with the evaluation of the results of the Refinement Step.

Before we describe the evaluation of the Refinement step, we first introduce our solution for storing clusters which we do not consider as choice for the active cluster for the Refinement or Expansion step in the inference process any longer.

Number of converged clusters If the application of the Expansion Step to the active cluster, $c^{(active)}$, did not improve the reference free energy, $F^{(ref)}$, we increase the **number of converged clusters**, $\mathscr{C}$, by one. Otherwise, if there is an improvement in either the Refinement step or the Expansion step we set $\mathscr{C} = 0$ after this step. We call this procedure the **reset** of the number of converged clusters. The number of converged clusters guides the choice of the active cluster and plays the role of the 'memory' of the Blockloading algorithm, as we will show in the next paragraph.

Cluster Selection Step In the Cluster Selection Step we choose the active cluster, $c^{(active)}$, according to the cluster selection method. We apply the cluster selection method to generate a list of the clusters of the hard clustered reference cluster partition, $\boldsymbol{Q}^{(ref,hard)}$. In this **cluster selection list**, the clusters are ordered according to the cluster selection method. For example, if we decided to use the cluster sizes for the cluster selection method, we list the clusters in descending order on the list dependent on the cluster sizes.
Now, we use the information of the number of converged clusters, $\mathscr{C}$ and choose the $(1+\mathscr{C})$-th cluster on the cluster selection list as the new active cluster.

Evaluation of the Refinement Step We evaluate the outcome of the Refinement Step with respect to the converged free energy. Before this evaluation, we remove clusters which were emptied during the Refinement step according to VBEM. We explained the occurrence empty clusters in the context of VBEM inference in chapter 6.
If $F^{(trial)} < F^{(ref)}$ holds, we update the reference free energy, $F^{(ref)}$, the cluster partition matrix, $\boldsymbol{Q}^{(ref)}$, and the model parameters, $\boldsymbol{\vartheta}^{(ref)}$.
We set the number of converged clusters, $\mathscr{C}$, to zero. We perform the Cluster Selection

step (see above) to choose the active cluster, $c^{(active)}$.
If the result is otherwise $F^{(trial)} \geq F^{(ref)}$, we keep the old reference values.

We remark that the Refinement Step can potentially change already converged clusters because additional vertices might be assigned to them in the Refinement Step. This is the reason why we choose to reset the $\mathscr{C}$ after each successful Refinement Step.
We also reset $\mathscr{C}$ after each successful Expansion step. This reset of $\mathscr{C}$ also leads to a frequent revisit of every cluster in complex networks like earthquake networks. This feature of our Blockloading algorithm increases the chance that all favourable local optima are identified.
Moreover, the change in the cluster assignment of the vertices $i \in I^{(active)}$ can affect the cluster assignment of the vertices $i \notin I^{(active)}$ too, because of network interdependency (see [35, 40] and section 3.2). Therefore, all clusters have to be updated until convergence is reached.

The Refinement Step is an *error correction* feature of the Blockloading algorithm which is already used during the calculation process. With the Refinement Step we want to prevent the algorithm from wrongly splitting too many clusters (also called oversplitting [79]). We proceed to the Expansion Step where we try to split the active cluster into two new clusters to improve upon the reference free energy $F^{(ref)}$.

Expansion Step We want to improve the reference free energy, $F^{(ref)}$, by finding a cluster assignment of the active vertices, $i \in c^{(active)}$, to two new clusters.
To prepare the split of the active cluster, we have to initialise start values for the active vertices, $i \in I^{(active)}$. We assign these start values by initialising a cluster partition of $K = 2$ clusters for the active vertices, $i \in I^{(active)}$. If we use our BlockVB algorithm with non informative hyperparameters (see section 4.2) we can initialise this partition with a start value algorithm from chapter 7.
If we use our fully Bayesian BlockVB++ algorithm with informative priors or our adaptive informative priors, we randomly initialise normalise 1×2–vectors for all vertices $i \in I^{(active)}$.
We store the active vertices of $\boldsymbol{Q}^{(ref)}$ for possible later use and use $\boldsymbol{Q}^{(ref)}$ as $\boldsymbol{Q}^{(start)}$. Then we set all matrix rows $i \in I^{(active)}$ from $\boldsymbol{Q}^{(start)}$ to zero and add an additional column of zeros to $\boldsymbol{Q}^{(start)}$ making it an $N \times (K^{(ref)} + 1)$–matrix. Thus we get two empty clusters $k_1^{(active)}$ and $k_2^{(start)}$ in $\boldsymbol{Q}^{(start)}$. The start values, which were initialised above, are inserted into the rows $i \in I^{(active)}$ of $\boldsymbol{Q}^{(start)}$. We use our adaptive informative or fixed informative hyperparameters for the inter cluster connections of the active clusters $k_1^{(active)}$ and $k_2^{(start)}$.
We run our BlockVB or BlockVB++ algorithm with $\boldsymbol{Q}^{(start)}$, $K^{(ref)} + 1$ and the active vertices, $I^{(active)}$, as the input. This run of the BlockVB(++) algorithm returns the trial free energy, $F^{(trial)}$, the trial cluster partition $\boldsymbol{Q}^{(trial)}$ and the model parameters $\boldsymbol{\vartheta}^{(trial)}$.

Evaluation of the Expansion Step We remove empty clusters like we did in the Refinement step. Now, we evaluate if the reference free energy was improved in the Expansion Step. If $F^{(trial)} < F^{(ref)}$ holds, the reference values $F^{(ref)}$, $\boldsymbol{Q}^{(ref)}$ and $\boldsymbol{\vartheta}^{(ref)}$

are updated with the results of the Expansion Step. We reset the number of converged clusters, $\mathscr{C}$, to zero.
We proceed with the Cluster Selection Step and apply the cluster selection method dependent on the new reference values to choose $c^{(active)}$.
If otherwise $F^{(trial)} \geq F^{(ref)}$ holds, we increment the number of converged clusters, $\mathscr{C}$, by one. The increased number of converged clusters, $\mathscr{C}$, may be high enough to fulfil our convergence criterion of our whole Blockloading algorithm so we check for the convergence.

Convergence Step If $(\mathscr{C}+1) > K^{(ref)}$ holds, all clusters have converged and the current reference values are returned as the result of our Blockloading algorithm.
If our Blockloading algorithm has not converged, we perform the Cluster Selection step and proceed with another iteration of the main loop.
We sum up the our Blockloading algorithm in Algorithm 7.

The Blockloading algorithm may also be used to calculate a partition with no more than a predetermined number of clusters by skipping the Expansion step every time the desired number of clusters is reached.
Using the Expansion Step cluster by cluster, the occurrence of empty columns in the matrix $\boldsymbol{Q}^{(ref)}$ is minimised. This saves computational time which would be otherwise wasted for non optimal columns.
The Blockloading algorithm allows the restart with a given partition matrix $\boldsymbol{Q}$ for further improvement.
We remark that other clustering algorithms in combination with a model selection criterion can be used within the algorithmic framework of our Blockloading algorithm instead of our BlockVB(++) algorithm and the free energy model selection criterion. We refer to chapter 6 for a review and presentation of other suitable model selection criteria for the Poisson SBM, like the ICL or ICL_{ex} criterion.
We note that we can use several different (random) initialisations of the start values for the same Initialisation, Refinement or Expansion step. Then we choose the best converged free energy of all those trials, $F^{(trial,best)}$, and compare it with the reference free energy, $F^{(ref)}$, in the evaluation of each respective step.

Complexity of the Blockloading algorithm It is difficult to state the order of complexity of the Blockloading algorithm because the number of clusters changes during the inference process. We recall from our complexity evaluation of the BlockVB (section 4.2) and the BlockVB++ algorithm (section 5.1) that both algorithms have computational costs of the order $\mathscr{O}(K^2 E_I)$ where K is the number of clusters and E_I the number of edges of all vertices with index $i \in I$.
We found in chapter 4 that the computational costs of one iteration of the batch algorithm was of the order $\mathscr{O}(K^2 E)$, where E denotes the number of edges in the network. The computational costs for model selection with the VBEM batch algorithm were of the order $\mathscr{O}(K_{\max}^3 E)$ with $K_{\max}$ being highest number of clusters.
Thus, the computational complexity of the the Blockloading Initialisation step is of the order $\mathscr{O}(E)$.
The biggest clusters with the most edges during the run time of the algorithm normally occur at the beginning of the inference process, when the number of clusters is small. We also expect from our experience from computational tests, that the second biggest

Algorithm 7: Blockloading Algorithm

Data: adjacency matrix $\boldsymbol{A}$; number of iterations of the Expansion Step, $n_{\mathbb{E}}$; number of vertices N
Result: partition matrix $\boldsymbol{Q}$; number of clusters K; free energy F; model parameters $\boldsymbol{\vartheta}$

```
/* Initialisation                                                    */
```
Initialisation Step
Cluster Selection Step
repeat
  Refinement Step
  if $F^{(ref)} > F^{(trial)}$ **then** `/* check if Refinement Step improved` F `*/`
    $converged = 0$
    update cluster partition and parameters
    Cluster Selection Step
  end
  Expansion Step
  if $F^{(ref)} > F^{(trial)}$ **then** `/* check if the Expansion Step improved` F `*/`
    $converged = 0$
    update cluster partition and parameters
    Cluster Selection Step
  else
    $converged \longleftarrow converged + 1$
    `/* Convergence Step                                             */`
    if $converged + 1 > K^{(ref)}$ **then**
      break
    end
    Cluster Selection Step
  end
until *convergence of all clusters*

cluster with respect to the number of edges occurs after the initialisation step and so on. This fact positively influences the order of computational complexity of the Blockloading algorithm when compared to the batch algorithm.
Moreover, if the number of clusters grows we expect the cluster sizes to decrease. We denote with $E_{I,\max,conv}$ the number of edges of the biggest cluster of $\boldsymbol{Q}^{(ref)}$ after convergence. If the number of clusters is low, the order of computational costs for one iteration of the Blockloading algorithm is approximately $\mathcal{O}(K_{\max}^2 E_{I,\max,conv})$ In addition we assume that $K^{(ref)} \approx K_{\max}$ which from our experience with numerical tests is justified. In addition, we observed that the number of clusters was near $K^{(ref)}$ most of the time during the inference process. Then the computational costs of one iteration of the Blockloading algorithm are approximately of the order $\mathcal{O}(K_{(ref)}^2 E_{I,\max,conv})$.

We will see in the numerical tests in chapter 10 that the total number of Refinement and Expansion steps which are needed until convergence are the deciding factor for the run time of our Blockloading algorithm. These number of Refinement and Expansion steps is also heavily influenced by the cluster selection method.
To achieve a significantly lower computational cost was one of our main motivation for the introduction of our more advanced versions of our Blockloading algorithms in sections 8.5, 8.6 and 8.7. The number of the Refinement and Expansion steps is also one of the most important topics of the following discussion of our Blockloading algorithm in section 8.2 which will lead to the development of our Blockloading++ algorithm in section 8.7.

8.2 Discussion of the Blockloading algorithm

8.2.1 Convergence to local optima

We recall that the Variational (Bayesian) EM algorithm for the SBM converges to a local optimum [50, 84, 73, 76]. This well known fact leads to the recommendation to use several restarts of the inference with different start values [50, 73]. This is also true for our BlockVB and BlockVB++ algorithms for the Poisson SBM.
If the VBEM inference algorithm converges to a certain cluster assignment matrix together with the model parameters for certain start values, we consider this as a local optimum. Therefore, we can only say in retrospect that a certain cluster assignment corresponds to a local optimum according to our VBEM inference algorithm.This local optimum can but does not have to be also a global optimum.
We observed in many numerical tests, that the VBEM inference did find the (known) global optimum of the SBM for certain networks for all or nearly or even all different start values we used for initialising the algorithm. On the other hand, there were other networks where especially the VBEM batch algorithm was highly dependent on the choice of the start values with widely varying results. So the quality and reliability of the SBM inference also depends on the network.

We will propose our classification of different local optima with respect to our Blockloading algorithm below in section 8.2.2.

8.2.2 Favourable and unfavourable local optima of exclusive cluster assignments

The experience from of our numerical tests shows that there is wide space of possible of cluster assignments our VBEM inference for the Poisson SBM can converge to. This also applies to the BlockVB and BlockVB++ algorithms.
Sometimes there is a known ground truth of the cluster assignments which corresponds to the global optimum of the chosen model selection criterion.
For all networks which we generated with the SBM, the global optimum with respect to the chosen model selection criterion corresponding to the ground truth was only reached if the correct number of clusters was identified by the inference algorithm.
According to these two observations we will now assume that the global optimum of the chosen model selection criterion coincides with the existing ground truth of the cluster assignments. We also assume that the global optimum of an estimated cluster assignments can only be reached if the correct number of clusters with respect to that global optimum was identified by the inference algorithm.
Our Blockloading algorithms returned very similar results in numerical tests with earthquake networks, where we do not know if there is a ground truth, and the global optimum of the model selection criterion is unknown. So, we considered the best result as a proxy for the unknown global optimum.
Therefore, our assumptions above about the ground truth and the global optimum as well as the number of clusters make sense for earthquake networks, too.
Our Blockloading algorithm is a divisive clustering algorithm where we start the inference process with all vertices in the same cluster. Then, we apply our nested split and refine procedure, we proposed in section 8.1.
We assume that the optimal number of clusters is bigger than one. Then by construction, our Blockloading algorithm will identify several local optima before it can converge to the global optimum. During tests we found that there are different kinds of local optima which can occur during the inference process of the Blockloading algorithm.
We concluded that there are local optima which are 'favourable' in the sense that a convergence of the Blockloading algorithm to the global optimum followed from the visits of these favourable optima during the inference process. On the contrary, we also observed 'unfavourable' local optima where the Blockloading algorithm in rare cases could get stuck. Such unfavourable local optima were also the motivation for the introduction of our Refinement step for escaping out of this unfavourable local optima. So, these unfavourable local optima are harmful for finding the global optimum.
The term 'bad local optimum' was used in [75, 76] in a different way to describe a local optimum of VEM inference of the SBM with the Wmixnet package where the values of the ICL model selection criterion were not convex with respect to the number of clusters. The term 'bad local optimum' was also used in [75, 76] to describe the situation that the absolute value of the ICL criterion did not increase with the number of clusters.
Another role plays the classification of split and merge moves. In [113] examples were shown for smart splits (or merges) which lead to 'plausible' clusters (correct splits according to the ground truth) and dumb splits (or merges) which were performed at random and not necessarily lead to the splits or merges of the correct clusters. These splits and merges were used in the Smart-Dumb/Dumb-Smart Gibbs sampler of [113].
We take a different viewpoint and asked first which kind of local optima favour the convergence to the global optimum with respect to our divisive Blockloading algo-

rithm. Now, we want to clarify which local optima favour a subsequent convergence of our Blockloading algorithm (or a split-merge or split-refine algorithm) to the global optimum and which do not. We will give a formal definition of a favourable and unfavourable local optimum below after some examples. Our main idea is, that a global optimum can be reached by performing correct splits, with respect to the ground truth, to a cluster partition which is a favourable local optimum. Our definition is a general one and applies to all clustering algorithm with exclusive cluster assignments of the vertices.
We take the perspective of a pure split merge inference algorithm for our definition of favourable and unfavourable local optima.

Example cluster assignment with global optimum of the known ground truth We present our description of favourable and unfavourable local optima for clustering with the Stochastic Block Model with the help of an example. This example can be generalised to other clustering algorithms with exclusive assignments of the vertices to clusters.

Let a network with $N = 10$ vertices $V = \{v_1, v_2, \ldots, v_{10}\}$ be given. We assume that the right cluster assignment $\boldsymbol{Z}$ which corresponds to the global optimum according to the ground truth or model selection criterion is given by

$$k_1 = \{v_1, v_2, v_3\}, \tag{8.3}$$
$$k_2 = \{v_4, v_5, v_6\}, \tag{8.4}$$
$$k_3 = \{v_7, v_8\}, \tag{8.5}$$
$$k_4 = \{v_9, v_{10}\}. \tag{8.6}$$

Examples of favourable and unfavourable local optima We see that the following cluster assignment, $\boldsymbol{Q}_1$, of the vertices, V, is of course no global optimum. We denote a cluster which was estimated by the inference algorithm by $\hat{k}_i$:

$$\hat{k}_1 = k_1, \tag{8.7}$$
$$\hat{k}_2 = k_2, \tag{8.8}$$
$$\hat{k}_3 = k_3 \cup k_4. \tag{8.9}$$

Nevertheless, the assignment $\boldsymbol{Q}_1$ comes close to the global optimum in the sense that a simple split of cluster $\hat{k}_3$ to the clusters k_3 and k_4 would be sufficient to reach the global optimum.
Another example where the global optimum could be found by multiple splits is given by:

$$\hat{k}_1 = k_1 \cup k_2, \tag{8.10}$$
$$\hat{k}_2 = k_3 \cup k_4. \tag{8.11}$$

On the other hand, this is not the case for the the cluster assignment $\boldsymbol{Q}_2$

$$\hat{k}_1 = k_1, \tag{8.12}$$
$$\hat{k}_2 = k_2 \cup \{v_9\}, \tag{8.13}$$
$$\hat{k}_3 = k_3 \cup \{v_{10}\}. \tag{8.14}$$

To transform the local optimum $\boldsymbol{Q}_2$ into the global optimum we would need for example two splits of clusters $\hat{k}_2$ and $\hat{k}_3$ and one merge or refinement step to reach the cluster assignment corresponding to the global optimum. The cluster k_4 of the ground truth was wrongly split with respect to the ground truth, which is also called over splitting [79]. Moreover, the parts of k_4 were wrongly merged with the clusters k_2 and k_3. The danger of over splitting was one motivation for us to design the Blockloading algorithm as a nested algorithm, where expansion and refinement of the cluster assignments are performed in succession.
All cases we have seen of the good local optima have in common that the global optimum could be reached by splits of existing clusters of the estimated cluster assignment. Thus, we call these local optima *favourable* and define them in the following Definition 4.

Definition 4 (Favourable local optimum)**.** *From an (estimated) cluster assignment, $\boldsymbol{Q}_1$ which corresponds to a favourable local optimum, the global optimum, with respect to the ground truth, can be reached by correctly splitting the existing clusters of $\boldsymbol{Q}_1$.*

The opposite of a favourable local optimum is an unfavourable local optimum, where a cluster was at least wrongly split with respect to the ground truth.

Definition 5 (Unfavourable local optimum)**.** *In the case of an unfavourable local optimum, at least one cluster was wrongly split with respect to the ground truth.*

In the case of unfavourable local optima, a cluster can also have been wrongly split and than merged with the wrong clusters according to the ground truth (see the example above).

Similarity of favourable local optima We assume that there are more than two clusters according to the ground truth. Then, a global optimum can be reached, by correctly splitting the right clusters, starting from different favourable local optima. But favourable local optima with the same number of clusters are similar in the sense that a global optimum can be reached by the same number of correct splits.

Up to now, we have used the term local optimum with respect to the estimated cluster assignment of all vertices like in the examples of $\boldsymbol{Q}_1$ or $\boldsymbol{Q}_2$. Now we want to restrict our notion what is a favourable and unfavourable local optimum to subsets. Such a subset could be a cluster of a local optimum which consists of several merged clusters according to the ground truth, like cluster $\hat{k}_3$ in example $\boldsymbol{Q}_1$. If this cluster is split into the true clusters k_3 and k_4 according to the ground truth, it would be a favourable local optimum with respect to the subset of cluster $\hat{k}_3$ of partition $\boldsymbol{Q}_1$.
Moreover, a favourable local optimum with respect to a subset can also be reached from an unfavourable local optimum as we see from the following cluster assignment $\boldsymbol{Q}_4$:

$$\hat{k}_1 = k_1 \cup k_2, \tag{8.15}$$

$$\hat{k}_2 = k_3, \tag{8.16}$$

$$\hat{k}_3 = \{v_{10}\}, \tag{8.17}$$

$$\hat{k}_4 = \{v_9\}. \tag{8.18}$$

We see that a split of cluster $\hat{k}_1$ of $\boldsymbol{Q}_4$ leads to a favourable local optimum with respect to the subset of the clusters $\hat{k}_1 \cup \hat{k}_2$.
We now propose a definition of a favourable local optimum with respect to a subset:

Definition 6 (Favourable local optimum with respect to a subset of the cluster assignment)**.** *A favourable local optimum with respect to a subset can be transformed with split moves to a cluster assignment of this subset which is identical with the same cluster assignment for this subset as for a global optimum.*

We will call it a favourable split move to emphasise the connection to our definition of a favourable local optimum and give a our definition of a favourable split below.
We also propose a favourable merge move, which would mean the merger of the clusters $\hat{k}_3$ and $\hat{k}_4$ of the example above. Such a merger could also be reached by one of our Refinement steps for the cluster $\hat{k}_3$ or $\hat{k}_4$. We note both expressions in the following definitions.

Definition 7 (Favourable merge move)**.** *A favourable merge move leads to favourable local optimum with respect to a subset of the cluster assignment.*

Definition 8 (Favourable split move)**.** *A favourable split move leads to a favourable local optimum with respect to a subset of the cluster assignment.*

8.2.3 Well separateness of clusters in the SBM

For the moment we assume for our discussion of favourable network properties that there is known a known ground truth of the given network which was generated according to a known SBM. We assume that this SBM has $K^\star = 2$ clusters according to the ground truth. We further assume that the solution equal to the known ground truth constitutes also the global optimum of the model selection criterion.
These assumptions make sense, if the network was indeed generated by a 'not too extreme' SBM.
We now want to define what 'not too extreme' means. When we analyse the parameters of the SBM, $\boldsymbol{\lambda}$, which govern the existence of edges and the weight of the edges in weighted networks, we see that there are more clearly defined differences between two clusters in some networks than others. For the Bayesian Poisson SBM, the edge rate parameters which govern the existence of edges in the network are given by $\lambda_{kl} \sim \text{Gamma}(\alpha_{kl}, \beta_{kl}), \forall k, l$.
By definition of the SBM, the clusters, k, k', are considered different if $\lambda_{kl} \neq \lambda_{k'l}$ or $\lambda_{lk} \neq \lambda_{lk'}$ for at least one pair $(k,l), (k'l) \in \{1, \ldots, K\}^2$ holds. This is generalisation to directed (and possibly weighted) networks of Assumption 1 from [25].
Of course, the more densely connected regions of the network are thus more well separated from each other and easier to identify during the clustering process than clusters with a low expected edge existence. So, cluster which have a lot of different pairs of edge rate parameters are also better separated.
Besides of the edge connection parameters, $\boldsymbol{\lambda}$, we also have to consider the number of vertices per cluster. Even if the edge rate parameters differ, but not by a wide margin, two clusters can be easier identified to be different if there are many vertices per clusters.
This observation is also linked to the topic of identifiability of the SBM which was discussed in [25] for the directed Bernoulli SBM. This is also the principle idea of the Largest Gaps (LG) algorithm proposed by [27] (see also [108] for the case of two clusters). The LG algorithm uses the distances between the ordered summed degrees of the

vertices.
So we have a well separated SBM with $K = 2$ clusters, if at least, without restriction of generality, there exists a pair (k',l') so that $\lambda_{kl} << \lambda_{k'l}$ or $\lambda_{kl} << \lambda_{kl'}$ holds and the clusters have at least a certain number or vertices. (According to the identification proposition of [25].)
The expression 'well separated' for different clusters was also used in [103] for the discussion of cluster selection methods for the bisection K–means algorithm.
Now we assume that each of the two clusters of the SBM has a minimum number of vertices per clusters and that $\lambda_{11} << \lambda_{22}$ and $\lambda_{12} << \lambda_{21}$ holds. In this case, we expect from our experience with numerical tests that our fully Bayesian BlockVB++ algorithm will converge to the global optimum of the model selection criterion for (nearly) all randomly initialised start values.
In a next step, we add a third cluster to the SBM and keep the other properties we assumed above. This third cluster can be well separated from the other two clusters or it could have a similar connection profile to one of the existing clusters. If the third cluster, k_3, is well separated, we expect to find the next, favourable local optimum without problems and independently of the start values. Because of the well separation of cluster, k_3, it holds that that at least for one $i \in \{1,\ldots,3\}$ that λ_{3i} or λ_{i3} is clearly bigger or smaller than the existing λ_{kl} with $k,l \in 1,2$.
If the newly added cluster k_3 is not well separated we have without restriction of generality that $\lambda_{3i} \approx \lambda_{1i}$ and $\lambda_{i3} \approx \lambda_{i1}$.
Of course we do not know in advance which clusters are well separated and which are not. We will discuss below, that the LG algorithm allows a good guess of the existence of local optima for certain networks. It even allows the correct identification of bigger clusters under some assumptions we will review below.

8.2.4 Restricting the Expansion step to two new clusters

If we restrict the Expansion step to two clusters, we aim to find one favourable local optima which separates two or more clusters. For the example above, we expect to find during the split to two clusters a well separated favourable local optimum [116]. Moreover, if we have successfully uncovered one favourable local optimum, we can use the resulting cluster assignment as start values for the next Expansion or Refinement step.
If we now assume that all clusters are well separated, we found that it does not matter in which order the local optima are uncovered during the run of our Blockloading algorithm.
We found during numerical tests that the order of the cluster choice is especially not important for the quality of the result (with respect to the model selection criterion), if use our fully Bayesian BlockVB++ algorithm with our adaptive informative priors regardless of the well separateness of the network.
From experience with the numerical tests we know that it is far easier to find one favourable local optimum per Expansion step than two or more at once. We showed that this also follows from the general nature of the SBM and the VBEM algorithm.
In addition, it allows to reuse already correctly identified favourable local optima to approximate the true global optimum. We remark that the step by step identification of local optima is unaffected by network interdependency because we do not restrict the likelihood or the variational bound of the likelihood (free energy). On the contrary, we expect an easier identification of not well separated clusters with the help of already correctly identified favourable local optima (well separated clusters). This is also the reason why we restrict the refinement to one cluster at once: It limits the possible num-

ber of local optima which are involved for the active vertices so that we can achieve an increased precision of the optimisation.

8.3 Different methods for choosing the active cluster

Now that we know the how different local optima and well separateness influence the inference process of the Blockloading algorithm, we use these insights to discuss the cluster selection methods.
We recall how the application of the cluster selection method of the the active cluster, $c^{(active)}$, works for *all* cluster selection methods. Dependent on the available information which includes the adjacency matrix, $\boldsymbol{A}$, the reference cluster partition, $\boldsymbol{Q}^{(ref)}$, and the model parameters, $\boldsymbol{\vartheta}^{(ref)}$, we apply the cluster selection method, which produces an ordered list of the existing clusters according to the chosen selection method.
We choose the $(1+\mathscr{C})$–th cluster on this list (where $\mathscr{C}$ is the number of converged clusters) as the new cluster, $c^{(active)}$. We recall that for $\mathscr{C}=0$ we choose the first entry on the cluster selection list. This is true after a successful Refinement or Expansion step and at the beginning of the main loop of the Blockloading algorithm.
The cluster selection method affects several objectives we have for our Blockloading algorithm. We want to minimise the number of unsuccessful Refinement or Expansion steps which did not improve the reference free energy, $F^{(ref)}$. Therefore we want to choose a cluster which can be split or refined for a maximum improvement of $F^{(ref)}$. In the end we want to achieve the best possible value for $F^{(ref)}$. To meet these objectives, we have to take into account that some clusters are better separated than others (see above). Clusters which are well separated offer a favourable local optima which can be more easily identified.
We consider two cases for the choice of the active clusters:

(i) We have a method, which has a significantly lower computational costs than our BlockVB(++) algorithm, to determine the existence of possible favourable local optima in the clusters.

(ii) We have no prior information about the existence of favourable local optima.

(iii) We have additional objectives which influence the inference process, for example the exclusion of irregularly or sparsely linked clusters or the overall speed.

We now propose three possibilities for the cluster selection method of the active cluster:

(i) Choose of the most densely linked clusters on the cluster selection list (*max–prob–method*).

(ii) Choose the cluster with the most vertices on the cluster selection list (*max–size–method*).

(iii) Choose the cluster with the largest gaps according to the LG algorithm. We expect this cluster to offer the highest potential to hold favourable local optima (*max–gap–method*).

We start our discussion with the Max–Prob–method which selects the most densely linked cluster of the cluster selection list.

8.3.1 Max–Probabilities–method

We find the most densely linked cluster with the help of the expectation value of the distribution of the edge existence probability parameters, $\boldsymbol{\theta}$, in the case of the Bernoulli SBM and the edge existence rates, $\boldsymbol{\lambda}$, in the case of the Poisson SBM. In the case of the Bayesian Bernoulli SBM, the parameters are distributed according to $\theta_{kl} \sim \text{Beta}(\zeta_{kl}, \eta_{kl})$. The expectation value of each parameter can be calculated according to $\mathbb{E}[\theta_{kl}] = \frac{\zeta_{kl}}{\zeta_{kl}+\eta_{kl}}$ [19].
The edge rate parameters for the Bayesian Poisson SBM are distributed according to $\lambda_{kl} \sim \text{Gamma}(\alpha_{kl}, \beta_{kl})$ (see chapter 2) and the expectation value can be calculated according to $\mathbb{E}[\lambda_{kl}] = \frac{\alpha_{kl}}{\beta_{kl}}$ [19].
We use these expectation values of the edge parameters to calculate the densely linked clusters with the help of $\sum_{k=1}^{K^{(ref)}} (\mathbb{E}[\lambda_{kl}] + \mathbb{E}[\lambda_{lk}]) \ \forall l = 1, \ldots, K^{(ref)}$.
The summed edge connection parameters account for the density of the cluster connections.
We introduced the Max–Probabilities–method with the objective to identify the well separated clusters first during the inference process of the Blockloading algorithm.
An additional feature of the Max–Probabilities–method is that we can impose a threshold on the linkage density to exclude sparsely and irregularly connected vertices. Such vertices constitute a vast majority of the vertices of earthquake networks [1].

Threshold for sparsely linked clusters We impose the user defined threshold T on the summed cluster connection parameters. If $\sum_{k=1}^{K^{(ref)}} (\mathbb{E}[\lambda_{kc}] + \mathbb{E}[\lambda_{ck}]) < T$, we can skip the Expansion and Refinement Step for the cluster c. The max-probabilities method avoids this cluster with sparsely connected vertices.
The exclusion of sparsely connected vertices from the active clusters in the first iterations with the max-probability-strategy allows to impose a threshold T on the edge existence probabilities for clusters we consider for optimisation. These vertices are hard to assign to a cluster and it might be useful to classify them as outliers.
Then, after the convergence of all other clusters $k_l \neq k_a$, we can try to split the cluster of outliers in a separate Expansion Step. This allows us to check if any of the two resulting new clusters k_1, k_2 has summed connection probabilities higher than the threshold T. In addition, we can try different start value partition matrices for the same active cluster in the Expansion and Refinement Step, which also decreases the probability to get trapped in an unfavourable local optimum. Otherwise, this cluster can be left out of the Refinement and Expansion Step.
We will also introduce in chapter 12.2 our new Stochastic Block Model with Irrelevant Vertices (SBMIV) which models outliers explicitly contrary to the normal SBM. Then in chapter 13 we will introduce a new VBEM based Relevance BlockVB algorithm together with our new Relevance Blockloading algorithm for inference of the SBMIV. Our new VBEM inference of the SBMIV will need no user specification of the threshold, T.

8.3.2 Max–Size–method

We explained how to find the cluster with the most vertices above in section 8 when we introduced our Blockloading algorithm. Choosing the largest cluster as the active cluster was also proposed as a cluster selection method for the bisecting K–means algorithm in [109, 110] and discussed in [103]. We assume that the network is well separated with respect to our chosen SBM, so that the favourable local optima of the unknown ground truth are found in the first try of the expansion step. Under this assumption, we can even skip the Refinement step. In addition, we assume that we have no prior knowledge of the distribution of the favourable local optima or the clusters and thus assume that all clusters of the ground truth to have approximately the same expected number of vertices per cluster.
Under these assumptions, we expect that the highest number of favourable local optima is located in the biggest cluster which occurs during the inference process.
We also found that this max–size–method needs only $K^{(ref)} - 1$ iterations to split all clusters and another $K^{(ref)}$ iterations to determine convergence.
Under the same assumptions as above, the max–probabilities–method tends to select the smallest cluster, which leads to to $3K^{(ref)} - 1$ iterations until convergence of all clusters. If one cluster per Expansion step is split, we see that we need $\mathscr{O}(K^{(ref)})$ iterations until convergence with both strategies. This was also remarked in [99] without stating the above assumptions.
So, if we do not have, want to or can gain any side information about the existence of the local optima, the max–size–method is expected to be the fastest method.
We also have to consider the resets of the number of converged clusters after a successful refinement or Expansion step of the Blockloading algorithm which also greatly affects the number of steps until convergence. We discuss these resets in section 8.4.

8.3.3 Optimal-Gap cluster selection method

We can use the the Largest Gaps (LG) algorithm, proposed in [27], to determine the potential number of favourable local optima with respect to a cluster. We will review the LG algorithm and also propose its use for finding start values for the SBM in section 9.2.
There, we will propose our original Optimal Gap algorithm which builds upon the same principles as the LG algorithm but is more robust against outliers. Therefore, we postpone our description of the Optimal-Gap method to chapter 9.

8.4 Jump Backs: Reset of the number of converged clusters

In this section we discuss the reset of the number of converged clusters, $\mathscr{C}$, to zero after each successful Refinement or Expansion step of our Blockloading algorithm. If the number of converged clusters, $\mathscr{C}$, is set back to zero the inference process of our Blockloading algorithm starts anew with the first cluster on the cluster selection list, which was returned by the cluster selection method.
This algorithmic design ensures a high number of revisits of the same, already optimised and converged clusters.
The reset of the number of converged clusters after each successful Refinement or

Expansion step ensures that **every** favourable local optimum is found with a high probability before the convergence of the algorithm.
We introduced this algorithmic feature to ensure a thorough optimisation of complicated networks, especially earthquake networks through many revisits of already converged clusters. For small to medium sized networks of approximately thousand to two thousand vertices this strategy achieves a viable computational speed.
When we use the max–size–method for choosing the active cluster, frequent resets of the number of converged clusters can lead to many unsuccessful revisits of the biggest clusters of the network. This is obviously computationally unfavourable.
If, on the other hand, we use the max–probabilities–method we visit each densely linked cluster if we have split one of biggest clusters successfully. These visits of the densely linked (normally smaller clusters) might be very suboptimal if most of the remaining favourable local optima and favourable split and merge moves are located in the biggest clusters. We have seen in the discussion of the choices for the cluster selection method above, that we often encountered this scenario.
Regardless of the cluster selection method, we always face the problem that the first clusters on the cluster selection list do not offer any favourable split moves any longer and are thus have converged with respect to the ground truth. A revisit of one these clusters obviously wastes computation time. It is now our objective to avoid these costly revisits. If we want to find an algorithmic improvement of our Blockloading algorithm for faster computational speed with the same or preferably better quality of the results this is a good starting point for improvements. Of course, we want to keep our algorithmic design of the Blockloading algorithm which is able to find all favourable local optima before the convergence of the algorithm.
In the following discussion, we will use the Refinement and Expansion Step explained in section 8.1 as building blocks to modify the way we:

(i) choose the active cluster

(ii) when and how the active cluster is chosen

(iii) introduce a memory for clusters which did not yield an improvement of the free energy beyond the number of converged clusters

(iv) reduce the amount of jump backs (resets) after a successful Refinement or Expansion step.

In the next section we develop an extended algorithmic framework which builds upon our Blockloading algorithm and uses most of the parts of it.

8.5 Automatic Blockloading

To improve our Blockloading algorithm we want to avoid useless revisits of clusters which do not offer favourable splits or refinements. The first idea to lower the number of Refinement and Expansion steps is to remove the reset of the number of converged clusters, $\mathscr{C}$, to zero after each successful Refinement or Expansion step. Instead of this reset, we propose an automatic increase of the number of converged clusters from $\mathscr{C}$ to $\mathscr{C}+1$ at the end of the main loop after the Refinement and Expansion step.
So, we increment the variable $\mathscr{C}$ regardless of the success of the previous Refinement

or the Expansion step. Therefore, we named this variant of our Blockloading algorithm the automatic Blockloading algorithm
This algorithmic design leads to a faster cycling through the clusters. Of course we might miss some favourable local optima because the inference process proceeded to other clusters **before having the chance to find every favourable local optimum**, which was ensured by the resets of $\mathscr{C}$ in our Blockloading algorithm.
We illustrate this fact with the following example: Assume that we use the max–size cluster selection method. Thus, after the initialisation step we choose the largest cluster as the active cluster, $c^{(active)}$. We assume that $c^{(active)}$ could be split several times because it consists of several favourable local optima with respect to the cluster, $c^{(active)}$. If we automatically increase the number of converged clusters after each iteration of the main loop, we would not select this cluster again during the run time of the algorithm. Moreover, we will only find one of those favourable local optima with respect to this cluster in one iteration of the main loop. Thus we would **miss some local optima necessary to reach the global optimum**, or with other words, the algorithmic design can prevent the identification of a global optimum. To find a remedy to this situation, we introduce restarts of the main loop after the convergence of all clusters. This gives us a chance to discover all favourable local optima in the situation we described above.

Restart free energy We store the reference free energy, $F^{(ref)}$, after the Initialisation step as $F^{(restart)}$. After convergence of all clusters, so that $(\mathscr{C}+1) > K^{(ref)}$ holds, we check if $F^{(ref)} < F^{(restart)}$ holds, e.g. the reference energy was improved after the last restart. If $F^{(ref)}$ was improved we set the number of converged clusters back to zero, e.g. $\mathscr{C} = 0$. If on the other hand $F^{(ref)}$ was not improved since the last restart the automatic Blocklaoding algorithm has converged.
Of course we can also impose a maximum limit of restarts to cut the computational costs.

The convergence requirements of the automatic Blockloading algorithm now demand the convergence of all clusters like in section 8.1 and our newly introduced convergence of the restart free energy, $F^{(restart)}$. We sum up the whole automatic Blockloading algorithm in **Algorithm 8**.
With the algorithmic design of automatic Blockloading we still have the problem, that many clusters could be visited without an improvement of the reference free energy, when most of the local optima are concentrated in a few clusters. We address this problem by introducing an algorithm which we called no reset Blockloading.

8.6 No Reset Blockloading algorithm

We build upon our previous discussion of the Blockloading and the automatic Blockloading algorithm. This time we want to ensure that the inference process can concentrate on those clusters where favourable local optima are located. We want to avoid that the inference process proceeds to other clusters prematurely.

Refinement Success Variable We achieve this objective with our newly introduced success variable, $\mathscr{S}$, which links the Refinement and the Expansion step. The success variable is evaluated at the end of the main loop and stores if the Refinement step was successful, e.g. if $F^{(trial)} < F^{(ref)}$ holds, which in turn influences the incrementation of the number of converged clusters, $\mathscr{C}$. We **do not** perform a cluster selection step

Algorithm 8: Automatic Blockloading Algorithm

Data: adjacency matrix $\boldsymbol{A}$; number of iterations of the Expansion Step, $n_{\mathbb{E}}$; number of vertices N
Result: partition matrix $\boldsymbol{Q}$; number of clusters K; free energy F; model parameters $\boldsymbol{\vartheta}$

```
/* Initialisation                                                    */
```
Initialisation Step
Cluster Selection Step
$F^{(restart)} \longleftarrow F^{(ref)}$
repeat
  Refinement Step
  if $F^{(ref)} > F^{(trial)}$ **then** `/* check if Refinement Step improved` $F^{(ref)}$ `*/`
    update cluster partition and parameters
  end
  Expansion Step
  if $F^{(ref)} > F^{(trial)}$ **then** `/* check if the Expansion Step improved` $F^{(ref)}$ `*/`
    update cluster partition and parameters
  end
  $converged \longleftarrow converged + 1$
  `/* Convergence Step                                               */`
  if $converged + 1 > K^{(ref)}$ **then**
    if $F^{(ref)} < F^{(restart)}$ **then** `/* check if` $F^{(restart)}$ *was improved* `*/`
      $F^{(restart)} \longleftarrow F^{(ref)}$
    else
      break
    end
  end
  Cluster Selection Step
until *convergence of all clusters and* $F^{(ref)}$

after the Refinement step and we **do not** reset the number of converged clusters, $\mathscr{C}$, regardless of the success.

If $F^{(trial)} < F^{(ref)}$ holds, we also update the reference values, $F^{(ref)}$, $\boldsymbol{Q}^{(ref)}$ and $\boldsymbol{\vartheta}^{(ref)}$.

We proceed with the **Expansion step**, we introduced in section 8, to split the active cluster, $c^{(active)}$, into two new clusters in search of an improved value of $F^{(ref)}$.

Contrary to the automatic Blockloading algorithm, we check if $F^{(ref)}$ was improved in the Refinement or Expansion step or in both steps with the help of the refinement success variable. If $F^{(ref)}$ was improved in either step, we **do not increment the number of converged clusters**, $\mathscr{C}$ and just set the refinement success variable back to zero. After the Cluster Selection Step another iteration of the main loop follows.
If no improvement of $F^{(ref)}$ was reached in the last iteration of the main loop, we increment the number of converged clusters from $\mathscr{C}$ to $\mathscr{C}+1$.
We keep the restart mechanic we introduced for the automatic Blockloading algorithm. We also check for the convergence in the same way as for the automatic Blockloading algorithm.
We sum up our whole no reset Blockloading algorithm in **Algorithm 9**.
With the algorithmic design of the no reset Blockloading algorithm, the inference algorithm has one chance to find all local optima before moving on. For some networks this might be not enough. We introduce our Blockoading++ algorithm which allows for a user defined number of retries for identifying all favourable local optima with respect to an existing cluster. At the same time, the Blockloading++ algorithm only allows limited resets of the number of converged clusters.
Thus, our Blockloading++ will strike a balance between avoiding too many costly complete resets of the number of converged clusters and on the other hand allowing for multiple tries to find all local optima on the first try.

8.7 The Blockloading++ algorithm

We now use the insights of our discussion of the Blockloading algorithm in section 8.2 together with the algorithmic ideas we newly introduced in sections 8.1, 8.5 and 8.6, to develop our greatly improved Blockloading++ algorithm. We will see in the numerical tests in chapter 10 that the Blockloading algorithm spends the majority of its runtime applying the Expansion or Refinement step without an improvement of the reference free energy, $F^{(ref)}$.
We explained in the preceeding sections that, the main reason for this waste of computational time are the frequent resets of the number of converged clusters, $\mathscr{C}$, after each successful Refinement or Expansion step. Every time there is a reset of $\mathscr{C}$ to zero, the Blockloading algorithm will restart at the very beginning of the cluster selection list. Our solution that we will propose as a remedy to this behaviour, will lead us to a greatly improved approach for the treatment of resets after a successful Refinement or Expansion step. We propose a refined approach for the storage of the converged clusters and selection of the active cluster. This approach uses concepts and builds upon the Blockloading variants we introduced in the last sections 8.1 to 8.6. We now present our Blockloading++ algorithm in the order of its algorithmic steps.
For this presentation, we use the terminology and algorithmic steps we introduced in sections 8.1 to 8.6. This includes the restart free energy, $F^{(restart)}$ from section 8.5 and

Algorithm 9: No Reset Blockloading Algorithm

Data: adjacency matrix $\boldsymbol{A}$; number of iterations of the Expansion Step, $n_{\mathbb{E}}$; number of vertices N
Result: partition matrix $\boldsymbol{Q}$; number of clusters K; free energy F; model parameters $\boldsymbol{\vartheta}$

```
/* Initialisation                                                    */
```
Initialisation Step
Cluster Selection Step
$F^{(restart)} \longleftarrow F^{(ref)}$
$success \longleftarrow 0$
repeat
  Refinement Step
  if $F^{(ref)} > F^{(trial)}$ **then** `/* check if Refinement Step improved` F `*/`
    $success = 1$
    update cluster partition and parameters
  end
  Expansion Step
  if $F^{(ref)} > F^{(trial)}$ **then** `/* check if the Expansion Step improved` F `*/`
    update cluster partition and parameters
    Cluster Selection Step
    `/* set success back to zero                                     */`
    success=0
  else
    if $success = 0$ **then**
      $converged \longleftarrow converged + 1$
      `/* Convergence Step                                           */`
      if $converged + 1 > K^{(ref)}$ **then**
        if $F^{(ref)} < F^{(restart)}$ **then** `/* check if` $F^{(restart)}$ `was improved */`
          $F^{(restart)} \longleftarrow F^{(ref)}$
        else
          return
        end
      end
      Cluster Selection Step
    else
      success=0
      Cluster Selection Step
    end
  end
until *convergence of all clusters*

the refinement success variable, $\mathscr{S}$, from section 8.6.

New variables for a smarter selection of the active cluster We also introduce several new variables for an even smarter selection of the active cluster, this includes our new shift variable, $\mathbb{S}$, the track variable $\mathbb{T}$ and a barrier variable $\mathscr{B}$. These variables will affect how we perform the cluster selection and determine the convergence of our Blockloading++ algorithm.
The shift variable shifts the cluster selection on the cluster selection. Therefore jump backs on this list after resets of $\mathscr{C}$ are limited when compared to our Blockloading algorithm. The track variable will keep track how often a cluster was chosen as the active cluster without an improvement of the model selection criterion in either in the Refinement step or in the Expansion step. Each time this happens, the track variable is increased.
If the track variable grows bigger than the barrier variable, the shift variable is increased and the track variable is set back to zero.
This newly introduce selection process is able to limit the complete restarts on the cluster selection list.
These new variables for a smarter inference process lead to the introduction of the Cluster Selection and Convergence step++.

Initialisation Step We start our Blockloading++ algorithm with the same Initialisation step as the Blockloading algorithm. So, all vertices are assigned to the same cluster at the beginning and we calculate the reference free energy, $F^{(ref)}$. Then, we choose the active cluster, $c^{(active)}$, according to the cluster selection method and try to improve $F^{(ref)}$ by a split of $c^{(active)}$ into two new clusters.
We evaluate the outcome of this split with respect to $F^{(ref)}$.
If this split was successful, we proceed with our new Cluster Selection Step++ (see below) where we apply the chosen cluster selection method, dependent on the number of converged clusters, $\mathscr{C}$, and the newly introduced shift variable, $\mathbb{S}$, to choose the active cluster, $c^{(active)}$.

Initialisation of the restart free energy Before we enter the main loop, we store the current reference value of $F^{(ref)}$ after a successful Initialisation step in the restart free energy, $F^{(restart)}$, which will be later used to decide if the inference continues after preliminary convergence of all clusters. We recall that we introduced the restart free energy in section 8.5 together with the introduction of our automatic Blockloading algorithm.

Cluster Selection Step++ We choose the active cluster $c^{(active)}$ according to our cluster selection method. We proposed several cluster selection methods in section 8.1. We recall that the choice of $c^{(active)}$ also depends on the number of converged clusters, $\mathscr{C}$.
In the Blockloading algorithm, we first sort all current clusters of the reference cluster assignment, $\boldsymbol{Q}^{(ref)}$, according to the cluster selection method in descending order. Then we choose, dependent on the value of $\mathscr{C}$, the active cluster as the $(1+\mathscr{C})$–th-cluster of this list.

We introduce the new **shift variable**, $\mathbb{S}$, which shifts our choice of the active cluster on this list. Contrary to the Blockloading algorithm, we choose the $(1+\mathscr{C}+\mathbb{S})$–th–cluster as the new active cluster $c^{(active)}$. Therefore, our shift variable, $\mathbb{S}$, prevents total jump backs to the very beginning of the cluster selection list. This shift variable is only incremented until its reinitialisation at a restart. In the reinitialisation in the Convergence step++ below, $\mathbb{S}$ is set back to zero.

Algorithm 10: Cluster Selection Step++

Data: number of converged clusters, $\mathscr{C}$, shift variable, $\mathbb{S}$

build ordered list of clusters according to cluster selection method
choose the $(1+\mathscr{C}+\mathbb{S})$–th cluster on this list

We now generalise this method for an improved handling of cluster resets. Contrary to the total resets of the number of converged clusters in the Blockloading algorithm we now provide an improved reset mechanism dependent on the shift variable, so that we do not jump back to the first cluster on the cluster selection list.
If we applied the Refinement **and** the Expansion to an active cluster without an improvement of $F^{(ref)}$ in either step, we increment our newly introduced **track variable**, $\mathbb{T}$, which keeps track of entirely unsuccessful iterations of the main loop. Then we check if the track variable, $\mathbb{T}$ grew bigger than the fixed **barrier variable**, $\mathscr{B}$. Our barrier variable, $\mathscr{B}$, prevents a premature increment of the shift variable, $\mathbb{S}$.

Main Loop After this initialisation we enter the main loop. The algorithmic framework of alternating between Refinement and Expansion steps remains the same for the Blockloading++ as for the Blockloading algorithm. As we explained above, the Blockloading++ algorithm uses an improved memory for the clusters which did not yield an improved reference free energy, $F^{(ref)}$, in the Refinement or Expansion step.

We start the main loop with the **Refinement step**, we explained in section 8.1.

New Evaluation to the Refinement step The evaluation of the Refinement step of our Blockloading++ algorithm differs from the procedure of the Blockloading algorithm. Our new idea for this improved evaluation is to link the evaluation of the Refinement with the Expansion step and thus to prevent too many resets of the number converged clusters to zero. For this purpose, we introduced our refinement success variable, $\mathscr{S}$, in section 8.6, which stores if the Refinement step was successful. The refinement success variable influences the reset of the number of converged clusters, $\mathscr{C}$. It is evaluated in the new Convergence step++ below.
Like for the no reset Blockloading algorithm, we **do not** apply the cluster selection step after the Refinement step and we **do not** reset the number of converged clusters, $\mathscr{C}$, regardless of the success in the Refinement step.

If $F^{(trial)} < F^{(ref)}$ holds, we also update the reference values, $F^{(ref)}$, $\boldsymbol{Q}^{(ref)}$ and $\boldsymbol{\vartheta}^{(ref)}$.

We proceed with the **Expansion step**, we introduced for the Blockloading algorithm in section 8.1, to split the active cluster, $c^{(active)}$, into two new clusters in search of an improved value of $F^{(ref)}$.

New Evaluation of the Expansion step If $F^{(trial)} < F^{(ref)}$ holds, we update the reference values, $F^{(ref)}$, $\boldsymbol{Q}^{(ref)}$ and $\boldsymbol{\vartheta}^{(ref)}$ once more.
Then we perform our new Cluster Selection step++ where we choose the new active cluster dependent on the number of converged clusters, $\mathscr{C}$ and the shift variable, $\mathbb{S}$.
If $F^{(ref)}$ was not improved, we first check if the preceding Refinement step was successful with the help of the refinement success variable $\mathscr{S}$.
If the Refinement step was successful, e.g. $\mathscr{S} = 1$, we set $\mathscr{S} = 0$, and perform the Cluster Selection step++ before another iteration of the main loop follows.
Otherwise, if the Refinement step was also not successful, we increment the number of converged clusters, $\mathscr{C}$, by one. We also increment the track variable, $\mathbb{T}$, by one.
In the case that neither the Refinement step nor the Expansion step did yield an improvement of $F^{(ref)}$, we check for convergence in our new Convergence Step++.

Convergence Step++ We check for the convergence of Blockloading++ dependent on the number of converged clusters, $\mathscr{C}$, and the shift variable, $\mathbb{S}$. If $(1 + \mathscr{C} + \mathscr{S}) > K^{(ref)}$ holds, all clusters have converged.
We check if we should reset all memory variables for another run of the main loop or if the maximum number of restarts, $\mathscr{R}_{\max}$, was reached.
If $\mathscr{R}_{\max}$ was not reached, we compare the reference free energy, $F^{(ref)}$, with the restart free energy $F^{(restart)}$. If the reference free energy was improved during the current run of the main loop, we update $F^{(restart)}$ and set the memory variables back to zero, e.g. $\mathscr{C} = 0$, $\mathbb{S} = 0$ and $\mathbb{T} = 0$. We increment the number of restarts by one.
We sum up our Convergence step++ in Algorithm 11.

Check for increase of the shift variable If Blockloading++ did not converge we check if the shift variable, $\mathbb{S}$, should be increased. This is the case, if the track variable, $\mathbb{T}$, is now bigger than the fixed barrier variable, $\mathscr{B}$. So, if $\mathbb{T} > \mathscr{B}$ holds, we increment the shift variable, $\mathbb{S}$, by one and set the track variable, $\mathbb{T}$, back to zero. Then we proceed with the Cluster Selection Step++ for the choice of $c^{(active)}$.

Another run of the main loop follows.

We sum up the whole Blockloading++ algorithm in **Algorithm 12**.

Discussion of the shift and converged variables We recall, that every time the track variable, $\mathbb{T}$ grows bigger than the fixed barrier variable, $\mathscr{B}$, the shift variable, $\mathbb{S}$, is increased by one. The number of converged clusters, $\mathscr{C}$, is set to zero after each successful Expansion step whereas the track variable, $\mathbb{T}$, keeps its value. Contrary to $\mathscr{C}$, the track variable is set back to zero each time it grew bigger than the barrier variable $\mathscr{B}$. Both, the track variable $\mathbb{T}$ and the number of converged clusters, $\mathscr{C}$, are increased

Algorithm 11: Convergence Step++

```
if (𝕊 + 𝒞 + 1) > K^(ref) then
    /* check if maximum number of restarts, ℛ_max, is reached */
    if ℛ > ℛ_max then
        break
    end
    if F^(ref) < F^(restart) then
        F^(restart) ⟵ F^(ref)
        /* reset of converged clusters, shift and track variable */
        𝒞 = 0
        𝕊 = 0
        𝕋 = 0
        ℛ ⟵ (ℛ + 1)
    else
        break
    end
end
```

if the Refinement and the Expansion did not improve the reference free energy, $F^{(ref)}$. This way our track variable keeps track of these unsuccessful iterations of the main loop. This is the reason why we introduced the track variable, $\mathbb{T}$, in addition to the number of converged clusters, $\mathscr{C}$.
This approach ensures that a growing number of converged clusters leads to a growing shift variable. It leads to a limited jump back on the cluster selection list when the number of converged clusters is set to zero. Our new procedure for the cluster selection with the variables $\mathbb{T}$, $\mathbb{S}$, $\mathscr{C}$ and $\mathscr{B}$ ensures that the inference process of Blockloading++ focuses on the current neighbourhood of the cluster selection list. We will see that this procedure hugely increases our chances of finding an active cluster where favourable optima are located. We also have a faster cycling through all clusters contrary to the Blockloading algorithm which takes the network interdependency (see section 3.2) better into account.
It also leads to a much faster run time of the Blockloading++ algorithm and reduces the absolute number of Refinement and Expansion steps. We will give examples for this in the numerical experiments in chapter 10.
We will see in chapter 10, that application of the Blockloading++ algorithm leads to a hugely improved runtime compared to the Blockloading algorithm with an improved quality of the results measured by the free energy.

Complexity of Blockloading++ The order of the computational costs we stated for our Blockloading algorithm at the end of section 8.1 also applies to our Blockloading++ algorithm. We saw already in section 8.1 during our discussion of the complexity of our Blockloading algorithm, that it is more important to achieve a minimal number of Refinement and Expansion step without a drop in the quality of the results.
This can be only analysed in numerical experiments. Our aim with the theoretical analysis of the Blockloading algorithm and our new algorithmic design of our Block-

loading++ algorithm was to achieve this objective. We verify in chapter 10 that our Blockloading++ algorithm has indeed a much faster run time for a complicated network than the Blockloading algorithm.

Algorithm 12: Blockloading++ Algorithm

Data: adjacency matrix, $\boldsymbol{A}$; barrier variable, $\mathscr{B}$; maximum number of restarts, $\mathscr{R}_{\max}$
Result: partition matrix $\boldsymbol{Q}$; number of clusters K; free energy F; model parameters $\boldsymbol{\vartheta}$

```
/* Initialisation                                          */
```
Initialisation Step
Cluster Selection Step++
$F^{(restart)} \longleftarrow F^{(ref)}$
```
/* Main Loop                                               */
```
repeat
```
    /* Optimise existing cluster partition                 */
```
    Refinement Step
    if $F^{(ref)} > F^{(trial)}$ **then** `/* Evaluation of the Refinement Step */`
        update $F^{(ref)}$, $\boldsymbol{Q}^{(ref)}$ and $\boldsymbol{\vartheta}^{(ref)}$
        Cluster Selection Step++
        $1 \longleftarrow$ *success*
    end
```
    /* Split active cluster into two new clusters          */
```
    Expansion Step
    if $F^{(ref)} > F^{(trial)}$ **then** `/* Evaluation of the Expansion Step */`
```
        /* update reference free energy, parameters and
        cluster partition                                   */
```
        update $F^{(ref)}$, $\boldsymbol{Q}^{(ref)}$ and $\boldsymbol{\vartheta}^{(ref)}$
```
        /* reset of number of converged clusters           */
```
        $0 \longleftarrow \mathscr{C}$
        Cluster Selection Step++
    else
        if *success=0* **then**
            $\mathscr{C} \longleftarrow (\mathscr{C}+1)$
```
            /* check for convergence                       */
```
            Convergence Step++
```
            /* continue if no convergence was reached      */
```
            $track \longleftarrow (track+1)$
            if $track > barrier$ **then**
                $shift \longleftarrow (shift+1)$
                $0 \longleftarrow track$
            end
            Cluster Selection Step++
        else
            $0 \longleftarrow$ *success*
            Cluster Selection Step++
        end
    end
until *until convergence of* $F^{(ref)}$*, maximum number of restarts*

Chapter 9

Start Value Considerations

The VBEM Algorithm for inference of the SBM converges to a local optimum [50, 73, 8, 116]. Thus, the choice of the start partition matrix $\boldsymbol{Q}^{(start)}$ affects the outcome of the VBEM algorithm. This is also true for our BlockVB and BlockVB++ algorithms for the optimisation of subsets.
In the numerical tests we saw that it takes more iterations of the VBEM algorithm to converge for a start partition matrix which assigns most of the vertices non optimal clusters. This is especially true if we use non-informative or neutral hyperparameters in our BlockVB algorithm.
We also noted during numerical tests, that a start partition matrix which assigns a lot of vertices to a non optimal cluster also often converges to a value of the free energy, F, which is far away from a global optimum. For our Blockloading and Blockloading++ algorithms, we need start values for the Expansion and Refinement Step. The initialisation with a second algorithm, like the Ascending Hierarchical Clustering algorithm [89, 88] below, is only necessary, if we use the BlockVB algorithm.
We will also review the Largest Gaps (LG) algorithm introduced in [27]. We will introduce variant of the LG algorithm for directed networks.
The possibility for application of the LG algorithm for finding start values for inference of the SBM was mentioned in the outlook of [27]. We will show how to find these start values for inference of the SBM with our directed version of the LG algorithm for VBEM inference.
Nevertheless, the LG algorithm can lead to very suboptimal results for small and real world networks because of the possibility of outliers [27].
We will introduce our original Optimal Gap algorithm which builds upon the LG algorithm, where we apply the principle of the LG algorithm to propose a divisive algorithm which is designed to be more robust against outliers than the LG algorithm.
In addition, we will show how the LG algorithm can be used to find the centres of K-means inference for the SBM.
We recall that our BlockVB++ algorithm is used by default with randomly initialised start values to expand the choice of possible start values.
So, the algorithms for finding start values are meant for the use with our BlockVB algorithm and nearly neutral hyperparameters of chapter 5.
The LG algorithm or our original Optimal Gaps (OG) algorithm can be used for prior division of very large networks.
Under some assumptions, our Optimal-Gap cluster selection method can identify clusters where favourable local optima with respect to that cluster are located.

9.1 Ascending Hierarchical Clustering algorithm

To select a start partition matrix for variational EM inference of the SBM, the use of an Ascending Hierarchical Clustering Algorithm (AHC) was proposed in the literature (see e.g. [89], [84], [73], [29]). We review the AHC Algorithm explained in the technical documentation of the mixnet package [89, 88]. The AHC algorithm is a good example of an agglomerative algorithm. We will explain below how to use the AHC algorithm for initialising start values for our Blockloading algorithm.
We follow [89] for the review of the AHC algorithm. For this review we need definitions for the distance between vertices and clusters.
The distance between two vertices i and j of a directed graph is defined as:

$$d(i,j) = \sum_{k=1}^{N} (A_{ik} - A_{jk})^2 + (A_{ki} - A_{kj})^2 = \|A_i - A_j\|^2. \tag{9.1}$$

The distance between two clusters is calculated with the help of the barycentres [89]. The barycentres of cluster q for rows and columns, (g_q^+, g_q^-) , with n_q being the number of vertices in cluster q are given by:

$$g_{qi}^+ = \frac{\sum_{k\in q} A_{ik}}{n_q}, \tag{9.2}$$

$$g_{qi}^- = \frac{\sum_{k\in q} A_{ki}}{n_q}, \tag{9.3}$$

for all vertices $i \in \{1,\ldots,N\}$. The distance between two clusters is defined as the Ward distance between the barycentres:

$$\Delta(q,l) = \frac{n_q n_l}{n_q + n_l} \left(\|g_q^+ - g_l^+\|^2 + \|g_q^- - g_l^-\|^2 \right). \tag{9.4}$$

Now, we can recall the **AHC algorithm** of [89] with some minor changes:
(i) Initialisation: Calculate the distance, Δ, between the clusters (eqn. (9.4)). In the first iteration, vertices are considered as clusters, and the distance $d(\cdot,\cdot)$ is used instead.
(ii) Merging Step: Merge two clusters if their distance Δ is the smallest. If two distances are equal, choose randomly two clusters for merging. The label of the new cluster is the smallest of the two previous labels.
(iii) Calculate the distance between groups.
(iv) Iterate **(i)**, **(ii)** and **(iii)** until the desired number of clusters, K, is reached.

The AHC algorithm is slow, except for small graphs. It was proposed in [89] to limit the AHC algorithm to a subgraph $G^{(sub)}$ with $n_0 < N$ vertices of the original graph. After convergence of the AHC for the n_0 vertices, the other vertices $i \notin G^{(sub)}$ are placed in the clusters with the nearest barycentres.
Another strategy proposed in [89] is to run the K-means algorithm for a user specified number $N_{\max}$ of regularly ordered centres. The centres are $N_{\max}$ vertices of the graph. Then K-means algorithm is run until no change of the centres occurs. We shortly reviewed K-means in chapter 7. The resulting clusters are used as input for the AHC algorithm. This initialisation strategy was also used in [73]. We will use it for tests with the VBEM batch algorithm for the Poisson SBM in chapter 10.
Now, we will explain how we use the AHC algorithm for initialising the start values of our Blockloading algorithm. We want to find the start partition matrix in the Expansion

step in section 8.1. We run the AHC algorithm for all vertices in the active cluster, until we reach $K = 2$ clusters. We use the AHC algorithm only together with our BlockVB algorithm.
After the first iterations of Blockloading, the number of vertices in the active cluster is small enough for most of the clusters, so that we can run the AHC without prior preparation through the K-means algorithm.
We initialise the start partition matrix in the Refinement step in section 8.1 with the AHC algorithm in the following way:

(i) Calculate the barycentres (eqn. (9.2) and (9.3)) of the existing clusters including the active cluster.
(ii) Find the shortest distance $d(\cdot,\cdot)$ for each vertex in the active cluster to one of the barycentres. Place the vertex in these clusters of the start partition matrix $\boldsymbol{Q}^{(start)}$ of the Refinement Step.

We also used a second way to initialise the start partition matrix $\boldsymbol{Q}^{(start)}$ for the Refinement step with random entries:

(i) Find all vertices i in the active cluster a.
(ii) Set all entries of the matrix rows $Q_i^{(ref)} \in \mathbb{R}^{1\times K}\ \forall i \in a$ to zero.
(iii) For each matrix row $i \in a$, draw a randomly initialized vector $Q_i^{(random)}$ with $\sum_{k=1}^{K} Q_{ik}^{(random)} = 1$ and $Q_{ik}^{(random)} \in (0,1)\ \forall k = 1,\ldots,K$.
(iv) Set $Q_i^{(ref)} = Q_i^{(random)}\ \forall i \in a$.

The initialisation with random start values is our default method for all tests and algorithms.
The AHC Algorithm provides no procedure for choosing the centres of the K-means preparation. We will address this issue in the next section 9.2. We will propose ways to find the centres (seeds) of K-means based on the properties of the SBM.

9.2 Review and application of the Largest Gaps Algorithm

We need a computationally viable algorithm to choose the centres of the K-means algorithm for large graphs with many clusters. Other ways tho find the centres of k-means in advance were proposed in [103, 13]. This algorithm should provide a fast approximation of the clusters of the SBM. If we choose the centres for K–means or the subnetwork of the AHC randomly, we might end up with many (or in extreme cases all) centres coming from one cluster. The result is a suboptimal initialisation of the start value partition matrix. In the following we will show how to lower the probability of such a scenario.
For the case of the undirected Bernoulli SBM with $K = 2$ clusters, an algorithm was introduced in [108] which can find the latent cluster assignments of the vertices of the SBM for graphs of $N = 30$ vertices or more. This algorithm uses the summed degrees of all vertices of the graph. The algorithm was expanded in [27] to the case of more than two clusters and provided an asymptotic criterion for model selection. It was called the Largest Gaps algorithm (LG algorithm).

We expand the LG algorithm to the directed Bernoulli SBM and the directed Poisson SBM. We calculate the degree $D_i^{(dir)}$ of vertex i for directed graphs as the arithmetic mean of the in–degree $D_i^{(in)}$ and the out–degree $D_i^{(out)}$:

$$D_i^{(in)} = \sum_{j=1}^{N} A_{ij};\ D_i^{(out)} = \sum_{j=1}^{N} A_{ij};\ D_i^{(dir)} = \frac{1}{2}\left(D_i^{(out)} + D_i^{(in)}\right). \tag{9.5}$$

Now, we define the probability $\overline{\theta}_k^{(dir)} = \frac{1}{2}\left(\sum_{l=1}^{K} \delta_l(\theta_{kl} + \theta_{lk})\right)$ for all clusters $k \in \{1,\ldots,K\}$. Then the degree $D_i^{(dir)}$ is a binomial distributed random variable conditionally on $Z_{ik} = 1$ with parameters $(N-1, \overline{\theta}_k^{(dir)})$. It is $\overline{\theta}_k^{(dir)} \in [0,1]$.
Equivalently, we define for the Poisson SBM
$\overline{\lambda}_k = \frac{1}{2}\left(\sum_{l=1}^{K} \delta_l\,(\lambda_{kl} + \lambda_{lk})\right)$ for all clusters $k \in \{1,\ldots,K\}$. Then, the degree $D_i^{(dir)}$ is a Poisson distributed random variable, conditionally on $Z_{ik} = 1$ with parameter $\overline{\lambda}_k$. It is assumed, that for all clusters $k,l \in \{1,\ldots,K\}$, the separation assumption $k \neq l \Rightarrow \overline{\theta}_k \neq \overline{\theta}_l$ holds [27]. We propose a similar assumption for the Poisson SBM: $k \neq l \Rightarrow \overline{\lambda}_k \neq \overline{\lambda}_l$ for all $k,l \in \{1,\ldots,K\}$.
The smallest gap between the $\overline{\theta}_k$ ($\overline{\lambda}_k$) is defined as in [27]: $\delta = \min_{q \neq r} |\overline{\theta}_q - \overline{\theta}_r|$. It was shown in [27] that a larger gap is to be expected between vertices of different clusters for the Bernoulli SBM. This leads to the LG algorithm. Now we repeat the **LG algorithm** of [27]:

(i) Calculate the vertex degrees $(D_i)_{i=1,\ldots,N}$. Norm the degrees to $T_i = \frac{D_i}{N-1}$.
(ii) Sort the T_i, $i = \{1,\ldots,N\}$ in ascending order:

$$T_i \leq \cdots \leq T_N. \tag{9.6}$$

(iii) Calculate the $N-1$ gaps, $G_i = T_{i+1} - T_i \forall i \in \{1,\ldots,N-1\}$.
(iv) Find the indices of the $K-1$ largest gaps G_{i_j}: $i_1 < \cdots < i_{K-1}$, such that for all $k \in \{1,\ldots,K-1\}$ and for all $i \in V \backslash \{i_1,\ldots,i_{K-1}\}$

$$T_{i_{k+1}} - T_{i_k} \geq T_{i+1} - T_i. \tag{9.7}$$

(iv) Setting $i_0 = 0$ and $i_K = N$, each index i is assigned to a cluster: $i \to k$ with $i_{k-1} < i \leq i_k$.

The calculation of the LG algorithm is very efficient and runs in linear time [27].
We use the vector $(D_i^{(dir)})_{i=1,\ldots,N}$ as the input for the directed Bernoulli SBM. The consistency of the LG algorithm was proved for the undirected Bernoulli SBM in [27]. With similar arguments the consistency of the LG algorithm for the directed Bernoulli SBM holds. The proof of the consistency of the LG algorithm in [27] is based on Hoeffding's inequality which only holds for bounded random variables. So, we cannot prove the consistency of the LG algorithm with the help of Hoeffding's inequality in the case of the Poisson SBM.
The vector of ordered degrees $d = (D_i)_{i=1,\ldots,N}$ is called the ordering vector in [108].
The ordered vertex numbers are stored in the vector s.
We found that the gaps provide a grid for choosing the $N_{\max}$ seeds of the AHC algorithm of Section 9.1. Alternatively, we portioned the vector s into k equally spaced intervals and choose a centre randomly from each interval. We use the LG algorithm to find the start partition matrix in the Expansion step of the Blockloading algorithm in

section 8.1.
Of course, the LG algorithm can be limited to the vertices in the active cluster.

LG algorithm for very large networks We want to use the LG algorithm for large graphs, in the first few iterations of Blockloading without the AHC algorithm to save computational time. Therefore we could simply choose the largest gap $\max_{i=1,\ldots,N-1} G_i$, and divide the vertices accordingly. Often, the active cluster is divided into two new clusters several times, before an optimum of ICL_{ex} or the converged free energy (see chapter 6) is reached. Choosing the largest gap will not necessarily yield the highest decrease of the free energy criterion.
Moreover, it was pointed out in [27] that the LG algorithm is suspect to outliers for smaller networks or real world networks. It was noted in [27] that the LG algorithm was not robust with respect to outliers because every normalised degree has the same weight. One outlier vertex can lead to very suboptimal results, where most of the vertices are assigned to a suboptimal cluster [27].
We found this especially important for earthquake networks, where are many vertices which are strongly connected to all other vertices (hubs) [1]. These hubs have the largest gaps by a wide margin with respect to the other vertices. Picking the gaps between is often not the best choice according to the free energy model selection criterion, though.

9.2.1 Optimal Gap algorithm

We propose a new algorithm for finding the optimal division of the active cluster in the Expansion step. We called this algorithm the Optimal Gap (OG) algorithm. Our OG algorithm can be also used for the initialisation of start values in the Expansion step.
Like the LG algorithm, it uses the sequence of summed and normalised vertex degrees of the network. To increase the robustness of our OG algorithm in contrast to the LG algorithm, we pick the largest gaps and test which split of the cluster assignment into two new clusters according to the chosen gaps results in the best value of the free energy model selection criterion. Then the the OG algorithm is repeated for the new cluster assignment. Therefore the OG algorithm can be used as a divisive algorithm. It was noted in [27], that the LG algorithm can also be applied in a divisive fashion.
Our OG algorithm can detect the outlier vertices we mentioned above, if the improvement of the model selection criterion is suboptimal when compared to the other chosen gaps.
Now, we present the **Optimal Gap algorithm (OG algorithm)**:

(i) Specify a number of gaps $n_g \ll n_a$. Find the n_g largest gaps $G_{i_1}, \ldots, G_{i_{n_g}}$.
(ii) For G_{i_j} divide the vertices according to **(iv)** of the LG algorithm in two clusters.
(iii) Form a start partition matrix $\boldsymbol{Q}_{i_j}^{start}$ where the active cluster is divided according to G_{i_j}.
(iv) Calculate $ICL_{ex_{i_j}}$ for $\boldsymbol{Q}_{i_j}^{start}$.
(v) Find $\min_j ICL_{ex_{i_j}}$.
(vi) Repeat steps **(i)** to **(v)** until no increase of the model selection can be found or the chosen number of clusters is reached.

We choose the start partition matrix $\boldsymbol{Q}_{i_j}^{(start)}$ corresponding to $\min_j ICL_{ex_{ij}}$. Thus, we have a provably good approximation of the partition matrix near a favourable local optimum (see section 8.2.2), with our start value partition matrix $\boldsymbol{Q}_{i_{opt}}^{(start)}$ for $K = 2$ clusters. We avoid outliers by calculating the decrease of the ICL_{ex} for several gaps n_g and not just the largest gap. We remark, that if we apply the OG algorithm repeatedly until the convergence of the free energy criterion is reached, this constitutes a clustering algorithm of its own.
The value of the gaps $\delta_{qr} = |\overline{\theta}_q - \overline{\theta}_r|$, $\forall q \neq r$ varies in most networks. When we run the Blockloading algorithm (or VBEM in general) for low number of clusters K, we observed that the clusters with the highest δ_{qr} tend to be found found first. This can explained by the different degrees of well separateness of the clusters. We discussed well-separateness of the clusters in section 8.2.3. The measure $\delta_{qr}, \forall q, r$ of the gaps is another way to infer the well-separateness of the clusters in the network, if the optimal cluster assignment is unknown. The results of the VBEM inference of the SBM also tend to be better if the values of the gaps are high.

Optimal-Gap cluster selection method We note that a different method was proposed as a cluster selection method for the bisecting K-means algorithm (chapter 7) in [103]. Three methods for the selection of the active cluster were discussed in [103]: Picking the largest cluster, splitting all clusters like in [99] and picking the cluster where the variance of all vertices with respect to the K-means centres was lowest. It was found in [103], that the method of selecting the cluster with the lowest variance of the vertices yielded the best results.
Contrary to the approach in [103], the LG algorithm and our OG algorithm make use of the summed degrees of the vertices. The LG algorithm or our OG algorithm can be used to predict the number of possible favourable splits which can be performed with respect to a chosen cluster.
To achieve this, we compared the gaps which are found within the existing clusters.
The gaps we need for these algorithms are only calculated once (point **(iii)** of the LG algorithm above), at the beginning of the inference process. Then we can reuse them as often as we want.

Optimal-Gap method with Largest Gaps algorithm We select the cluster of all unconverged clusters on the cluster selection list which has the largest gap according to the LG algorithm.

Optimal-Gap method with Optimal Gap algorithm To exclude outliers which can lead to a biased cluster selection, we use our OG algorithm above. We choose the n_G largest gaps and calculate the ICL_{ex} model selection criterion. Then we choose the cluster as the new active cluster, where the ICL_{ex} criterion was improved the most.
In numerical tests, we found the OG algorithm to be less affected by outliers in real world networks than the LG algorithm.
Both methods can be used to predict if a favourable split of a cluster could be performed in the Expansion step. Such a split would lead to a favourable local optimum with respect to active cluster chosen with the Optimal-Gap method.

We conclude that the sequence of gaps between the normalised summed vertex degrees provide a valuable hint where these favourable splits could be found.
We sum up the **Optimal-Gap cluster selection method**:

(i) Pick the n_G largest gaps of the summed degrees of the vertices which arise between vertices in the same not converged clusters.
(ii) Calculate for each gap, g_{i_j}, the resulting value of the model selection criterion, in this case the $ICL_{ex_{i_j}}$ or free energy criterion, if the respective cluster is split into two new cluster according to the chosen gap.
(iii) Choose the gap and split which lead to the best value of the max
(iv) The cluster j where the optimal gap is located, is chosen as the new active cluster.

Start value initialisation with the Optimal Gap algorithm Now, we show how the OG algorithm can be used for the Optimal-Gap cluster selection method of section 8.3. We use the optimal gap which was determined by the OG algorithm to apply this cluster selection method. The vertices of the active cluster are assigned according to this optimal gap to the two new clusters.
If we use the start value initialisation of the LG or the OG algorithm with the same number of gaps, n_G, the Blockloading(++) algorithm will return only one result because VBEM inference is deterministic with respect to the start values which are always the same in this case.
We recall from above, that this procedure can be repeated in a divisive fashion, for very large networks, like for the LG algorithm [27].

Chapter 10

Numerical Experiments

In the preceding chapters we introduced several original algorithms for estimating the Bayesian Poisson Stochastic Block Model (SBM) for a given directed or undirected network. We note that the methods we proposed for the Bayesian Poisson SBM transfer to other types of the SBM too, like the Bayesian Bernoulli SBM. We introduced our new BlockVB and BlockVB++ VBEM algorithms for inference of subsets of the Bayesian Poisson SBM in section 4.2 and chapter 5. We explained how to use our new adaptive informative priors for the fully Bayesian BlockVB++ inference algorithm in chapter 5.
We introduced our new Blockloading algorithm and derived from it the automatic and the no-reset Blockloading algorithms. We used the insights from the discussion of the Blockloading algorithm and its derivatives to develop our even more sophisticated Blockloading++ algorithm.
Now we test and compare all those methods.
For this comparison in the numerical tests we now derive our main criteria and objectives to compare our different algorithms.
All our algorithms which are based on our Blockloading algorithm as well as the batch VBEM inference (algorithm 3) seek to optimise the free energy (ILvb) model selection criterion of the Bayesian Bernoulli or Poisson SBM (chapter 6). Therefore we can compare all our methods dependent on the returned converged free energy. This leads to the *first criterion* we want to answer with the help of the numerical tests:

• Which of our newly introduced algorithms returns the best value of the free energy model selection criterion?

We have seen that the VBEM inference for the SBM depends on the start value assignments of the vertices due to the general algorithmic VBEM framework. One motivation for the introduction of our new Blockloading methods were the unsatisfactory preliminary results of the existing batch algorithm approach for VBEM inference for the SBM especially for complicated networks like earthquake networks. Of course in addition to achieving the best values for the model selection criterion we prefer algorithms which return the same or nearly **the same converged free energy and cluster assignments of the vertices for different start values**. To measure this **reliability** of the different algorithms we compare the returned cluster assignments of the vertices with the help of the *Normalised Mutual Information (NMI)* criterion [112]. The NMI criterion will be reviewed in section 10.2. This leads us to the *second criterion* for our tests:

• Which method is the most reliable method when compared to the best returned cluster assignments of the vertices in all tests, $Q^{(ref)}$?

The technique to measure the reliability of the algorithms by comparing the returned cluster partitions for different start value initialisations when no ground truth is known was also used in [12, 45, 116]. Here and in the case of [116] we used the best of all results as a proxy for the ground truth. We also compare the number of returned number of clusters for different runs of the same algorithm to judge the reliability with respect to finding the optimal number of clusters.
We also compare the run time of all algorithms. We found that finding reliably the best value of the model selection criterion to be the most important objectives for our algorithms. Nevertheless, we need an acceptable run time of each of the algorithms. Therefore we note our *third criterion*:

• Which method has the fastest computational speed?

The run time of the algorithm is the deciding criterion if the other criteria are about the same for two or more methods.
Using these criteria and selected synthetic and earthquake networks, we want to find the best version of our Blockloading or Blockloading++ algorithm we proposed in chapter 8 for the Bayesian Poisson SBM.
We will also compare the results or Blockloading(++) algorithms to the classic batch algorithm of section 4. We provide the detailed specifications of our tested versions of the Blockloading(++) algorithm in section 10.3.

10.1 Comparison to implementations of other algorithms for the inference of the SBM

We will compare our overall best performing Blockloading or Blockloading++ algorithm of the tests to the implementations of selected state of the art inference algorithms for the SBM.
Direct comparisons with downloaded software includes the **'blockmodels' package** (https://cran.r-project.org/web/packages/blockmodels) [78] of [77], the only other state of the art implementation of a Variational EM based inference algorithm for the SBM based on split merge concepts we are aware of. The blockmodels package is implemented in R and its computationally intensive parts in C++. It can be run in parallel. We used the R-package R.matlab (https://github.com/HenrikBengtsson/R.matlab) [17] to transfer data from R to Matlab and vice versa.
We also tested the VBEM batch algorithm for the inference of the weighted SBM (WSBM) of [8], which was implemented in the **WSBM 1.2 package** (http://tuvalu.santafe.edu/∼aaronc/wsbm/) [7]. The WSBM 1.2 package is implemented in Matlab and its computationally demanding parts in mex-C++.
For an indirect comparison with additional methods, we reproduced a test proposed in [30] of a network generated by a large and complex Bernoulli SBM with $N = 10000$ vertices. Results for this test were given in [30] for the following methods the **greedy-ICL algorithm** presented in [30], the **colsbm algorithm** of [85], the **vbmod algorithm**

of [50] and the **spectral clustering** version of [107]. We compare our Blockloading(++) methods with these given results in section 10.4.1.

10.2 Comparison of cluster assignments

In this section we give a short review of the **Normalised Mutual Information (NMI)** criterion which is widely used in the literature, e.g. in [34, 84, 73, 30, 12], to compare the cluster assignments of two exclusive partitions of vertices of a network. This short review of the NMI follows closely the presentation of [112]. For more information about the NMI we refer to [112].
Let there be two possibly different exclusive cluster assignments of the vertices of an network, V, which are given by $\boldsymbol{W} = \{w_1, w_2, \ldots, w_R\}$ and $\boldsymbol{M} = \{m_1, m_2, \ldots, m_S\}$. The first step for the comparison of these two cluster assignments is to set up the *contingency table*. The entries, $n_{ij}, (i,j) \in \{1,\ldots,R\} \times \{1,\ldots,S\}$, of the $R \times S$ contingency table are the number of vertices which are assigned to the cluster w_i and to the cluster m_j, so it holds that $|w_i \cap m_j| = n_{ij}$.
Now, to prepare the calculation of the information theoretic quantities needed for the calculation of the NMI, we calculate the sums, $\sum_{j=1}^{C} n_{ij} = a_i, \forall i \in \{1,\ldots,R\}$ and $\sum_{i}^{R} n_{ij} = b_j, j \in \{1,\ldots,C\}$. It holds that $\sum_{i=1}^{N}\sum_{j=1}^{N} n_{ij} = N$, where N is the number of vertices in the network.
The *entropy* of the partition $\boldsymbol{W}$ is now given by [112]

$$H(\boldsymbol{W}) = -\sum_{i=1}^{R} \frac{a_i}{N} \ln\left(\frac{a_i}{N}\right). \tag{10.1}$$

The entropy measures how many bits are needed on average to encode the partition W [112]. The *Mutual Information* between the partitions $\boldsymbol{W}$ and $\boldsymbol{M}$ is defined to be [112]

$$I(\boldsymbol{W},\boldsymbol{M}) = \sum_{i=1}^{R}\sum_{j=1}^{S} \frac{n_{ij}}{N} \ln\left(\frac{\frac{n_{ij}}{N}}{\frac{a_i b_j}{N^2}}\right). \tag{10.2}$$

The Mutual Information measures the information which is shared between the partitions $\boldsymbol{M}$ and $\boldsymbol{W}$ [112]. The Mutual Information is normalised to account for the possibility of different numbers of clusters of the partitions [30]. One possibility to normalise the Mutual Information is given by [112]:

$$NMI(\boldsymbol{W},\boldsymbol{M}) = \frac{2I(\boldsymbol{W},\boldsymbol{M})}{\max\left(H(\boldsymbol{W}), H(\boldsymbol{M})\right)} \in [0,1]. \tag{10.3}$$

The NMI measures how much information two cluster assignments share [112]. A NMI($\boldsymbol{W}$,$\boldsymbol{M}$) of 1 means that both partitions $\boldsymbol{M}$ and $\boldsymbol{W}$ are identical. The NMI is zero when no information about $\boldsymbol{W}$ can be inferred from $\boldsymbol{M}$ [112].

10.3 Overview of Methods for inference of the Bayesian Poisson SBM

Based on our Blockloading and Blockloading++ framework and our BlockVB and BlockVB++ VBEM inference algorithms, we consider the most interesting and representative combinations of the options our framework offers. We give an overview of

the methods based on our methods we want to test. To find the best inference algorithm for the Bayesian Poisson SBM, based on the criteria we described in the introduction of this chapter, we have to make a choice for each of three algorithmic parts of our Blockloading(++) which constitute one algorithmic method: The Blockloading(++) method, the cluster selection method and the VBEM inference algorithm.
We note that there are many more input specifications with respect to our algorithmic framework, which we presented in the past chapters. These specifications, like the number of iterations or the choice of the hyperparameters or different model selection criteria, could also be compared against each other. Of course it is impossible to implement and test all algorithmic combinations which possibly arise from our framework in a reasonable amount of time. For all of these inputs we had to make some preliminary choices. These choices are partly inspired by preliminary numerical tests we performed during the development of these methods. We do not think that these choices decisively affect the outcome with respect to the main criteria for the comparison of the algorithmic design choices we stated at the beginning of this chapter. To keep the presentation concise we take our 'workhorse model' the Poisson SBM as a representative example for nearly all our experiments. It allows the application to networks with any combinations of weighted, unweighted, directed or undirected edges. Most networks fall into one or several of these categories.
For the test of an unweighted Bernoulli SBM in section 10.4, we will also use our Bernoulli BlockVB algorithm of appendix A.
All our methods were implemented in Matlab and computationally intensive parts in mex-C.

Options of the Blockloading(++) framework We have to choose one of the algorithms which are based on our Blockloading and Blockloading++ framework that we proposed in chapter 8. There, we proposed the Blockloading algorithm (algorithm 7) and based on the Blockloading algorithm the automatic Blockloading algorithm (algorithm 8) and the no reset Blockloading algorithm (algorithm 9). Building upon these algorithms we proposed our more sophisticated Blockloading++ algorithm (algorithm 12).
We need to choose the cluster selection method to use one of our Blockloading(++) algorithms.

Cluster Selection method We presented several possibilities for the cluster selection method in chapter 8, which selects the active cluster out of all not converged clusters. There, we showed that the concept of the maximum probabilities (max–prob, MP) method which starts with the most densely connected clusters could be used to minimise the influence of sparsely connected clusters on the inference process. On the other hand we discussed in chapter 8 that the strategy of starting the inference with the clusters with the most vertices first (max–size method) could be favourable to achieve a lower number of iterations. Therefore, these two strategies seem to be interesting choices. We use both strategies in combination with the Blockloading and the Blockloading++ framework.
The two opposite strategies for choosing the smaller clusters first (min–size method, MS) is in most cases redundant to the max–probabilities method and the strategy of selecting the sparsely connected clusters (min–prob method) is in most cases similar to the (max–size method). We remark that one can construct cases where this similarity does not apply.

Method Abbreviation	Full Name of Method
Bl	Blockloading
aBl	automatic Blockloading
nrBl	no Reset Blockloading
Bl++	Blockloading++
Sta	BlockVB and start values
Bay	BlockVB++ and adaptive priors
MS	maximum Cluster Size
MP	maximum Probabilities

Table 10.1: Overview of the abbreviations of algorithmic parts

We designed our Optimal-Gap cluster selection for very large networks. For the networks we will test below, we do not use the Optimal Gap method because these networks are small enough to be handled with either the max-size or max-density cluster selection method. We will postpone the test of the Optimal Gap cluster selection method to later work, where we expect a larger difference of the results.

Options for the VBEM inference algorithm In addition we have to choose the VBEM inference algorithm for subsets of the vertices of the network. We proposed the BlockVB algorithm (algorithm 4 in chapter 4.2) which we use with nearly neutral hyperparameters for the prior distributions (see chapter 5) together with a separate algorithm for initialising start values.
In this thesis, we use the Ascending Hierarchical Clustering algorithm we reviewed in section 9.1 as the separate algorithm for the start value initialisation but other algorithms like K-means can also be used.
Building upon the BlockVB algorithm we presented our original fully Bayesian BlockVB++ algorithm in chapter 5 (algorithm 5), which is applied with adaptive informative hyperparameters and random cluster assignments instead of start values assignments with a second helper algorithm like the AHC algorithm. The BlockVB algorithm was also used in [116]. We apply both algorithms with ten iterations for the main loop of the algorithm and two iterations of the E–step.

Free Energy and Hyperparameters For comparison of our different algorithms, we should obviously take exactly the same hyperparameters for the (converged) free energy in all tests. We stated and discussed the choice of our default prior hyper parameters for the free energy in chapter 5.

Abbreviations of algorithms and methods We sum up the abbreviations of all algorithms in table 10.1. The abbreviations of all methods we will test below are stated in table 10.2. These methods consist of different algorithmic parts. The abbreviation Bl++ Bay MS refers for example to our Blockloading++ algorithm together with our BlockVB++ algorithm and maximum size cluster selection method.

Method	Blockloading	VBEM inference	cluster selection method
Bl Sta MS	Blockloading	BlockVB start values	max cluster size
Bl Sta MP	Blockloading	BlockVB start values	maximum probabilities
Bl Bay MP	Blockloading	BlockVB++ adaptive priors	maximum probabilities
Bl Bay MS	Blockloading	BlockVB++ adaptive priors	max cluster size
aBl Bay MS	automatic Blockloading	BlockVB++ adaptive priors	max cluster size
nrBl Bay MS	no Reset Blockloading	BlockVB++ adaptive priors	max cluster size
Bl++ Sta MP	Blockloading++	BlockVB start values	maximum probabilities
Bl++ Bay MP	Blockloading++	BlockVB++ adaptive priors	maximum probabilities
Bl++ Bay MS	Blockloading++	BlockVB++ adaptive priors	max cluster size

Table 10.2: Overview of all tested combinations of our algorithmic methods based on the Blockloading algorithm and their abbreviations.

Specifications of the Batch VBEM algorithm The batch algorithm for finding the optimal number of clusters and converged free energy (algorithm 6, chapter 6) can be considered until the year 2013 as the state of the art algorithm for the application of Variational (Bayesian) EM methods to the SBM [35, 50, 84, 73].
We apply the VBEM batch algorithm (algorithm 3, chapter 4) with twenty iterations of the main loop and five iterations of the E–step. As explained in chapter 6, we have to choose different numbers of clusters $K_1, \ldots, K_k$. For each current number clusters, K_i, we restart the batch algorithm 30 times with different start values for the cluster assignments of the vertices.
Following [73], we use a combination of the K–means algorithm (see e.g. [19]) and the AHC algorithm (section 9.1) for initialising start values for the cluster assignments of the vertices. The K–means algorithm is initialised with twice the number of the current number of clusters and run for forty iterations. Then we use the resulting cluster assignments as the input for the AHC algorithm, which is run until the number of clusters we want is reached.

10.4 Synthetic Networks

10.4.1 Comparison with existing methods

We compare our Blockloading and BlockVB algorithms for the Bernoulli SBM, we propose in appendix A, with existing methods. We reproduced a test performed in [30] for large graphs with a complex structure. The greedyICL algorithm presented in [30], the colsbm algorithm of [85], the vbmod algorithm of [50] and the spectral clustering algorithm in the version of [107] were tested in [30]. The greedICL uses greedy optimisation, the colsbm a collapsed Gibbs Sampler and the vbmod algorithm uses a VBEM batch algorithm for the restricted SBM [30].
The test networks are generated with the Bernoulli SBM of chapter 2. There are $N = 10000$ vertices, $K = 50$ clusters. The probabilities for the cluster assignments are given by the vector $\boldsymbol{\delta} = (1/50, \ldots, 1/50) \in \mathbb{R}^{1\times 50}$. The probabilities for the existence of an edge are generated according to

$$\theta_{kl} = \begin{cases} ZU + (1-Z)\varepsilon, \text{ if } k \neq l \\ U, \text{ if } k = l \end{cases} \tag{10.4}$$

with $Z \sim \text{Bernoulli}(0,1)$, the uniform distribution $U \sim \mathscr{U}(0.45)$ and $\varepsilon = 0.01$.
In [30], the greedyICL algorithm was applied to 20 simulated graphs. There, it was found that the greedyICL algorithm outperformed the other algorithms with an average NMI of 0.88 between the real cluster partition and the calculated one. The greedyICL returned $K = 80$ clusters most of the time and the colsbm algorithm more than 240 clusters [30]. The spectral clustering algorithm was initialised with the correct number of clusters according to the ground truth [30].
We applied our Blockloading algorithm for the Bernoulli SBM to 20 simulated graphs. We used the AHC algorithm of Section 9.1 with a subnetwork for the Expansion Step and a random sub matrix for the Refinement Step. The active cluster was chosen as the cluster with the most vertices (max-size-method) (see section 8.3.2).
We found that our Blockloading algorithm identified the correct number of clusters, $K = 50$ and the correct partition with a NMI of 1.0 (each vertex was placed in the correct cluster) for each of the 20 runs.
We also tested the WSBM 1.2 package [7] and the Blockmodels package [78]. The WSBM 1.2 package, which we initialised with Bernoulli or Poisson distributions for $K = 50$ clusters, crashed. The Blockmodels package initialised for the Bernoulli SBM took more than two ours to perform the first split to two clusters, and we cancelled the test.
We conclude that our Blockloading algorithm outperforms the other algorithms mentioned above in this test. We summarise the test results in table 10.3. The Blockloading algorithm is the only algorithm of the mentioned algorithms in this Section, which is able to identify the correct number of clusters and to achieve the maximum NMI score.

10.4.2 Large and complex Poisson SBM

In order to test the Blockloading algorithm for the Poisson SBM we constructed the following example graph which was generated as explained in chapter 2. Each generated graph has $N = 10\,000$ vertices and $K = 50$ clusters. The vector of the Multinomial distribution is given by $\boldsymbol{\delta} = (1/50, \ldots, 1/50) \in \mathbb{R}^{1\times 50}$. We draw $Z \sim \text{Bernoulli}(0.2)$

method	mean of NMI
Blockloading	1.0
greedyICL	0.88
spectral clustering	0.71
colsbm	0.67
vbmod	0.66

Table 10.3: Comparison of other clustering algorithms with our Blockloading (Bl Sta MS (Bernoulli version)) algorithm for complex network with $N = 10000$ vertices generated by Bernoulli SBM. All test results except for Blockloading reproduced from [30]. Mean value of NMI for twenty tests.

and set $\varepsilon = 1$. The rates of the edge weights λ_{kk} within the clusters are set in ascending order to $(0.1, 0.2, \ldots, 1, 1.5, 2, \ldots, 21) \in \mathbb{R}^{1\times 50}$. We set the parameters $\lambda_{k\cdot}$ according to

$$\lambda_{kl} = \begin{cases} Z\lambda_{kk} + (1-Z)\varepsilon, \text{ if } k \neq l \\ \lambda_{kk}, \text{ if } k = l \end{cases}. \tag{10.5}$$

The generated graph has an asymmetric structure and varying edge weights.
Again, we used the AHC algorithm for the Expansion step and the random initialization for the Refinement Step.
We generated 20 realisations of the graph and ran the Blockloading algorithm for the Poisson SBM. We found that Blockloading returned the correct number of clusters for each test. We calculated a NMI of 1.0 for each comparison between the calculated partition matrix $\boldsymbol{Q}$ and the true partition matrix $\boldsymbol{Q}_{true}$.
This example shows, that our Blockloading algorithm is able to reliably calculate correct results for large weighted graphs with complex structure.

10.5 Earthquake Networks

The earthquake network (EN) was introduced by S. Abe and N. Suzuki in 2004 to model the spatiotemporal time series of earthquakes [1]. We first review the construction of the earthquake network. Then we will discuss several properties and applications for the earthquake network given in the literature. The main focus of this thesis is that we will analyse the earthquake network from the perspective of model based clustering with the Poisson Stochastic Block Model (SBM), introduced in chapter 2. It was shown in ([1, 2, 3]) that important statistical properties of earthquake activity are inherited by the EN.
In section 10.5.1 we review how to construct the earthquake network.

10.5.1 Construction of the Earthquake Network

The earthquake network (EN) is constructed for a chosen geographical area and time span. A grid of squares is placed on the area of interest. According to this grid, we can construct cubicle cells to include depth information.
When we construct the square grid, we have to take into account the curvature of the earth surface and therefore need a projection of the curved geographic area of interest. Such an approach takes the departure into account.
There are several ways to choose a projection method to account for the departure. The authors of e.g. [6] used Mercator coordinates of a sphere to approximate the surface of earth. We use Universal transverse Mercator (UTM) coordinates of the WGS 84 ellipsoid following [63]. To calculate the UTM coordinates with high precision, we used the accompanying software of [63]. We downloaded this software, called Geographic Lib [64], at https://sourceforge.net/projects/geographiclib/files/distrib/ .
The earthquake network unfolds in the following way [1]:

(i) Place a vertex in the first cubicle cell where an earthquake occurs the first time during the observation interval.
(ii) Place a second vertex in the cubicle cell where the next time seismic activity occurs and place a (directed) edge between the last two vertices of seismic activity pointing to the latest vertex of activity.
(iii) If seismic activity occurs in direct succession in the same cell, place a self edge (also called tad pole or loop) connecting the vertex in the cell with itself.
(iv) Continue until the end of the chosen time span.

The placement of a self–edge according to **(iii)** models a direct after shock [1]. By construction, multi–edges are possible if earthquakes arise in direct succession in two cubicle cells/vertices already linked by an edge. We observed that multi–edges arise very often in the earthquake networks we analysed. We treat all multi–edges which connect two vertices in the same direction as one weighted edge. So, the earthquake network is by construction a directed and weighted network with self–edges.
The only parameter which has to be set in advance besides the geographic area and the observation time span is the side length of the cubicle cells [2].
In order to allow the application of methods which can only be applied to undirected networks without weights and self edges, these features of the earthquake network were removed in e.g. [2]. Thus, the clustering coefficient for networks introduced in [114] could be applied to a simple earthquake network without loops, edge directions and edge weights in [2]. The same applies to application of the modularity measure of [94]

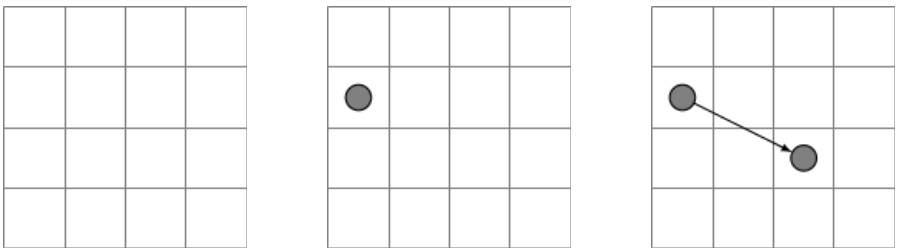

Figure 10.1: Schematic description of the construction of the earthquake network introduced in [1]. First, we put a grid of cells on the area of interest (left side). Second, we place a vertex in the first cell where seismic activity occurs at the start of observation interval (middle). Third, we place a second vertex where the next time seismic activity occurs and place an edge between the last two vertices of seismic activity. If there is seismic activity between existing vertices, we place an edge between those two vertices. This continues until the end of observation.

to earthquake networks where loops, edge directions and weights were removed for finding communities in [5].
We mentioned in chapter 2, that we would need additional terms for the SBM to model loops [123]. Inference algorithms for the SBM are normally discussed and stated for the SBM without loops in the literature [97, 35, 72, 84, 40, 8]. Tests and benchmarks for inference of the SBM are also applied to networks without loops in the literature [35, 84, 73, 40, 8]. So, the application of the SBM without loops allows for the comparison with other inference algorithms and their implementations presented in the literature. Therefore, we opted to remove the loops of all networks.

Magnitude of Completeness The magnitude of completeness, M_c, is an important concept to ensure with probability one that no seismic event of a magnitude larger or equal than the magnitude of completeness was missed during the time interval of observation for a given data set [90, 53]. Determining the correct magnitude of completeness for given earthquake data is an essential and mandatory first step before we can start further analysis of the data [90]. It is noted in [90] that finding the right magnitude of completeness with one of the state of the art methods is not a trivial task. If we choose the M_c too high we would unnecessarily exclude events from our investigation and if otherwise M_c is chosen too low, the data would be biased [90].
Contrary to e.g. [1, 2, 6, 3], where the concept of the magnitude of completeness is not used for the construction of the earthquake network, we will only include data according to the magnitude of completeness for the construction of all our earthquake networks.

10.6 Southern California Earthquake Network

The earthquake catalogs of the Southern California Earthquake Data Center (SCEDC)[104] is an often used data base for the construction of an earthquake network e.g. in [1, 2, 3, 6]. We downloaded the catalog data from (http://scedc.caltech.edu/) [104]. We constructed the earthquake network of the Southern California area (32s, 37n; 122w, 114w),

for the time interval from January 1, 1984 to December 31, 2015 (see section 10.5.1). The authors of the studies [1, 2, 3, 6] also constructed the earthquake network for the time interval from January 1, 1984 onwards. The choice of the start of the time interval beginning from January 1, 1984 was explained in [3] with an incompleteness of the data for the year 1983. It was remarked in [2] that the 'typical size of a fault' is 5 km and therefore the minimum size of the cubicles should be at least 5 km. We chose a side length of 10 km for the cubicle cells.
This results in 4256 cubicle cells per depth level.
In the recent study of [53], which is also downloadable from the SCEDC website (http://www.scedc.caltech.edu), the magnitude of completeness for the Southern California Seismic Network (SCSN) earthquake catalog was found to be $M_c = 3.25$ for all local events in the catalog beginning with the start of recording in 1932 and $M_c = 1.8$ for all data from 1981 onwards. Other events which were recorded in the SCSN catalog like artificial quarry blasts or sonic booms were excluded from the study in [53]. Therefore we also only included local events in the construction of our earthquake network of Southern California.
So, we only used local events with a magnitude $M \geq 1.8$ on the Richter Scale for the construction of the SCEN. There occurred a total of 143918 earthquakes which meet these criteria in the chosen time interval for the construction of the network.
The resulting adjacency matrix of the network has $N = 3163$ vertices and 68654 edges. We did not include unconnected vertices, which model regions without earthquake activity during the observation time, into the construction of the network. We removed all self–edges which model direct aftershocks from the network by setting the diagonal of the adjacency matrix to zero. The maximum edge edge weight is 222 and the minimum 1. The network is directed by construction. The maximum depth of an event during the observation time is 29.9 km. Therefore we have three depth levels in our earthquake network.
We start our tests with the SCEN in the next section 10.7.

10.7 Methods for inference of the Poisson SBM for the Southern California Earthquake Network

Literature about the application of clustering methods to the earthquake networks of [1] is scarce. The application of clustering according to the modularity measure of [94] to earthquake networks, where edge weights, edge directions and loops were removed, was proposed in [4]. Calculating the modularity measure aims at finding the most densely connected vertices in the network, called communities [94]. Contrary to being able to only find communities with the modularity measure of [94], the Stochastic Block Model cannot only identify densely connected vertices as clusters but also heterogeneous structures of the network [71]. We recall that the SBM clusters vertices according to the same edge connection behaviour, which also includes hubs, disassortative and assortative mixing [71]. The existence of hubs in earthquake networks was shown to be the consequence of big earthquakes (mainshocks) in [1]. These hubs were considered to be important parts of the earthquake network [1].
We recall that our main objective in this thesis was to find a computationally viable way for high quality estimation of the SBM for a given earthquake network. To the best of our knowledge, the estimation of the SBM for the analysis of earthquake networks of [1] or the estimation of model of comparable complexity has never been done

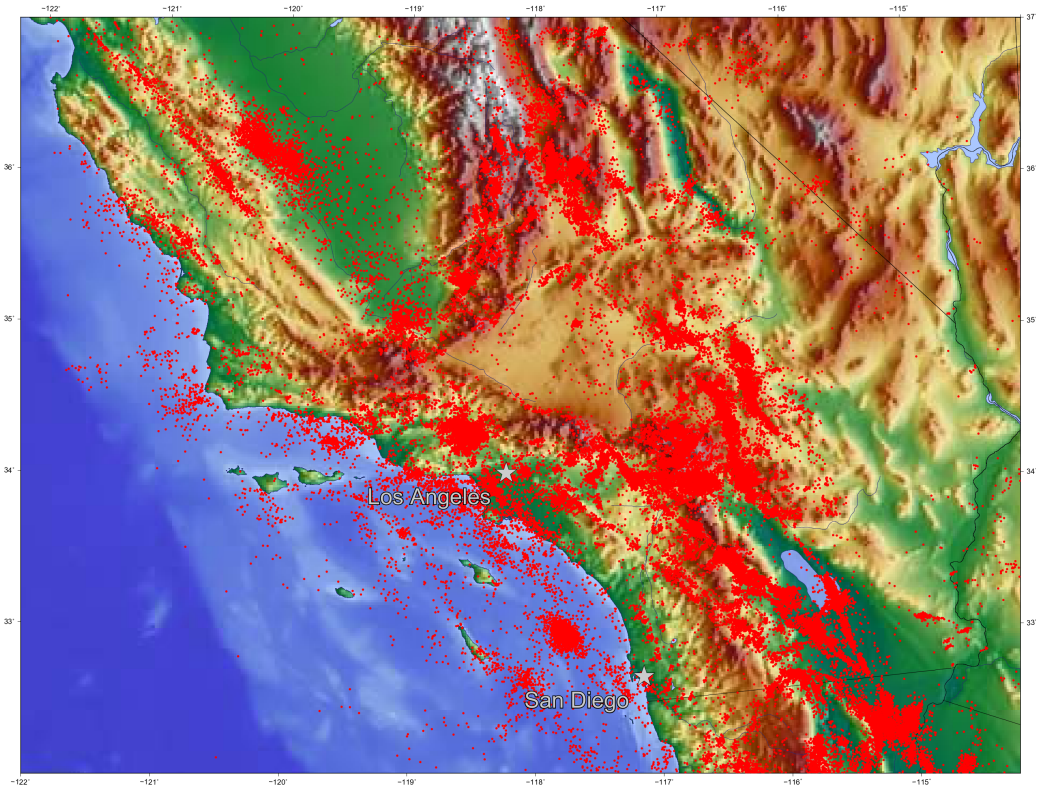

Figure 10.2: Map of the Southern California area under consideration for the construction of the earthquake network. All earthquakes from 1982 to 2011 marked with a red dot. Map was drawn with the Generic Mapping Tools (GMT) [115]. Earthquake data provided by [104].

or proposed before. In the next section we show the results of the application of our new estimation algorithms for the Poisson SBM to the Southern California earthquake network of section 10.6.

10.7.1 Test of Blockloading(++) methods

We now want to find the best of our methods based on the Blockloading, Blockloading++, BlockVB and BlockVB++ algorithms for the Poisson SBM. For the tests in this section, we use the directed and weighted earthquake network of the Southern California area which we presented above in section 10.6. We stated the main test criteria at the beginning of this chapter. We described the methods we want to test in section 10.3.
We did 30 runs of each algorithm. All our algorithms we will test now are listed in table 10.2.
We used the same machine with an Intel i5 3570K CPU with four 3.4 Ghz cores for all tests in this section.
We start our evaluation of the tests of the SCEN of section 10.6 with the first criterion.

Evaluation of Model Selection Criterion The best result (lowest converged free energy) of all algorithms and tests was returned by our Blockloading++ algorithm with fully Bayesian BlockVB++ inference algorithm and maximum cluster size selection method (Bl++ Bay MS) with a converged free energy of $F^{(ref)} = 179924$ and an optimal number of clusters of $K^{(ref)} = 56$. The run time of this best run was 2 minutes and 38 seconds.
The highest converged free energy of all test runs (worst result) was returned by the Blockloading algorithm with the BlockVB inference algorithm and maximum size cluster selection method (Bl Sta MS). It returned a converged free energy of $F^{(ref)} = 183932$ and $K^{(ref)} = 44$ clusters.
We present a detailed graphical presentation with the help of a box plot of the results of the converged free energy for all tests and methods in figure 10.3.
We remark that even the worst result of all test runs based on our Blockloading(++) algorithms returns a clearly better value of the free energy than the VBEM batch algorithm as we will see in section 10.7.2 below.

Reliability We calculate the Normalise Mutual Information (see section 10.2) of the cluster assignment of the best result, $\boldsymbol{Q}^{(ref)}$, according to the free energy criterion, compared to all other cluster assignments to assess the reliability of the different algorithms. We present the results of the reliability measured this way in figure 10.4.
All our Blockloading and Blockloading++ algorithms achieved an NMI of bigger than 0.8 compared to the best result. Our best performing method in this regard is again our Bl++ Bay MS algorithm with a NMI bigger than 0.9 for all initialisations. Therefore our Bl++ Bay MS algorithm is the most reliable of all tested methods.

Number of Clusters An overview of the number of clusters returned in all tests is given in figure 10.5. Finding approximately the same number of clusters for different initialisations is also an important criterion when we assess the reliability of our algorithmic methods. The optimal number of clusters of all test runs, according to the free energy models selection criterion, is $K^{(ref)} = 56$. All methods which employ the

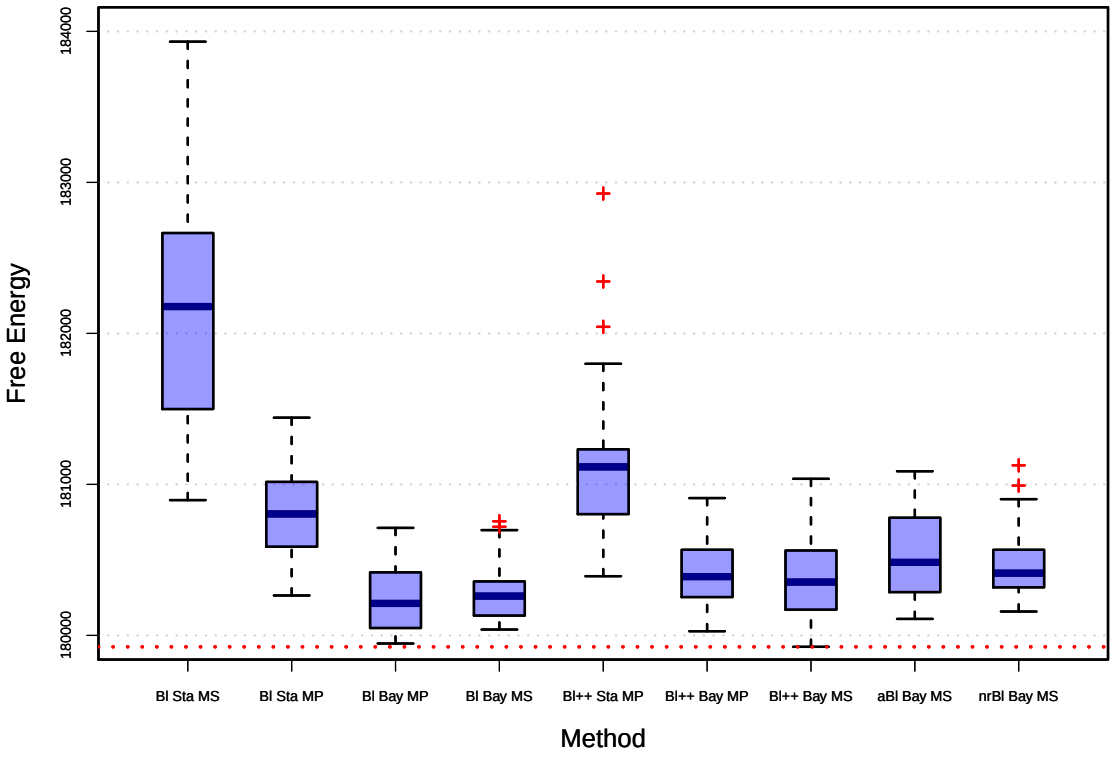

Figure 10.3: Boxplot of the converged free energy of different Blockloading and Blockloading++ algorithms applied to the Poisson Stochastic Block Model for the Southern California earthquake network from 1984 to 2015 with edge weights, edge directions and without loops. The full names of all abbreviations are given in tables 10.1 and 10.2 in section 10.3. The best result (lowest converged free energy), marked by the red line, was returned by the Blockloading++ algorithm with BlockVB++ inference and max cluster size method (Bl++ Bay MS), with a converged free energy of $F^{(ref)} = 179924$.

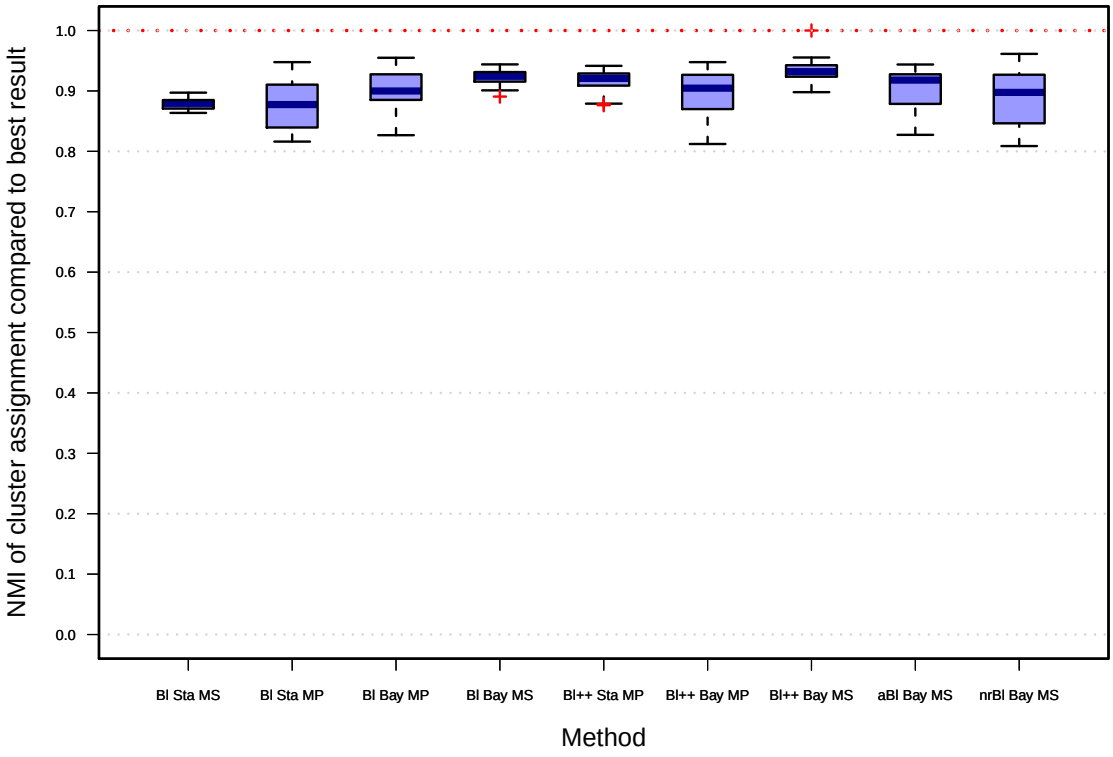

Figure 10.4: Boxplot of the Normalised Mutual Information (NMI) of all cluster assignments compared to the reference cluster assignment, $\boldsymbol{Q}^{(ref)}$. The cluster assignment matrix $\boldsymbol{Q}^{(ref)}$ yields the best converged free energy, $F^{(ref)}$. A NMI of 1.0 signals similar cluster assignments and is the best possible value of the NMI. The full names of all abbreviations are given in tables 10.1 and 10.2 in section 10.3. All Blockloading methods have an NMI better than 0.8. The best method, the Blockloading++ algorithm with BlockVB++ maximum size selection method (Bl++ Bay MS) reaches a NMI of at least 0.9. The reference cluster assignment, $\boldsymbol{Q}^{(ref)}$, reaches a NMI of 1.0 by definition, and is marked by the red line.

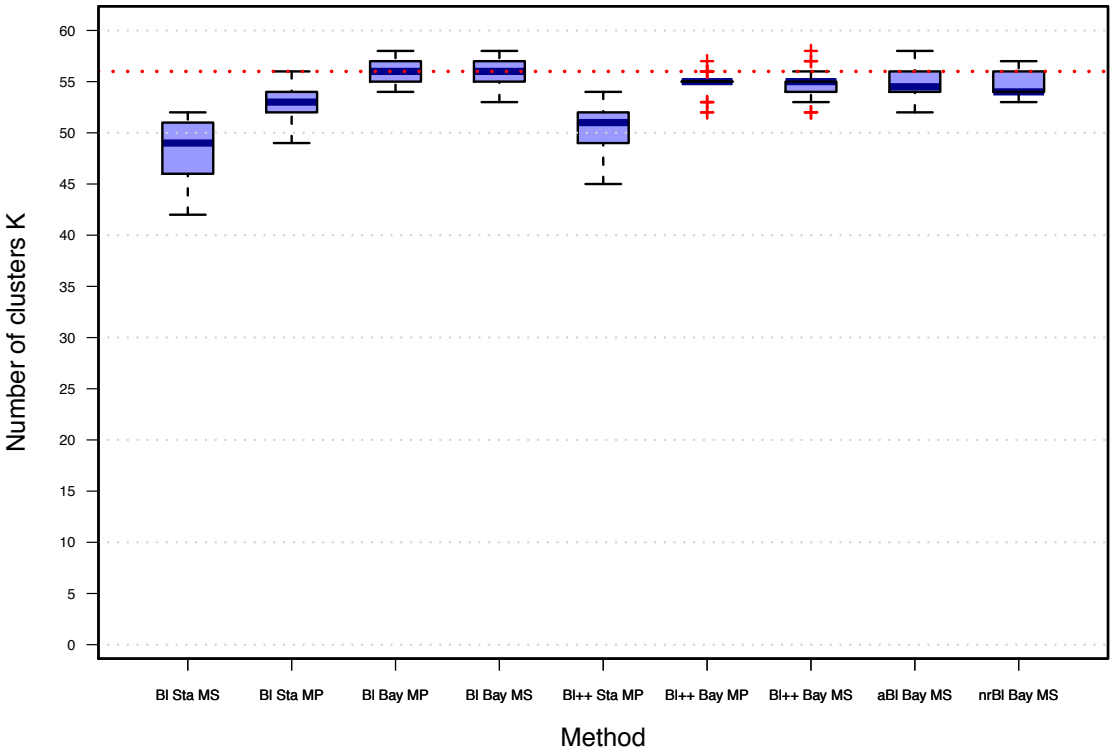

Figure 10.5: Boxplot of the number of clusters, K, returned by different Blockloading(++) based methods.The full names of all abbreviations are given in tables 10.1 and 10.2 in section 10.3. The optimal number of clusters, $K^{(ref)} = 56$ of the best result of all tests according to the free energy model selection criterion is marked with a red line. We see that all fully Bayesian BlockVB++ based methods return a number of clusters, K, which is close to the optimal number of clusters, $K^{(ref)}$.

fully Bayesian BlockVB++ inference algorithm with our adaptive informative prior parameters return cluster numbers around the optimal number of clusters with $K = 52$ to $K = 58$ clusters. Therefore, we conclude that our BlockVB++ algorithm increases the reliability of finding the optimal number of clusters when compared to the traditional start value approach.

Run Time Our last main criterion was the run time of our different algorithms. Here the results are clearly divided between the use of our Blockloading and our more sophisticated Blockloading++ algorithm. The fastest run time of 2 minutes and 5 seconds was achieved by the Blockloading++ algorithm with BlockVB inference and maximum probability cluster selection method (Bl++ Sta MP). As stated above, the computational time of the best result, measured by the free energy model selection criterion, which

was returned by the Bl++ Bay MS method, took 2 minutes and 38 seconds. Nearly all runs of our more sophisticated automatic Blockloading, no reset Blockloading and Blockloading++ based methods were faster than ten minutes.
On the contrary, the run time of all our methods based on the Blockloading algorithm took ca. 15 to 55 minutes. The fastest median run time of a Blockloading based method was more than 27 minutes. On the contrary the slowest median run time of a Blockloading++ based method was ca. 6.5 minutes and the fastest median run time of the Blockloading was under 4 minutes. Therefore our advanced Blockloading++ algorithms offer an at least fivefold increase when compared to the Blockloading based methods.
All results of the run time are presented in figure 10.6. We conclude that our efforts to improve our Blockloading to the Blockloading++ algorithm yield a dramatic increase of the computational speed.
We take an in-depth look at the results returned by the Blockloading and Blockloading++ algorithm to find out where this high increase of speed with better results for the free energy model selection criterion comes from. We recorded the value of the returned free energy and the reference free energy after each Refinement and Expansion step. We will analyse these records below.

Speed of Convergence and the Number of Iterations When we stated the order of the computational costs of our Blockloading algorithm in section 8.1, we claimed that the run time needed for our Blockloading algorithm and its variants mainly depends on the number of Expansion and Refinement steps until convergence. The Expansion and Refinement are each performed once per iteration of the main loop in all algorithms we presented in chapter 8.
We documented the number of iterations of the main loop for all test runs and present the results in figure 10.7.
There we can see that the total number of iterations of the main loop needed until convergence is clearly linked to the run time we presented in figure 10.6. We see that there is clear gap of the number iterations needed for Blockloading based methods compared to automatic, no reset or Blockloading++ based methods.
The highest number of iterations needed being 6886 for the Bl Bay MS method and the lowest 346 for the Bl++ Sta MP method. The best run or our Bl++ Bay MS method needed a total of 437 iterations.
We conclude that we successfully decreased the number of iterations and therefore the total number of Expansion and Refinement steps with the introduction of our Blockloading++ algorithm while at the same time increasing the quality of the best results.
We recorded the value of the converged free energy after each Refinement and Expansion step for all tested methods. The results of the best run of each method, with respect to the free energy criterion, is presented in figure 10.8. So, this figure shows exemplarily the speed of convergence of our different methods.
It shows once more the difference in speed of the convergence of the free energy between methods based on the Blockloading algorithm and methods based on the automatic- and no-reset Blockloading and the Blockloading++ algorithm. We also see that the Blockloading algorithm spends a lot of Expansion and Refinement steps for an only slight improvement of the free energy towards the end of run time. Contrary to these slight increases, the Blockloading++ algorithm has a much faster improvement of the free energy and spends a lot less time for slight improvements of the free energy towards the end of the run time. This was to be expected of our preceding analysis of

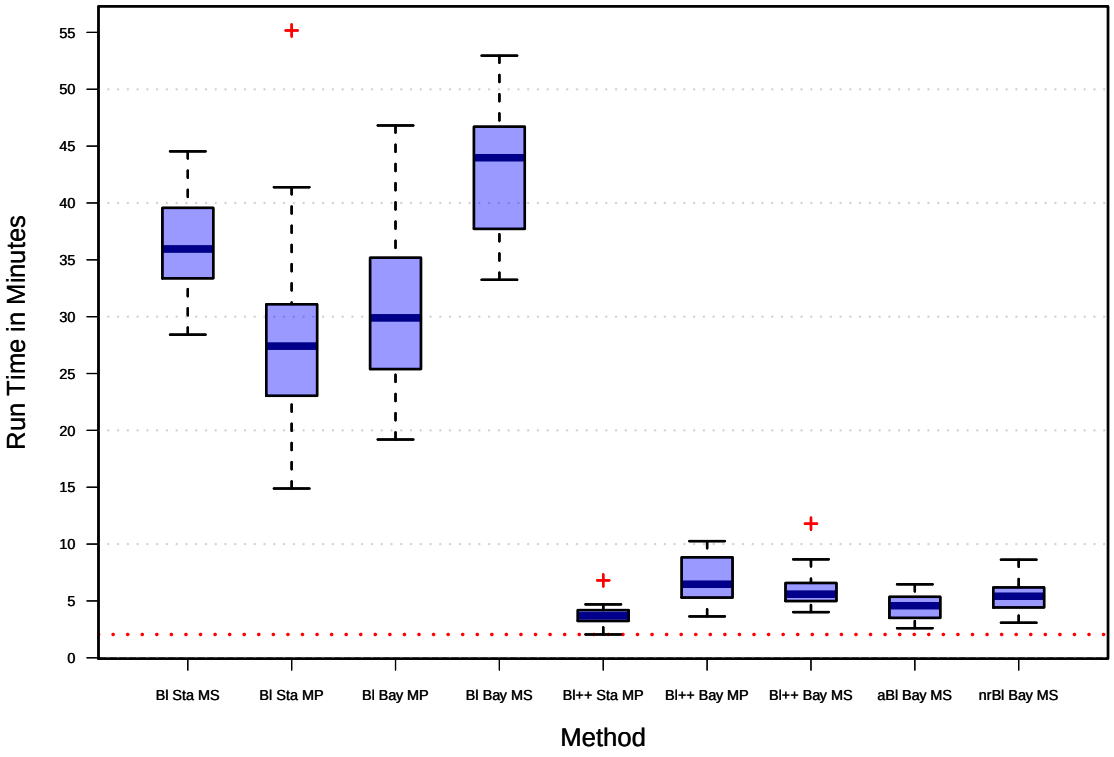

Figure 10.6: Boxplot of the run time of the tested Blockloading(++) methods. The full names of all abbreviations are given in tables 10.1 and 10.2 in section 10.3. We see that there is clear gap of the computational speed between methods which are based on our Blockloading algorithm and our more sophisticated Blockloading++ algorithm. Nearly all methods based on the Blockloading++ algorithm achieve run times of less than ten minutes, with the fastest run time being ca. 2, marked with a red line, and the longest 11 minutes, whereas the run time of methods based on the Blockloading algorithm require a run time of ca. 15 to 55 minutes.

the results we presented above.

Conclusion of Blockloading and Blockloading++ tests for Southern California Earthquake Network We have tested several configurations of our methods based on our Blockloading, automatic Blockloading, no reset Blockloading and Blockloading++ algorithm. Our most important criterion was the best value of the free energy model selection criterion which could be achieved by the method. This was achieved by the Blockloading++ algorithm with BlockVB++ inference and maximum size cluster selection method (Bl++ Bay MS) with a value of $F^{(ref)} = 179924$ and $K^{(ref)} = 56$ clusters and a run time of 2 minutes and 38 seconds which is close the overall fastest run time of all tested algorithms. This method had also the highest reliability when measured by the NMI calculated in comparison to the best result of all tests measured by the free energy. All returned numbers of clusters by this method were very close to the optimal number of clusters. In addition, the run time of the Bl++ Bay MS method is one of the fastest of all tested methods.
We will see below that all tested methods we proposed in chapter 8 brought a dramatic increase in quality of the results, reliability and computational speed when compared to the batch algorithm.
The fully Bayesian BlockVB++ algorithm with adaptive informative hyperparameters returns better results than the BlockVB algorithm with separate initialisation of the start values in combination with both the Blockloading algorithm and the Blockloading++ algorithm.
The dramatic increase in computational speed of our Blockloading++ algorithm compared to our Blockloading algorithm shows that our analysis, discussion and efforts in sections 8.2 to 8.7 were successful.
An additional advantage of Blockloading++ is that the max-size cluster selection method has become viable for complicated real world networks contrary to the Blockloading algorithm. In fact, the max-size cluster selection method in combination with Blockloading++ returned even better results than maximum probability method. This finding is in contrast to Blockloading algorithm where the opposite holds.
These tests show that it is justified to choose the Bl++ Bay MS method as our new reference algorithm for further tests and comparisons with the batch algorithm and other methods.

10.7.2 Comparison of Blockloading++ and Batch Algorithm for the Poisson SBM

We stated above that the batch algorithm (algorithm 6) can be considered the state of the art algorithm for the application of V(B)EM inference to the SBM until ca. the year 2013. We initialised the VBEM batch algorithm (see chapter 4 and algorithm 3) for 1 to 60 clusters for the SCEN. We gave the specifications of the algorithm in section 10.3. The batch algorithm is initialised with a fixed number of clusters contrary to the Blockloading algorithms. We present the returned values of the converged free energy per fixed cluster of the batch algorithm in figure 10.13.
The best result returned by our VBEM batch algorithm was a converged free energy of $F^{(best,batch)} = 186735$ and $K = 47$ clusters. The run time of all 30 initialisations per cluster was a total of 1083 minutes or 18 hours and 5 minutes. The run time of our reference Blockloading++ was only 2 minute and 38 seconds for the best run on the

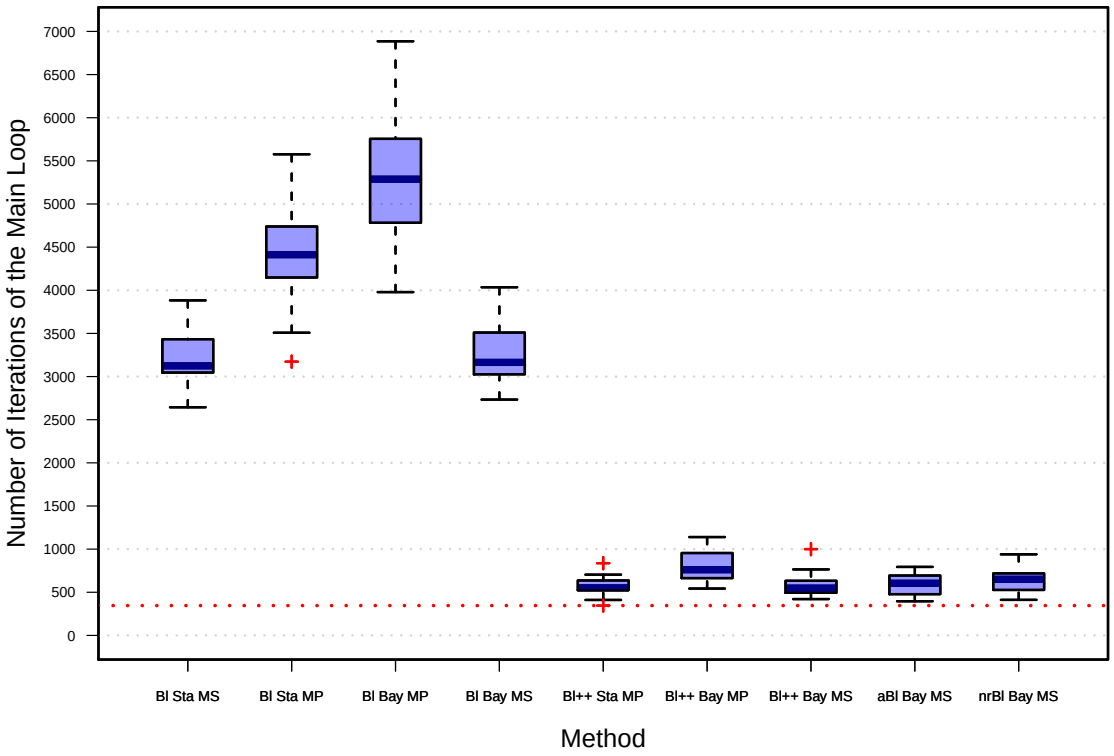

Figure 10.7: Boxplot of the performed iterations of the main loop until convergence of our Blockloading and Blockloading++ algorithms for the SCEN. The full names of all abbreviations are given in tables 10.1 and 10.2 in section 10.3. We see that there is clear gap of the required number of iterations of the main loop until convergence between the Blockloading algorithm and its more advanced variants. The minimum number of iterations is marked with the red line and was achieved by the Bl++ Sta MP method. The highest number of iterations was performed by the Bl Bay MP algorithm.

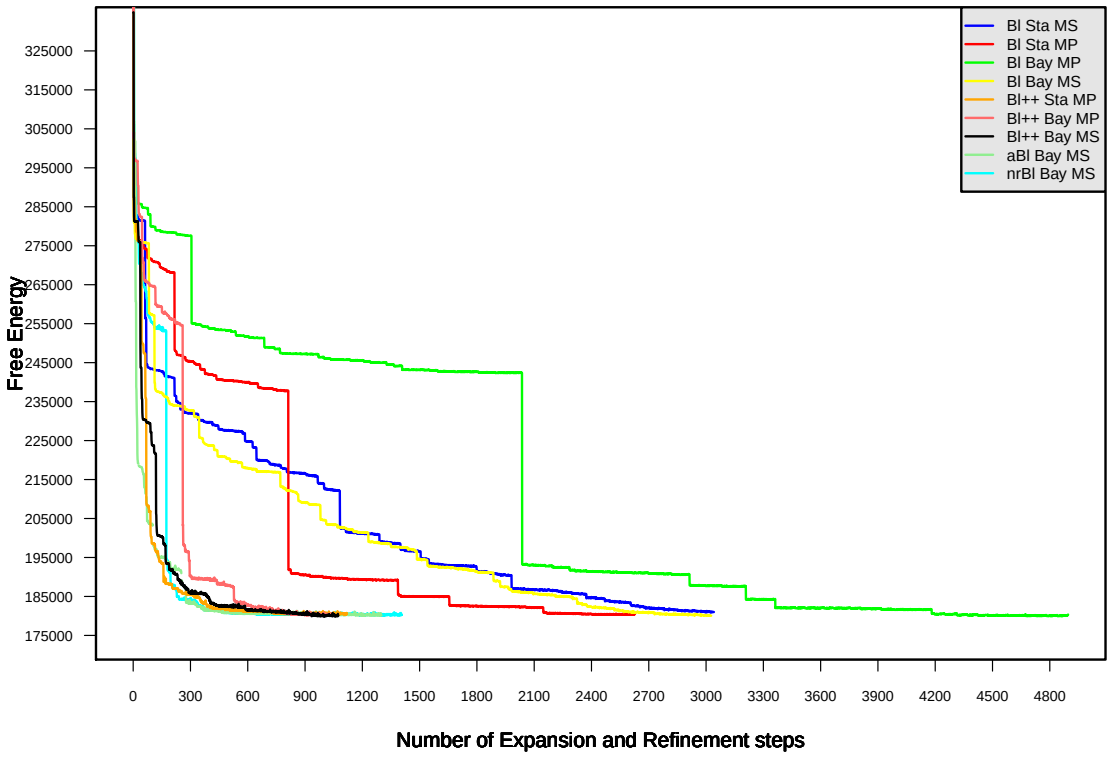

Figure 10.8: Plot of the speed of convergence measured by the total number of Expansion and Refinement steps of the best result, measured by the free energy criterion, of all test runs and all tested methods for the SCEN. The full names of all abbreviations are given in tables 10.1 and 10.2 in section 10.3. It can be seen that methods based on the automatic Blockloading (aBL), no-reset Blockloading (nrBl) and Blockloading++ algorithm (Bl++) converge much more rapidly than methods based on the Blockloading algorithm (Bl).

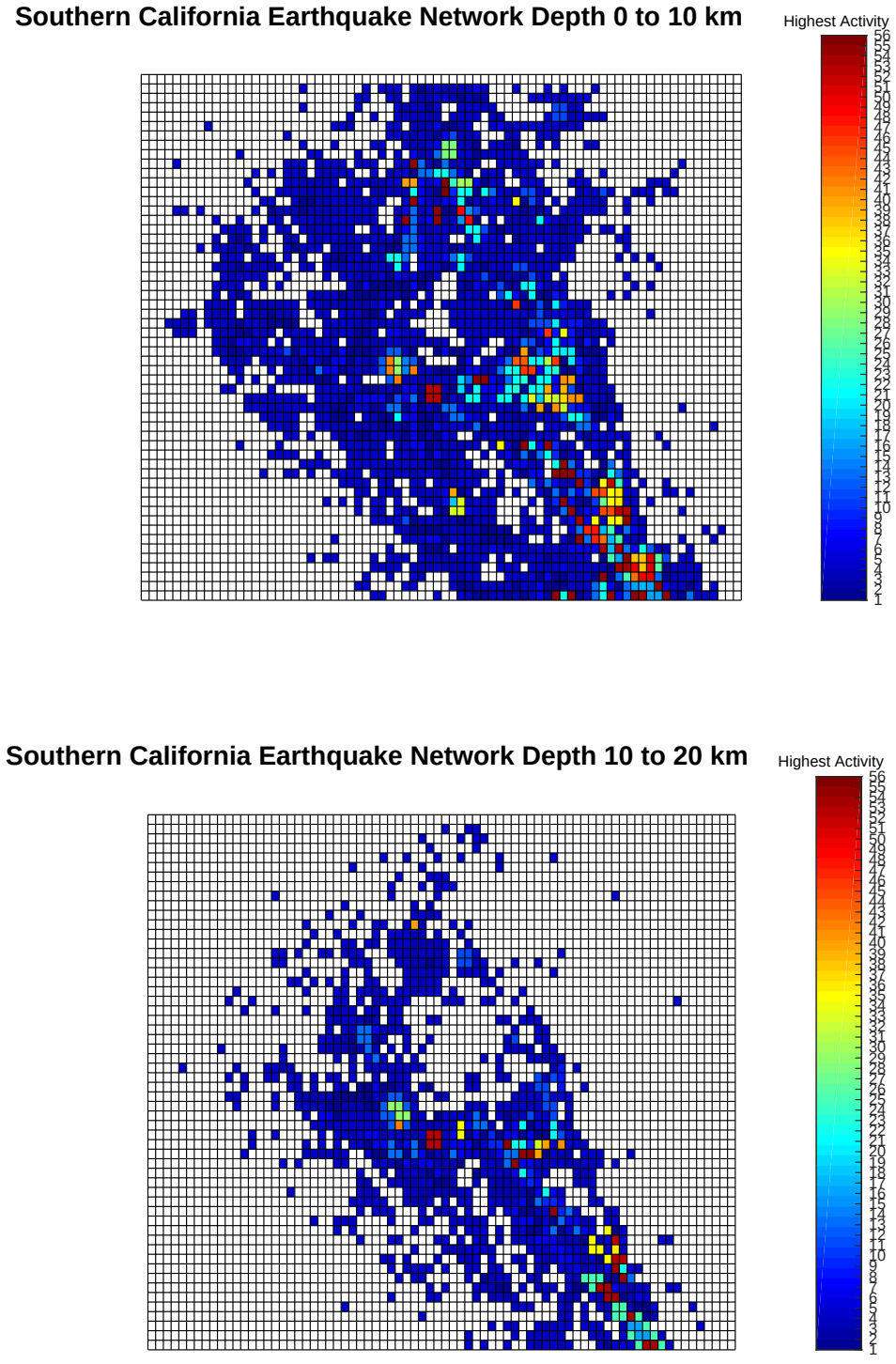

Figure 10.9: Color grid of the best result of the Clustering with the Blockloading++ BlockVB++ Maximum size algorithm of the Southern California Earthquake Network 1984 to 2015, see section 10.6. Geographic regions and depth levels with the same cluster memberships are marked in the same colour. Color codes chosen dependent on the connection intensity. Several hub clusters of the network, found by our Blockloading++ method, are coloured in dark red in the map.

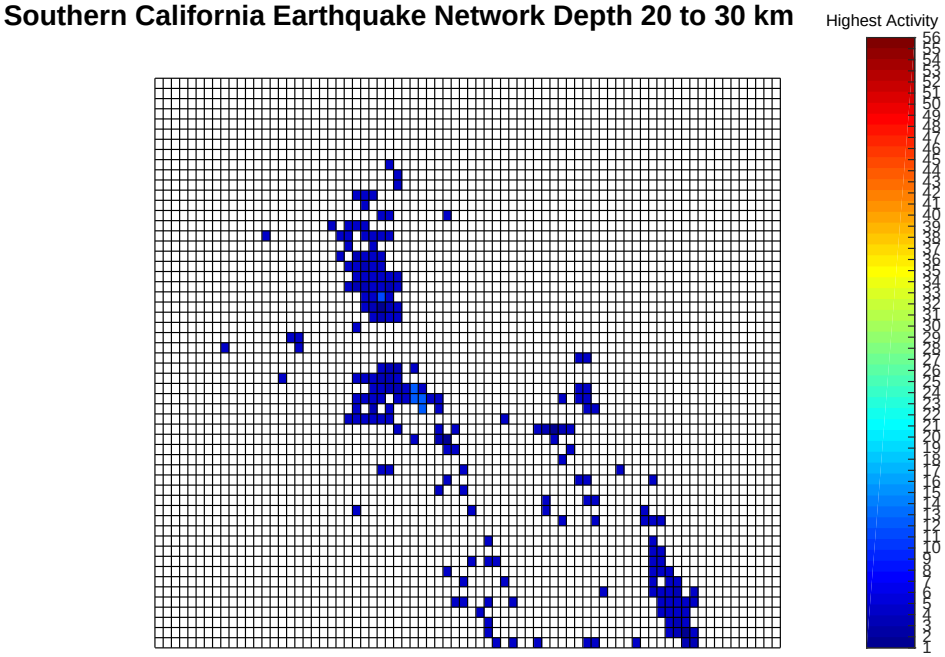

Figure 10.10: Color Grid of clustering of best result of Southern California Earthquake Network for a depth of 20 to 30 km. See description of figure 10.9 for explanation.

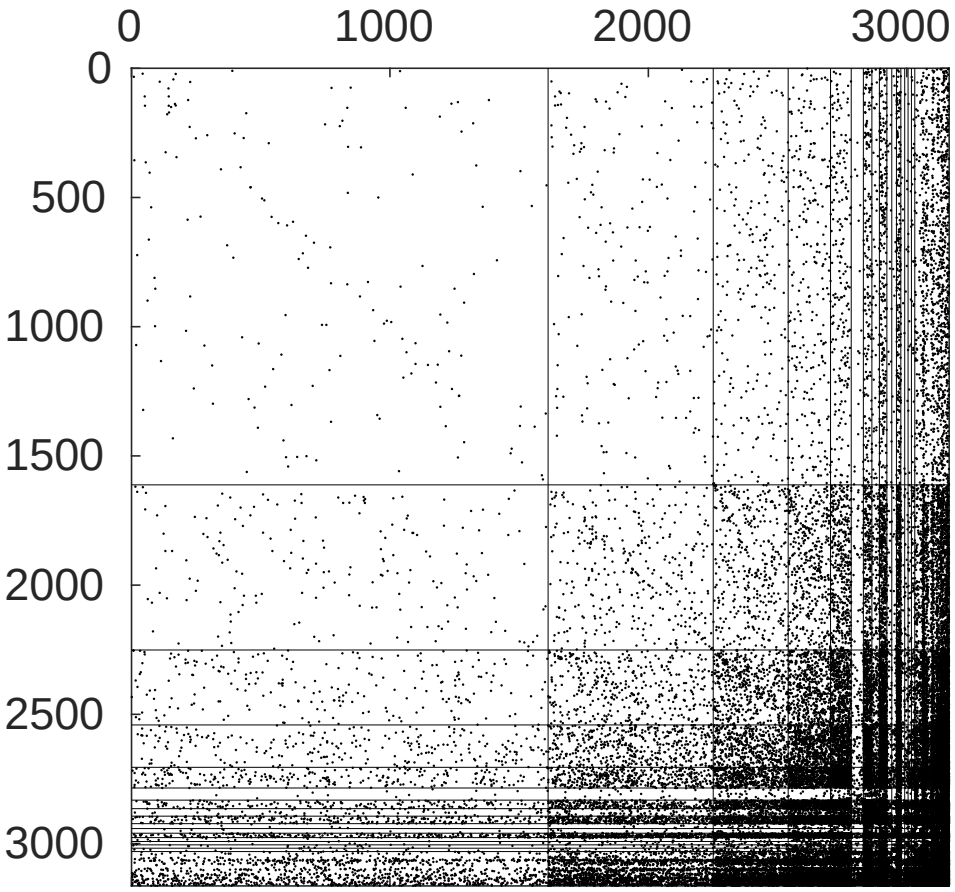

Figure 10.11: Dot–dot representation of the adjacency matrix of the Southern California Earthquake Network. The adjacency matrix, which includes 3163 vertices, is ordered according to the number of vertices per cluster. The clustering is the best result of the Blockloading++ (Bl++ Bay MS) algorithm for the Poisson SBM. The first 16 of the 56 clusters are separated with black lines.

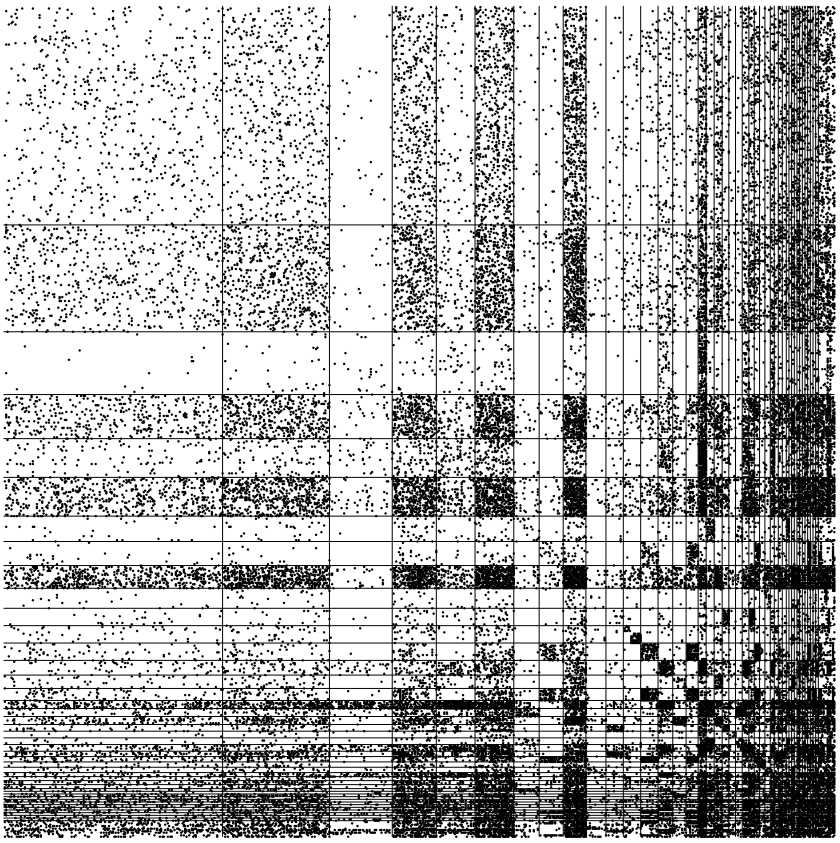

Figure 10.12: Dot–dot representation of an excerpt without the biggest three clusters of the adjacency matrix of the Southern California Earthquake Network. The clustering is the best result of the Blockloading++ (Bl++ Bay MS) algorithm for the Poisson SBM. The adjacency matrix, which includes 622 vertices, is ordered according to the number of vertices in each cluster in descending order. The first 43 clusters shown are separated with black lines.

	VBEM batch	Bl++ Bay MS
free energy criterion	186735	179925
$\Delta F^{(ref)}$	6811	0
Run Time	**18 h 5 min (1083 min)**	**2 min 38 sec**
NMI w.r.t. best partition	0.86	(1)
Optimal Number of clusters	47	56

Table 10.4: Comparison of the best results returned by our VBEM batch algorithm for the Poisson SBM and our Blockloading++ algorithm with BlockVB++ inference and maximum size cluster selection method (Bl++ Bay MS).

same machine. So, our reference Blockloading++ algorithm provides **a 433-fold speed increase**, when compared to the batch algorithm.
We remark that the worst result returned by our reference Blockloading++ algorithm with BlockVB++ inference and max-sizes cluster selection method, was a converged free energy of $F = 181037$ and the best result returned by a method based on the Blockloading++ algorithm a converged free energy $F^{(ref)} = 179924$ and $K^{(ref)} = 56$. Therefore we have a difference of the best converged free energy of the batch and the Blockloading++ algorithm of, $\Delta F^{(ref)} = F^{(best,batch)} - F^{(ref)} = 6811$. The worst result of all Blockloading algorithm was a converged free energy of $F = 183932$ which is still clearly better than the best result returned by the batch algorithm. We sum up the different result in table 10.4.

10.7.3 Comparison with the Blockmodels and WSBM package

We use the Southern California Earthquake Network (SCEN), we introduced in section 10.6, as an example to compare our Blockloading++ algorithm with our BlockVB++ inference and maximum size cluster selection method (Bl++ Bay MS) with the VEM based split-merge inference implementation of the Poisson SBM which is provided by the Blockmodels package [77, 79, 78].
We reviewed the Blockmodels package in section 7.4.2.
We also tested the WSBM 1.2 package [7] which implements the algorithms of [8]. The WSBM 1.2 package is implemented in Matlab and its computationally demanding parts in mex-C++. We initialised the WSBM 1.2 package, which implements a batch VBEM algorithm for the Poisson distribution with $K = 55$ clusters. After more than 48 hours run time and 18 iterations, it failed to converge and we cancelled the test.
The Blockmodels package is a state of the art split-merge inference software based on the Variational EM algorithm (section 3.3) and was published in 2015. We used the default settings for inference with the Blockmodels package. This includes a number of clusters which has no preset limit of the number of clusters at the beginning of the inference process. For each number of clusters, the Blockmodels package uses 5 re-initialisations with different start values.
The Blockmodels package is implemented in the 'R' statistics environment and ANSI C++ for the time critical parts. We ran all tests in this subsection on the same machine, an Intel Core i7 processor with 4 physical cores and 2.6 GHz. The Blockmodels software package provides parallelised inference. The default settings of the Blockmodels package include the use of 8 cores (threads).
On the contrary, our implementation of our Blockloading++ algorithm is not explic-

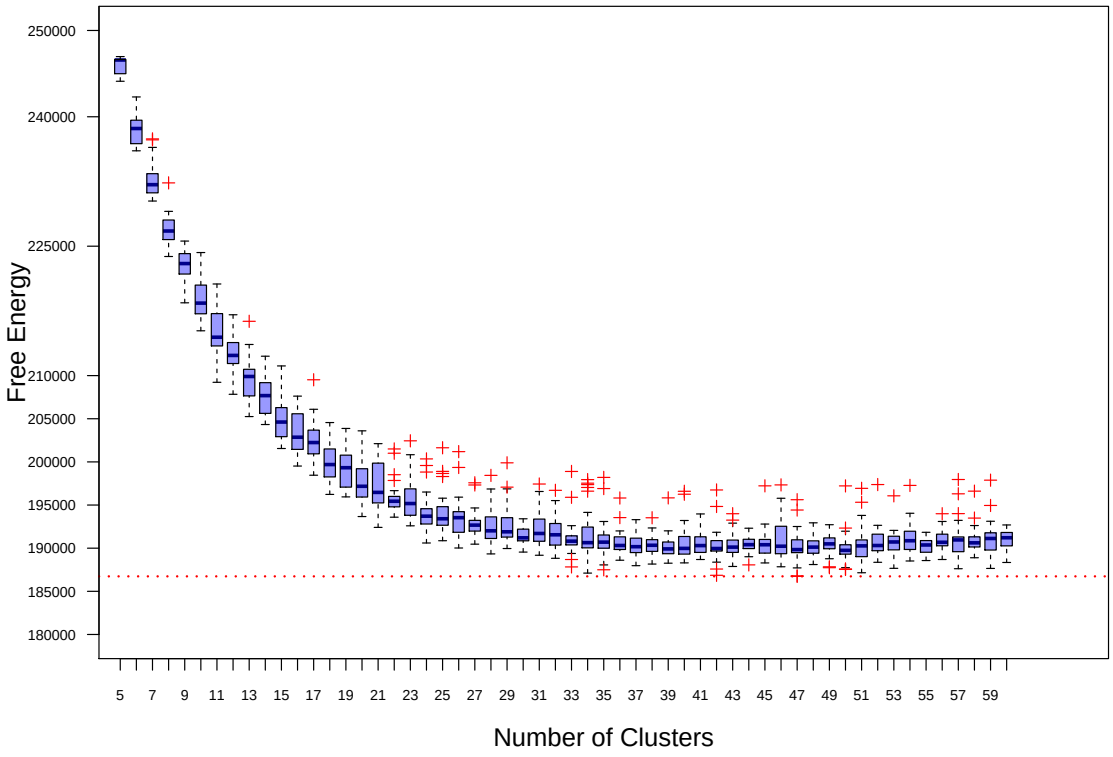

Figure 10.13: Box plot of the converged free energy per cluster returned by the Batch VBEM algorithm for the Bayesian Poisson SBM of the directed Southern California Earthquake Network with edge weights. We performed 30 initialisations per number of clusters. For specifications of the VBEM batch algorithm we refer to section 10.3. Results for 5 to 60 clusters are shown. Tests for 1 to 4 clusters are not shown here and yielded a higher (worse) converged free energy than the results presented in the figure. The best converged free energy of $F = 186735$ was returned for $K = 47$ clusters and is marked with the red line.

itly parallelised. Parallelisation of our Blockloading++ algorithm would be possible, though. We implemented our Blockloading++ algorithm in Matlab and C.
We recall from section 7.4.2 that the Blockmodels package optimises the ICL model selection criterion for the Poisson SBM during its split-merge inference process. We reviewed the ICL criterion for the Poisson SBM (Poisson ICL) in section 6.4.
The Blockmodels package provides different possibilities for setting the maximum number of clusters. We used the default setting where no prior limit on the potential number of clusters is set. Therefore, Blockmodels explored possible numbers of clusters from $K = 1$ to $K = 50$ in search of an optimum of the ICL criterion. For each number of clusters, except for $K = 1$, Blockmodels used 5 re-initialisations with different start values. The best result returned by our tests with the Blockmodels package had an optimal number of $K^{(BM)} = 33$ with an ICL value of $ICL^{(BM)} = -300916$. We recall that a higher value of the ICL criterion signals the better model according to the ICL criterion.
The computational time of the Blockmodels test until convergence was 2 days and 10 hours (3480 min).
A direct comparison with the best result of our Blockloading++ algorithm in section 10.7 is not viable because we did not optimise the Poisson ICL criterion but the free energy of the Poisson SBM during the inference process. We discussed the differences between the approaches of the free energy and the ICL criterion for the SBM in chapter 6. So, we saw no use in calculating the free energy of the best result of the Blockmodels package or the Poisson ICL criterion of the best result of our Blockloading++ algorithm.
In order to allow a viable comparison, we implemented the Poisson ICL criterion for our Blockloading++ algorithm (Bl++ Bay MS version) and ran a new test of our Blockloading++ algorithm where the Poisson ICL criterion was optimised instead of the Poisson free energy (see also section 8.1).
We noticed during cross validation that there were slight differences in the outcome of the ICL calculation for the same cluster partition with the Blockmodels package and our implementation. Therefore, we calculated all values of the ICL criterion shown here with the Blockmodels package. This includes the cluster assignment $\boldsymbol{Q}^{(BL++)}$ which was returned by our Blockloading++ algorithm optimising the ICL criterion.
This test run of our Blockloading++ algorithm took a computational time of 2 minutes and 16 seconds. Our Blockoading++ algorithm returned a value of the ICL criterion of $ICL^{(BL++)} = -295748$, which is better than the best result returned by the Blockmodels package. This amounts to a difference of $\Delta ICL^{(ref)} = 5168$ between the two results of the two tested algorithms.
Our Blockloading++ algorithm returned an optimal number of clusters according to the ICL criterion of $K^{(BL++)} = 37$ clusters.
There is a striking difference between the run time of the Blockmodels package and our Blockloading++ algorithm. Even if we count the use of 5 re-initialisations of the Blockmodels package per number of clusters with different start values as five different tests, our Blockloading++ algorithm is still definitely faster.
The maximum number of clusters which was searched for an optimum by the Blockmodels package was $K = 50$ clusters. This is much higher than the optimal number of clusters $K = 33$ finally found by the Blockmodels package. The maximum number of clusters which is searched by the Blockmodels package depends on the exploration factor, which has to be provided by the user. We used the default value of 1.5. This means that up to 1.5 times of the highest number of current clusters is searched during the inference process. On the contrary, our Blockloading++ algorithm avoided higher

	Blockmodels [79]	Blockloading++ (Bl++ Bay MS)
ICL criterion (Blockmodels)	-300916	**-295748**
$\Delta ICL^{(ref)}$	5168	(0)
Run Time	**2 days 10 h (3480 min)**	**2 min 16 sec**
NMI of both partitions	0.82	0.82
Optimal Number of clusters	33	37

Table 10.5: Comparison of the best results returned by the Blockmodels package [79] for the Poisson SBM and our Blockloading++ algorithm with BlockVB++ inference and maximum size cluster selection method (Bl++ Bay MS) which was optimised for the ICL criterion of the Poisson SBM (section 6.4. Higher values of the ICL criterion are better. Values in brackets follow by definition. The ICL criterion was calculated with the Blockmodels [79] package for both results.

number of clusters which are far away from the eventual optimal result of $K^{(BL++)} = 37$ clusters.
In addition, we recall from section 10.7, that our Blockloading++ provided an inference of more than $K = 50$ clusters in search of the optimal converged free energy for the same network in under 10 minutes.
The long run time of the Blockmodels package for this test is the reason why we performed only one test run.
This test shows, that our Blockloading++ algorithm for the Poisson SBM clearly outperformed the Blockmodels package in terms of computational speed and quality of the results. We sum up the results of both tests in table 10.5.

10.7.4 Comparison of the Free Energy and ICL Criterion for the Southern California Earthquake Network

We discussed the difference of the objectives of the converged free energy and the ICL model selection criterion in chapter 6. Now, we compare the optimal cluster assignments which were returned by our Blockloading++ algorithm when optimising the ICL criterion, $\boldsymbol{Q}^{(ref,icl)}$, and the free energy criterion, $\boldsymbol{Q}^{(ref,free)}$, for the Southern California earthquake network (SCEN) with the Normalised Mutual Information (NMI) criterion.
We recall from section 10.7, that the best result according to the converged free energy had $K = 56$ clusters and a converged free energy of $F^{(ref)} = 179924$. We sum up the results in table 10.6. Despite the clear difference in the number of clusters according to the ICL criterion or the free energy, the shared information by the two cluster partitions $\boldsymbol{Q}^{(ref,free)}$ according to the free energy criterion and $\boldsymbol{Q}^{(ref,icl)}$ has a rather high NMI of 0.91.
Nevertheless, the ICL seems to miss some clusters which was also observed in [73] in numerical tests of the ICL criterion for the Bernoulli SBM.

10.8 Link Prediction and Weight Correlation of the Southern California Earthquake Network

Link prediction is used to judge how well the clustering result of a network or a network model fits to the data and can also be used to compare different methods

	Blockloading++ with free energy	Blockloading++ with ICL
ICL criterion (Blockmodels)	-310062	**-295748**
$\Delta ICL^{(ref)}$	5168	(0)
converged free energy	**179924**	185630
$\Delta F^{(ref)}$	(0)	5706
Run Time	2 min 38 sec	**2 min 16 sec**
NMI of both partitions	0.91	0.91
Optimal Number of clusters	56	37

Table 10.6: Comparison of the best results according to the converged free energy and the ICL criterion optimised by our Blockloading++ algorithm with BlockVB++ inference and maximum-size cluster selection method for the Poisson SBM (Bl++ Bay MS) of the Southern California Earthquake Network (SCEN). Higher values of the ICL criterion and lower values of the converged free energy are better. Values in brackets follow by definition. The ICL criterion was calculated with the Blockmodels [79] package for both results.

[92, 119, 105, 28]. We apply link prediction and weight correlation to the case of the Southern California Earthquake Network (SCEN), according to the Poisson SBM. As we saw above, the SCEN was difficult to cluster reliably due to its complexity and size. We follow the approach of link prediction in [92] which was implemented in [91].
The main idea of evaluation of this model based cluster assignments through link prediction is to delete a given percentage of the edges of the network uniform at random [91]. The same percentage number of non-links or non existing edges is also treated as missing [92]. Then the estimated model is used to predict the probabilities for the existence or non-existence to the previously deleted edges [92]. Based on the probabilities returned by the estimated model, the Area Under Curve (AUC) of the Receiver Operating Characteristic (ROC) is calculated to evaluate the result. An introduction to the AUC criterion is provided in [37].

Our Tests We delete edges from the adjacency matrix $\boldsymbol{A}$ uniformly at random, with the help of software provided by [91], to produce a test set of edges for link prediction. These deleted links and the non-edges treated as missing are also called 'held-out' links or edges [105]. We choose 2.5 percent to 90 percent of the edges and non-edges of the network as the test set in order to produce different test matrices.
The result is a test adjacency matrix, $\boldsymbol{A}^{(test)}$, of the network with a differing percentage of deleted edges for each test. We produce two types of test adjacency matrices: For one adjacency matrix, $\boldsymbol{A}^{(test,w)}$, we keep the edge weights and for the other matrix, $\boldsymbol{A}^{(test,unw)}$, we replace them with unweighted edges. Then we run our Blockloading++ algorithm with max-size-cluster selection method and BlockVB++ algorithm (Bl++ MS Bay) for both adjacency matrices, $\boldsymbol{A}^{(test,w)}$ and $\boldsymbol{A}^{(test,unw)}$. We run five tests per percentage number of deleted edges. For each test, we generate a new set of held-out edges.
The inference of the Poisson SBM returns the cluster partition matrices, $\boldsymbol{Q}$, and the model parameters, $(\boldsymbol{\alpha}, \boldsymbol{\beta})$, for the Gamma distribution on the rate parameters, $\boldsymbol{\lambda}$, of the Poisson SBM.
In the case of the unweighted graph, we calculate the expectation value of the rate

parameters, $\mathbb{E}[\lambda_{kl}] = \frac{\alpha_{kl}}{\beta_{kl}} \forall k,l$, as an estimator for the probabilities of the edge existence of the held out links. Then, we calculate the AUC with the software [91]. We present the results, coloured in blue, of the AUC in figure 10.14. There we see that for all percentage numbers of deleted links, the AUC is above 0.93 on average, where the best possible value of the AUC is 1.0.
In the case of the weighted network, we first estimate the cluster assignments, $\boldsymbol{Q}^w$, according to the Poisson SBM with our Blockloading++ algorithm. Then we remove the edge weights of the adjacency matrix and recalculate the parameters for $(\boldsymbol{\alpha}^w, \boldsymbol{\beta}^w)$ dependent on $\boldsymbol{Q}^w$ with the help Proposition 2 in section 4.1. Then we calculate the estimator for the probabilities according to $\mathbb{E}[\lambda_{kl}^w] = \frac{\alpha_{kl}^w}{\beta_{kl}^w}$. We use these probabilities to calculate the AUC with the help of [91]. The results are shown in green colour in figure 10.14. We see that up to 80 percent removed edges, the AUC is higher than 0.9 and for 90 percent removed higher than 0.89 on average for five tests.

Weight Correlation In the case of the weighted networks, we also calculate the correlation between the estimated weights of the held-out edges according to the expectation values of the rate parameters $\mathbb{E}[\lambda_{kl}]$ (see above) and the true weights of the network. We deleted weights and edges to generate the held-out data. This test was proposed in [119]. We remark that we generated the held-out weights with [91] in the same way as above. In [119] edges were deleted. We present the result for the SCEN in figure 10.15. There, we see that for up to 20 percent of the deleted weights, the correlation is higher than 0.71 on average for five tests.

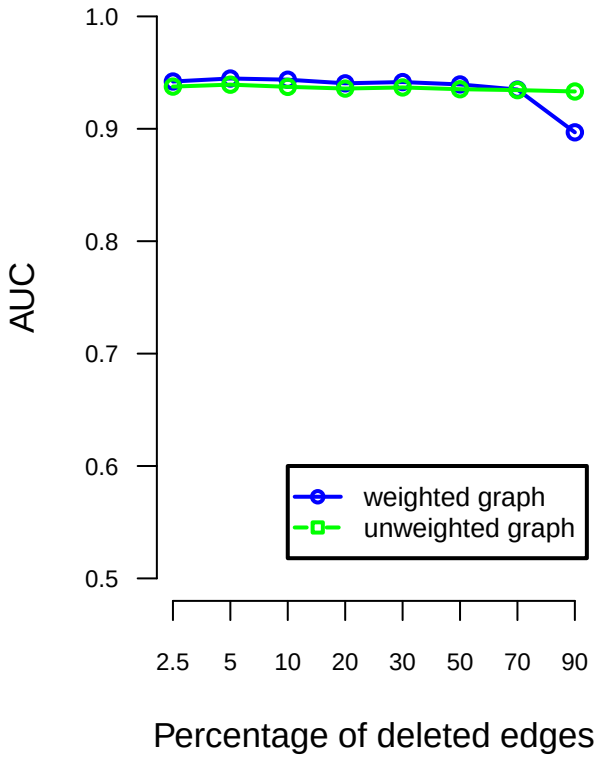

Figure 10.14: Value of the Area Under Curve (AUC) of the Receiver Operating Characteristic (ROC) of link prediction of the Southern California Earthquake Network. The highest possible value of the AUC is 1.0. We estimated the Poisson SBM with the Blockloading++ algorithm for the weighted (blue) and unweighted (green) adjacency matrix. The weights of the weighted graph were removed before calculating the estimators for the probabilities of edge existence of the weighted network. Average results of five tests for different percentage number of deleted edges is shown.

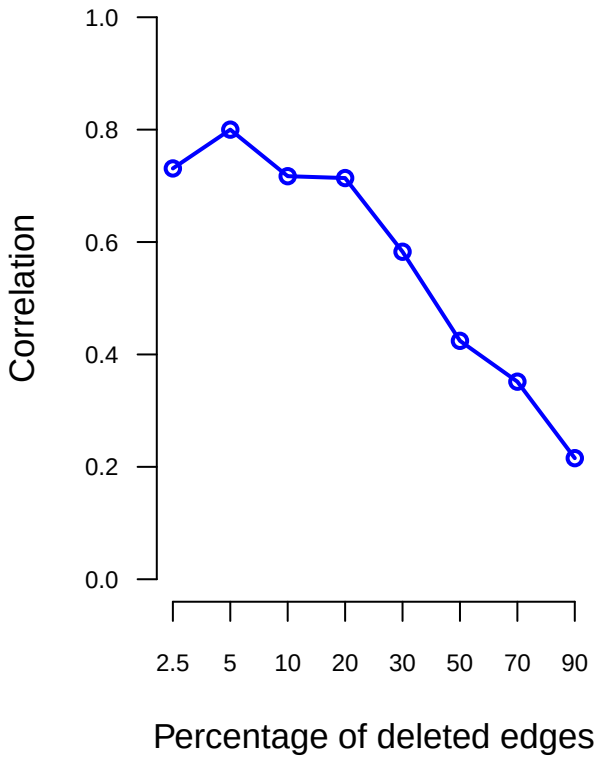

Figure 10.15: Weight Correlation of predicted weights according to the Poisson SBM for the weighted Southern California Earthquake Network. Estimation was performed five times with the Blockloading++ algorithm (Bl++ MS Bay) for different percentages of deleted edges of the true network.

Chapter 11

Introduction to the Stochastic Block Model with irrelevant Vertices and its VBEM Inference

Networks arise in different scientific areas like Protein–Protein interaction networks in Biology [87, 35] or actor based networks in Sociology [108, 97, 126]. The Stochastic Block Model (SBM) [97, 35, 84] is a well established and widely used model for the clustering of networks. In the SBM, the vertices of the network are grouped in clusters (or blocks) based on the edge connection profile of the vertices. The results of the SBM are easily interpretable [108, 97].

Often real world networks are given without a known ground truth of the cluster assignment of vertices. In this situation, the task is to infer the optimal hidden cluster assignment of vertices together with the optimal number of clusters and the optimal parameters of the model.

The analysis of the statistics of many real world networks shows, that most of the vertices are sparsely and irregularly connected to other vertices of the network [71]. If the network is weighted, e.g. the edges of the network have different weights, there can also be a huge variance of possible weights. Networks which exhibit such a connection behaviour usually have a heavy tails or scale free distribution of vertex degrees [71]. If these edge connection properties are present in a network for many vertices, the process of inferring a SBM for the network is difficult to impossible because the huge component of irregularly and sparsely connected vertices cannot be clustered with clear results according to a SBM. In the literature , these vertices are called *irrelevant* or *noisy* vertices [44, 54]. Moreover, the irrelevant vertices can disturb the inference process of the relevant vertices which can be clustered according to a SBM [116]. So, we would prefer to identify these irregular vertices before or during the inference process to avoid biased results.

All these observations also apply to our main application the earthquake networks introduced in [1]. Examples of the scale free distribution of the vertex degrees of earthquake networks as well as the highly varying intensity of edge connections were shown to hold for the earthquake networks which were analysed in [1, 6].

The normal SBM offers no dedicated mechanism to model the irrelevant vertices. The best we can hope for is to group these vertices in one cluster and keep them in this cluster during the optimisation. This requires an inference mechanism which locks these vertices into one cluster. An inference algorithm which has this property for the SBM was proposed in [116] and chapter 8 with our Blockloading algorithm. Nevertheless, the irrelevant vertices are not modelled explicitly by the SBM and the danger of over– or under fitting the number of clusters remains, which can lead to inferior results.

We propose the weighted Poisson Stochastic Block Model with Irrelevant Vertices (SBMIV) to address these limitations of the SBM. The SBMIV builds upon the Subset Infinite Relationals Model (SIRM) introduced by [54]. The SIRM is an extension of the Infinite Relational Model (IRM) introduced by [67] as a variant of the SBM with an unlimited number of clusters. The SIRM builds upon work presented in [47], [46] and [44], [54]. Vertices with irregular and sparse edge connections, which are hard to cluster according to the IRM, are considered as irrelevant in the SIRM, whereas vertices which could be clustered according to the IRM are considered as relevant. A hidden variable R_i is introduced for each vertex, with $R_i = 1$ if the vertex is relevant and $R_i = 0$ if it is irrelevant. The edge connections of the irrelevant vertices with all other vertices of the network are generated with the same parameter which is distributed according to a Beta prior distribution. The SIRM is a model for networks with undirected and unweighted (simple) edges. The edge connection probabilities of the SIRM are modelled according to a Bernoulli distribution like for the Bernoulli SBM in chapter 2. Contrary to the SBM, a Chinese Restaurant (CRP) Prior [20] is set for the proportions of the number of vertices in the clusters of the IRM. It was noted in [54], that the use of the CRP prior for the proportions of clusters sizes in the IRM favours the emergence of minute clusters and can also lead to biased results of the cluster assignments when irrelevant vertices are present in the network. Contrary to the IRM, non–informative priors are set on the size proportions in the Bayesian variant of the SBM [73, 116] which does not lead to such a bias.

For inference of the SIRM, a Gibbs Sampling algorithm was proposed by [54], which samples the cluster and relevance assignments of the vertices together.

We adopt an extension of our Blockloading algorithm for inference of the SBMIV which we call *Relevance Blockloading*. The Blockloading algorithm builds on an adopted Variational Bayesian Expectation Maximization (VBEM) algorithm and it was shown in [116] that it outperforms Spectral Clustering of [107], collapsed Gibbs Sampling of [85] and greedy algorithms [29]. In addition, we provided tests our Blockloading algorithms for an earthquake network in chapter 10, where the outperformance of the blockmodels package [79, 78], a state of the art implementation of a Variational EM split-merge inference algorithm for the SBM, was shown.

We propose an algorithmic framework which allows the use of the Integrated Likelihood Variational Bayes (ILVB) or free energy criterion for the SBM of [50, 73], which we discussed in section 6.2 in chapter 6 for the Bayesian Poisson SBM, as a model selection criterion for the optimal cluster assignment of the vertices and number of clusters of the SBMIV. Our Relevance Blockloading algorithm offers a fully Bayesian inference process based on the use of our relevance informative hyperparameters, which is independent of other algorithms for finding a start cluster assignment of the vertices. The Relevance Blockloading algorithm uses our Relevance BlockVB algorithm which provides VBEM inference for the SBMIV building upon our BlockVB algorithm of section 4.2. We propose an original way for the choice of informative priors on the relevance assignment of the vertices, which allows us to calculate the relevance of vertices in the first iteration of the algorithm. This procedure for finding

the relevant vertices as a first step is only dependent on the choice of the informative hyperparameters and will yield the same result when initialised with the same informative priors. So, for this filtering of vertices restarts with different initial relevance assignments are unnecessary. An approach for the selection of relevant features with informative priors in a Variational Bayesian framework was also proposed in [44].
In the literature, it is differentiated between three main algorithmic frameworks for determining the relevant vertices [44]: There is the filtering approach where the inference of the relevance of the vertices is calculated in a separate step from the assignment of relevant vertices to the clusters. Second there are embedded algorithms where the relevance classification and the cluster assignment are combined in one step. The subset clustering methods of [46, 44, 54] are examples of such an embedded method. Lastly, there are wrapper methods which are feature selection algorithms which '[...] wrap feature search around the learning algorithms that will ultimately be applied, and utilize the learned results to select the features.' [44].
We will propose both a filtering and an embedded variant of our Relevance Blockloading algorithm and compare them in numerical tests. Both of our algorithmic variants allow for a step by step expansion of the number of relevant vertices in a network partition which is growing during the inference process. This algorithmic procedures makes the Relevance Blockloading algorithm more efficient than previous variational methods.
We developed the SBMIV and its Relevance Blockloading inference framework with special regard to the model based clustering of Earthquake Networks.
Earthquake Networks, which were introduced in [1], are an example of the aforementioned real world networks with no known ground truth but a large number of sparsely and irregular connected vertices which renders the reliable inference with a model based clustering approach difficult.

Chapter 12

The Stochastic Block Model with irrelevant Vertices

12.1 Review of the Poisson Stochastic Block Model

We shortly recall the main facts about the Poisson SBM [84] from our review in chapter 2 where also more background information of the Bayesian Poisson SBM is presented. A graph $G = (V,E)$ consists of a set V of N vertices or vertices and a set of (directed) edges E connecting the vertices. The edges connecting the vertices are given by an adjacency matrix $\boldsymbol{A}$. If there is an edge from vertex i to vertex j it is $A_{ij} = w$, where $w \in (0,1,2,\dots)$ is a discrete valued weight. If there is no edge from vertex i to vertex j, it is $A_{ij} = 0$. Here, we will consider directed and weighted graphs unless otherwise stated.
The following Stochastic Block Model (SBM) was introduced in [84] as an algorithm for generating graphs and builds on the simple edge version of the SBM of [108]. We assume that $\boldsymbol{A}$ was generated by the SBM. The SBM assigns the vertices V of the graph depending on their connection probability patterns to clusters. The SBM consists of K clusters. To each vertex i, the SBM assigns a unique cluster membership. A vertex belongs to cluster k with probability π_k, with $\sum_{k=1}^{K} \pi_k = 1$. The cluster membership is given by the random variable $\boldsymbol{Z}_i \in R^{1\times K}$, with $Z_{ik} = 1$ if i is an element of cluster k and $Z_{ik} = 0$ otherwise. $\boldsymbol{Z}$ is the $N \times K$ cluster indicator matrix with matrix rows $\boldsymbol{Z}_i$ for $i \in \{1,\dots,N\}$. An edge exists within each cluster k with a weight according to the rate λ_{kk} and between cluster k and l with the rate λ_{kl}. So, the weighted Poisson SBM is generated in the following way [50, 116]:
(i) Roll a k – sided dice with $p(i \in k|Z_{ik} = 1) = \pi_k$ for side k for each vertex i, to determine the unique cluster membership of the vertex.

(ii) Draw a realization from

$$f(\cdot;\lambda_{kl}) = \frac{\lambda_{kl}^{A_{ij}}}{A_{ij}!}\exp(-\lambda_{kl}), \tag{12.1}$$

for the edge A_{ij} from vertex i to vertex j, with $i \in k$ and $j \in l$.

Then, the joint probability distribution for directed graphs is:

$$p(\boldsymbol{A},\boldsymbol{Z},\boldsymbol{\pi},\boldsymbol{\lambda}) = \prod_{i\neq j}^{N}\prod_{k,l}^{K} f(A_{ij};\lambda_{kl})^{Z_{ik}Z_{jl}} \prod_{i=1}^{N}\prod_{k=1}^{K} \pi_k^{Z_{ik}}. \tag{12.2}$$

The results of the clustering are easily interpretable. Variants of the SBM for simple graphs exist [52, 108]. For example it is possible to replace the Poisson distribution in (12.2) with a Bernoulli distribution [108, 35]. Using the Poisson distribution also works for unweigthed graphs. In the following, we call this SBM the Poisson SBM contrary to the Bernoulli SBM of [108, 35].

12.2 The Stochastic Block Model with Irrelevant Vertices

We propose the weighted Stochastic Block Model with irrelevant vertices (SBMIV). We build on the Subset Infinite Relational Model (SIRM) proposed in [54] for networks with simple (undirected and unweighted) edges. Contrary to the SIRM, we use the Poisson distribution to generate weighted and directed edges in the SBMIV like for the Poisson SBM of [84].
We consider a network with N vertices. Following [54], each vertex i is considered relevant with the probability of $\phi_i \in [0,1]$. Thus, the relevance, $\boldsymbol{R}$, of the vertices is generated according to

$$p(\boldsymbol{R}|\boldsymbol{\phi}) = \prod_{i=1}^{N} \phi_i^{R_i}(1-\phi_i)^{(1-R_i)}. \tag{12.3}$$

If $R_i = 1$, the cluster membership Z_i of vertex i is determined acceding to

$$p(\boldsymbol{Z}|\boldsymbol{\pi},\boldsymbol{R}) = \prod_{i=1}^{N}\left(\prod_{k=1}^{K} \pi_k^{R_i Z_{ik}}\right). \tag{12.4}$$

Otherwise the vertex is not considered as cluster able and there is no cluster assignment for that vertex. The cluster membership is a $\boldsymbol{Z}_i \in \mathbb{R}^{K\times 1}$ vector. For all pairs of vertices $(i,j) \in \{1,\ldots,N\}^2$, where $R_i = 1$ and $R_j = 1$ holds, we generate the edges between those vertices dependent on their cluster assignments, $\boldsymbol{Z}_i$ and $\boldsymbol{Z}_j$, according to

$$p(\boldsymbol{A}|\boldsymbol{Z},\boldsymbol{R},\boldsymbol{\lambda}) = \prod_{\substack{i,j\\ i\neq j}}^{N}\prod_{k,l}^{K}\left(\frac{\lambda_{kl}^{A_{ij}}}{A_{ij}!}\exp\left(-\lambda_{kl}\right)\right)^{Z_{ik}Z_{jl}R_iR_j}. \tag{12.5}$$

If one vertex is or both vertices of the vertex pair (i,j) are irrelevant, the edges connecting this pair of vertices are generated with the same rate γ. So, it follows that

$$p(\boldsymbol{A}|\boldsymbol{R},\gamma) = \prod_{\substack{i,j\\ i\neq j}}^{N}\left(\frac{\gamma^{A_{ij}}}{A_{ij}!}\exp(-\gamma)\right)^{\left(1-R_iR_j\right)}. \tag{12.6}$$

These considerations lead to the joint probability distribution of the SBMIV, for better readability we define $\boldsymbol{\vartheta} = (\boldsymbol{\lambda}, \boldsymbol{\pi})$:

$$p(\boldsymbol{A}, \boldsymbol{Z}, \boldsymbol{R}, \boldsymbol{\vartheta}, \boldsymbol{\phi}, \gamma) = \prod_{\substack{i,j \\ i \neq j}}^{N} \prod_{k,l=1}^{K} \left(\left(\frac{\lambda_{kl}^{A_{ij}}}{A_{ij}!} \right)^{R_i Z_{ik} R_j Z_{jl}} \exp\left(- R_i R_j Z_{ik} Z_{jl} \lambda_{kl} \right) \right) \prod_{i=1}^{N} \left(\prod_{k=1}^{K} \pi_k^{R_i Z_{ik}} \right)$$
$$\prod_{i \neq j}^{N} \left(\left(\frac{\gamma^{A_{ij}}}{A_{ij}!} \right)^{(1 - R_i R_j)} \exp\left(-(1 - R_i R_j)\gamma \right) \right) \prod_{i=1}^{N} \left(\phi_i^{R_i} (1 - \phi_i)^{(1 - R_i)} \right). \tag{12.7}$$

In the case of an undirected network, the product over all i, j is replaced with the product over $i < j$. We generate a network according to the SBMIV in the following way (cf. [50, 116]):

Generation of weighted SBM with irrelevant vertices **(i)** Flip a biased coin for each vertex $i \in \{1, \ldots, N\}$. With the probability ϕ_i the vertex is considered relevant, $R_i = 1$, and otherwise irrelevant, $R_i = 0$.

(ii) Roll a K–sided dice with $p(i \in k | Z_{ik} = 1, R_i = 1) = \pi_k$ for side k for each vertex i to determine the exclusive cluster assignment of the vertex.

(iii) Draw a realisation from $f(\cdot; \lambda_{kl})$ for the edge A_{ij} from vertex i to vertex j for all relevant vertices ($R_i = 1$ and $R_j = 1$) and cluster memberships $i \in k,\ j \in l$.

(iv) For all vertices i, j with $R_i = R_j = 0$, $R_i = 1$ and $R_j = 0$ or $R_i = 0$ and $R_j = 1$, draw realisation from $g(\cdot; \gamma) = \frac{\gamma^{A_{ij}}}{A_{ij}!} \exp(-\gamma)$ for the edge A_{ij}.

The increased flexibility of the SBM concerning the proportions of the cluster sizes compared to the IRM and SIRM of [54] also applies to SBMIV. In the next section, we will address the Bayesian view of the SBMIV.

12.3 The Bayesian SBMIV

To prepare the inference with Variational Bayesian EM methods, we state the Bayesian view of the SBMIV. The idea of the Bayesian treatment of the SBM is to set prior distributions for unknown parameters of the SBMIV, $\boldsymbol{\Theta} = (\boldsymbol{\lambda}, \boldsymbol{\pi}, \boldsymbol{\phi}, \gamma)$. Therefore, the model parameters are treated as random variables. For the simple edge Bernoulli SBM, this idea was used in [108, 73] with conjugate prior distributions. The SIRM is also a Bayesian model [54] which allows for the use of conjugate priors. Like in the SIRM, we place a Beta$(\phi_i; \zeta_i^0, \eta_i^0)$ prior distribution on the parameters ϕ_i which are conjugate to the Bernoulli distribution. Following [97, 50, 73], a Dirichlet Dir$(\boldsymbol{\pi}; \boldsymbol{\delta}^0)$ prior distribution, which is conjugate to the Multinomial distribution of the cluster assignments, is place on the parameter $\boldsymbol{\pi}$.
We place a Gamma$(\lambda_{kl}; \alpha_{kl}^0, \beta_{kl}^0)$ prior distribution, which is conjugate to the Poisson distribution, on the parameters λ_{kl}, like we did for the Poisson SBM in chapter 2 [96, 116]. We introduce a Gamma$(\gamma; \alpha_\gamma^0, \beta_\gamma^0)$ prior distribution over the rate of the

irrelevant edges, γ. We sum up the Bayesian treatment of the WSBMIV as:

$$\phi_i \sim \text{Beta}(\phi_i; \zeta_i^0, \eta_i^0) \equiv p(\phi_i), \tag{12.8}$$

$$\boldsymbol{\pi} \sim \text{Dir}(\boldsymbol{\pi}; \boldsymbol{\delta}^0) \equiv p(\boldsymbol{\pi}), \tag{12.9}$$

$$\gamma \sim \text{Gamma}(\gamma; \alpha_\gamma^0, \beta_\gamma^0) \equiv p(\gamma), \tag{12.10}$$

$$\lambda_{kl} \sim \text{Gamma}(\lambda_{kl}; \alpha_{kl}^0, \beta_{kl}^0) \equiv p(\lambda_{kl}). \tag{12.11}$$

The model generation of the Bayesian SBMIV is the same as in section 12.2, except that the model parameters have to be drawn from their respective prior distributions first, before generating the relevance assignment in **(i)**, the cluster assignment in **(ii)** and the edges in steps **(iii)** and **(iv)**.
Now, we can state the joint probability distribution of the Poisson SBMIV

$$\begin{aligned} p(\boldsymbol{A}, \boldsymbol{Z}, \boldsymbol{R}, \boldsymbol{\vartheta}, \boldsymbol{\phi}, \gamma | \, \boldsymbol{\delta}^0, \boldsymbol{\alpha}^0, \boldsymbol{\beta}^0, \boldsymbol{\zeta}^0, \boldsymbol{\eta}^0, \alpha_\gamma^0, \beta_\gamma^0) &= p(\boldsymbol{\pi}) p(\gamma) \prod_{i=1}^{N} p(\lambda_{kl}) \prod_{i=1}^{N} p(\phi_i) \\ &\times \prod_{\substack{i,j \\ i \neq j}}^{N} \prod_{k,l=1}^{K} \left(\left(\frac{\lambda_{kl}^{A_{ij}}}{A_{ij}!} \right)^{R_i Z_{ik} R_j Z_{jl}} \exp\left(- R_i R_j Z_{ik} Z_{jl} \lambda_{kl} \right) \right) \prod_{i=1}^{N} \left(\prod_{k=1}^{K} \pi_k^{R_i Z_{ik}} \right) \\ &\times \prod_{i \neq j}^{N} \left(\left(\frac{\gamma^{A_{ij}}}{A_{ij}!} \right)^{(1 - R_i R_j)} \exp\left(-(1 - R_i R_j) \gamma \right) \right) \prod_{i=1}^{N} \left(\phi_i^{R_i} (1 - \phi_i)^{(1 - R_i)} \right). \end{aligned} \tag{12.12}$$

The introduction of the prior distributions over the parameters of the Bayesian Poisson SBMIV allows us the application of our new adaptive informative hyperparameters we introduced in chapter 5 for the VBEM inference of the Bayesian Poisson SBM. In chapter 13 below, we will introduce our new relevance hyperparameters for the prior distributions of those model parameters of the SBMIV which affect the relevance of each vertex. There, we will see that our relevance hyperparameters together with our new algorithmic approach are essential for the inference of the relevance of the vertices.

Chapter 13

Inference of the SBMIV

13.1 Variational Bayesian EM Inference

In the previous sections (12.2, 12.3), we explained the generation of a network according to the Poisson SBMIV. Now, we treat the inverse problem of clustering a given network according to the SBMIV. For a network given by the adjacency matrix $\boldsymbol{A}$, we want to infer the latent variables $\boldsymbol{Z}$ and $\boldsymbol{R}$ and the unknown parameters $\boldsymbol{\Theta} = (\boldsymbol{\lambda}, \boldsymbol{\pi}, \gamma, \boldsymbol{\phi})$ of the SBMIV.
We use the Variational Bayesian Expectation Maximisation (VBEM) framework of [15, 19, 50, 71, 73] to optimise the latent variables and unknown parameters of the negative log-likelihood of the SBMIV, $-\ln p(\boldsymbol{A}|K)$. We reviewed the general VBEM framework in section 3.4 in chapter 3.
In the following, we will propose an original VBEM algorithm for the inference of our Poisson SBMIV. The aim of our VBEM algorithm is to approximate the intractable negative log-marginal-likelihood of the SBMIV,

$$-\ln p(\boldsymbol{A}|K) = \sum_{\boldsymbol{Z},\boldsymbol{R}} \int p(\boldsymbol{A}, \boldsymbol{\Theta}, \boldsymbol{Z}, \boldsymbol{R}) \mathrm{d}\boldsymbol{\Theta}, \tag{13.1}$$

with a tractable distribution $q(\cdot)$. Then, an upper variational bound of $-\ln p(\boldsymbol{A}|K)$, also called free energy [38, 50], dependent on the variational distribution $q(\boldsymbol{Z}, \boldsymbol{R}, \boldsymbol{\Theta})$ is derived with Jensen's inequality [38, 19, 50] .
To achieve a tractable variational distribution $q(\cdot)$ for the solution of the log–likelihood of the SBMIV, we use the mean–field assumption of [19, 50, 73] for the inference of our SBMIV in the following way:

$$q(\boldsymbol{Z}, \boldsymbol{R}, \boldsymbol{\Theta}) = q(\gamma) q(\boldsymbol{\pi}) \prod_{i=1}^{N} q(\phi_i) \prod_{k,l}^{K} q(\lambda_{kl}) \prod_{i=1}^{N} q(\boldsymbol{Z}_i) \prod_{i=1}^{N} q(\boldsymbol{R}_i). \tag{13.2}$$

We do not have to assume a functional form for the variational distributions $q(\cdot)$ but can infer the functional form of each distribution $q(\cdot)$ from the optimisation of the upper bound [19]. We calculate the variational bound of the negative log–likelihood, where

we omit K for the sake of brevity, in the following way:

$$-\ln\ p(\boldsymbol{A}|K) = -\ln \sum_{\boldsymbol{Z},\boldsymbol{R}} \int p(\boldsymbol{A},\boldsymbol{Z},\boldsymbol{R},\boldsymbol{\lambda},\boldsymbol{\pi},\boldsymbol{\phi},\gamma)\mathrm{d}\boldsymbol{\Theta} \tag{13.3}$$

$$= -\ln \sum_{\boldsymbol{Z},\boldsymbol{R}} \int \frac{p(\boldsymbol{A},\boldsymbol{Z},\boldsymbol{R},\boldsymbol{\Theta})}{q(\boldsymbol{Z})q(\boldsymbol{R})q(\boldsymbol{\Theta})} q(\boldsymbol{Z})q(\boldsymbol{R})q(\boldsymbol{\Theta})\mathrm{d}\boldsymbol{\Theta} \tag{13.4}$$

$$\leq -\sum_{\boldsymbol{Z},\boldsymbol{R}} \int \ln\left(\frac{p(\boldsymbol{A},\boldsymbol{Z},\boldsymbol{R},\boldsymbol{\Theta})}{q(\boldsymbol{Z},\boldsymbol{R},\boldsymbol{\Theta})}\right) q(\boldsymbol{Z},\boldsymbol{R},\boldsymbol{\Theta})\mathrm{d}\boldsymbol{\Theta} \tag{13.5}$$

$$\equiv F\left(q(\boldsymbol{Z},\boldsymbol{R},\boldsymbol{\Theta})\right). \tag{13.6}$$

We provide the free energy of the SBMIV after convergence (converged free energy) in Proposition 12 in appendix C.1 below. Now, we optimise the free energy (variational bound) dependent on the variational distributions $q(\cdot)$. The VBEM algorithm has an EM–like structure for the optimisation of the variational bound F with respect to the variational distribution $q(\cdot)$. The VBEM algorithm consists of two main steps: In the Expectation Step (E–step), the latent variables are optimised. In the Maximisation Step (M–step), the parameters $\boldsymbol{\Theta}$ are updated.
In the case of the SBMIV, we want to infer two different types of latent variables $\boldsymbol{R}$ and $\boldsymbol{Z}$. Each variable $\boldsymbol{Z}_i$ depends on the variable R_i in the SBMIV. The same situation applies to the SIRM of [54], where a Gibbs sampling approach was proposed in which the variables R_i and $\boldsymbol{Z}_i$ are sampled together [54].
We propose an original algorithm to solve the SBMIV with the VBEM framework. We start with the M–step and calculate the optimal update of $q(\boldsymbol{\Theta}) = q(\boldsymbol{\lambda},\boldsymbol{\pi},\boldsymbol{\phi},\gamma)$ at step t, where we keep $q^{(t)}(\boldsymbol{Z})$ and $q^{(t)}(\boldsymbol{R})$ fixed:

$$\{q^{(t+1)}(\boldsymbol{\Theta})\} = \arg\min_{\{q(\boldsymbol{\Theta})\}} F\left(q^{(t)}(\boldsymbol{Z}), q^{(t)}(\boldsymbol{R}), q^{(t)}(\boldsymbol{\Theta})\right). \tag{13.7}$$

We provide the update equations for the variational distributions of the parameters $q^{(t+1)}(\boldsymbol{\Theta})$ in the following Propositions. We begin with the optimisation of the free energy with respect to the distributions $q(\phi_i)\ \forall i$.

Proposition 6. *The optimisation of the variational bound $F[q(\boldsymbol{Z},\boldsymbol{R},\boldsymbol{\Theta})]$ with respect to $q(\phi_i)\ \forall i = 1,\ldots,N$ shows, that $q(\phi_i)$ has the functional form of a $\mathrm{Beta}(\phi_i;\zeta_i,\eta_i)$ distribution. It has the same functional form as the prior distribution $p(\phi_i) = \mathrm{Beta}(\phi_i;\zeta_i^0,\eta_i^0)$. The update equations of the hyperparameters $\boldsymbol{\zeta}$ and $\boldsymbol{\eta}$ are given by:*

$$\zeta_i = \rho_i + \zeta_i^0 \tag{13.8}$$

$$\eta_i = (1-\rho_i) + \eta_i^0, \tag{13.9}$$

where $\boldsymbol{\rho}$ is the relevance assignment.

Proof. See appendix C.1. □

Proposition 7. *The optimisation of the variational bound $F[q(\boldsymbol{Z},\boldsymbol{R},\boldsymbol{\Theta})]$ with respect to $q(\boldsymbol{\pi})$ shows, that $q(\boldsymbol{\pi})$ has the functional form of a Dirichlet $\mathrm{Dir}(\boldsymbol{\pi};\boldsymbol{\delta})$ distribution. It has the same functional form as the prior distribution $p(\boldsymbol{\pi}) = \mathrm{Dir}(\boldsymbol{\pi};\boldsymbol{\delta}^0)$. The update equations for the hyperparameters, $\delta_k\ k \in \{1,\ldots,K\}$ are given by:*

$$\delta_k = \sum_{i=1}^{N} \rho_i Q_{ik} + \delta_k^0, \tag{13.10}$$

where $\boldsymbol{Q}$ is the cluster partition matrix and $\boldsymbol{\rho}$ the relevance assignment.

Proof. See appendix C.1. □

Proposition 8. *The optimisation of the free energy (upper variational bound) $F[q(\boldsymbol{Z},\boldsymbol{R},\boldsymbol{\Theta})]$ with respect to $q(\gamma)$ shows, that $q(\gamma)$ has the functional form of a Gamma$(\gamma;\alpha_\gamma,\beta_\gamma)$ distribution. It has the same functional form as the prior distribution $p(\gamma)=$ Gamma$(\gamma;\alpha_\gamma^0,\beta_\gamma^0)$. The update equations of the hyperparameters α_γ and β_γ are given by:*

$$\alpha_\gamma = \sum_{\substack{i,j \\ i\neq j}}^{N}(1-\rho_i\rho_j)A_{ij}+\alpha_\gamma^0, \tag{13.11}$$

$$\beta_\gamma = \sum_{\substack{i,j \\ i\neq j}}^{N}(1-\rho_i\rho_j)+\beta_\gamma^0, \tag{13.12}$$

where $\boldsymbol{\rho}$ is the relevance assignment.

Proof. See appendix C.1. □

Proposition 9. *The optimisation of the free energy (upper variational bound) $F[q(\boldsymbol{Z},\boldsymbol{R},\boldsymbol{\Theta})]$ with respect to $q(\lambda_{kl})$ for all $(k,l)\in\{1,\dots,K\}^2$ shows, that $q(\lambda_{kl})$ has the functional form of a Gamma$(\lambda_{kl};\alpha_{kl},\beta_{kl})$ distribution. It has the same functional form as the prior distribution $p(\lambda_{kl})=$ Gamma$(\lambda_{kl};\alpha_{kl}^0,\beta_{kl}^0)$. The update equations of the hyperparameters α_{kl} and β_{kl}, $\forall(k,l)=\{1,\dots,K\}^2$, are given by:*

$$\alpha_{kl} = \sum_{i\neq j}^{N}\rho_i\rho_j Q_{ik}Q_{jl}A_{ij}+\alpha_{kl}^0, \tag{13.13}$$

$$\beta_{kl} = \sum_{i\neq j}^{N}\rho_i\rho_j Q_{ik}Q_{jl}+\beta_{kl}^0, \tag{13.14}$$

where $\boldsymbol{Q}$ is the cluster partition matrix and $\boldsymbol{\rho}$ the relevance assignment.

Proof. See appendix C.1. □

These Propositions show, that the variational distributions $q(\cdot)$ have the same functional form as the prior distributions $p(\cdot)$ of section 12.3 .
We continue the optimisation of the free energy with respect to the latent variables in the E–step. The optimisation of the free energy with respect to $q(\boldsymbol{Z}_i)$ in Propostion 10 shows, that $q(\boldsymbol{Z}_i)$ has the functional form of a Multinomial distribution for each vertex i.

Proposition 10. *The optimisation of the free energy (upper variational bound) with respect to $q(\boldsymbol{Z}_i);\forall i=1,\dots,N$, $\{q^\star(\boldsymbol{Z}_i)\}=\underset{\{q(\boldsymbol{Z}_i)\}}{\arg\min}\,F\left(q(\boldsymbol{Z}),q(\boldsymbol{R}),q(\boldsymbol{\Theta})\right)$, shows that $q^\star(\boldsymbol{Z}_i)$ has the functional form of a multinomial distribution:*

$$q^\star(\boldsymbol{Z}_i)=\mathscr{M}\left(Z_i;1,\boldsymbol{Q}_i=\{Q_{i1},\dots,Q_{iK}\}\right). \tag{13.15}$$

The update equation for $\mathbb{E}(Z_{ik})=Q_{ik}$, $\forall(i,k)\in\{1,\dots,N\}\times\{1,\dots,K\}$ is given by:

$$\begin{aligned} Q_{av}\propto\exp\Big(&\sum_{\substack{i=1\\ i\neq a}}^{N}\sum_{q,l}^{K}\rho_i\rho_a Q_{iq}A_{ai}\mathbb{E}(\ln\lambda_{vq})+\sum_{\substack{i=1\\ i\neq a}}^{N}\sum_{q,l}^{K}\rho_i\rho_a Q_{iq}A_{ia}\mathbb{E}(\ln\lambda_{qv})\\ &-\sum_{\substack{i=1\\ i\neq a}}^{N}\sum_{q=1}^{K}\rho_a\rho_i Q_{iq}\big(\mathbb{E}(\lambda_{vq})+\mathbb{E}(\lambda_{qv})\big)+\rho_a\mathbb{E}(\ln\pi_v)\Big), \end{aligned} \tag{13.16}$$

where $\mathbb{E}(\ln\lambda_{ql}) = \psi(\alpha_{ql}) - \ln(\beta_{ql})$, $\mathbb{E}(R_a) = \rho_a$, $\mathbb{E}(\lambda_{ql}) = \frac{\alpha_{ql}}{\beta_{ql}}$, $\mathbb{E}(\ln\pi_q) = \psi(\delta_q) - \psi\big(\sum\big(\sum_{l=1}^{K}\delta_l\big)\big)$ *and* $\psi(\cdot)$ *is the Digamma function.*

Proof. See appendix C.1. □

The cluster assignment is a *fuzzy*–update in eqn. 13.16, where a probability $Q_{ak} \in [0,1]$ is given for the cluster membership of vertex a in cluster k. We show in the next Proposition that the variational distribution $q(R_i)$ has the functional form of a Bernoulli $\mathrm{Ber}(R_i;\rho_i)$ distribution.

Proposition 11. *The optimisation of the free energy (upper variational bound) with respect to* $q(R_i); \forall i = 1,\ldots,N$, $\{q^\star(R_i)\} = \arg\min_{\{q(R_i)\}} F(q(\boldsymbol{Z}), q(\boldsymbol{R}), q(\boldsymbol{\Theta}))$, *shows that* $q^\star(R_i)$ *has the functional form of a Bernoulli distribution:*

$$q^\star(R_i) = \mathrm{Ber}(R_i;\rho i). \tag{13.17}$$

The update equation for $\mathbb{E}(R_i) = \rho_i, \forall i \in \{1,\ldots,N\}$ *is given by:*

$$\rho_a^\star = \frac{1}{1+\exp(-U_a)}, \tag{13.18}$$

with

$$\begin{aligned} U_a \equiv & \sum_{\substack{i=1\\ i\neq a}}^{N}\sum_{q,l}^{K}\rho_i Q_{iq}Q_{al}A_{ia}\mathbb{E}(\ln\lambda_{ql}) + \sum_{\substack{i=1\\ i\neq a}}^{N}\sum_{q,l}^{K}\rho_i Q_{il}Q_{aq}A_{ai}\mathbb{E}(\ln\lambda_{ql}) \\ & - \sum_{\substack{i=1\\ i\neq a}}^{N}\sum_{q,l}^{K}\rho_i Q_{iq}Q_{al}\mathbb{E}(\lambda_{ql}) - \sum_{\substack{i=1\\ i\neq a}}^{N}\sum_{q,l}^{K}\rho_i Q_{il}Q_{aq}\mathbb{E}(\lambda_{lq}) - \mathbb{E}(\ln\gamma)\sum_{\substack{i\neq a\\ i=1}}^{N}\rho_i(A_{ia}+A_{ai}) \\ & + 2\mathbb{E}(\gamma)\sum_{\substack{i=1\\ i\neq a}}^{N}\rho_i + \mathbb{E}(\ln\phi_a) - \mathbb{E}(\ln(1-\phi_a)) + \sum_{q=1}^{K}Q_{aq}\mathbb{E}(\ln\pi_q), \end{aligned} \tag{13.19}$$

where $\mathbb{E}(\log\lambda_{vk}) = \psi(\alpha_{vk}) - \log(\beta_{vk})$, $\mathbb{E}(\lambda_{vk}) = \frac{\alpha_{vk}}{\beta_{vk}}$, $\mathbb{E}(\ln\gamma) = \psi(\alpha_\gamma) - \ln(\beta_\gamma)$, $\mathbb{E}(Z_{ik}) = Q_{ik}$, $\mathbb{E}(\pi_q) = \psi(\delta_q) - \psi\big(\sum\big(\sum_{l=1}^{K}\delta_l\big)\big) = G_q$, $\mathbb{E}(\ln\phi_a) = \psi(\zeta_a) - \psi(\zeta_a+\eta_a)$, $\mathbb{E}(\ln(1-\phi_a)) = \psi(\eta_a) - \psi(\zeta_a+\eta_a)$ *and* $\psi(\cdot)$ *is the Digamma function.*

Proof. See appendix C.1. □

The update of the relevance assignment of vertex a, ρ_a in eqn. 13.18, is also a fuzzy–update like the update of the cluster assignment, $Q_{ak}; \forall k \in \{1,\ldots,K\}$, above, which gives us the expected value of the relevance assignment of vertex i, ρ_i. The fuzziness of $\boldsymbol{\rho}$ poses a problem for the update equation of the cluster assignment (eqn. 13.16), because it can lead to a bias.
We introduce the following rule to get a hard assignment of the relevance of vertex i dependent on $\rho_i^\star$: If $\rho_i \geq 0.5$ holds, we set $\rho_i = 1$ and otherwise we set $\rho_i = 0$. This rule is inspired by the Classification EM algorithm (CEM algorithm) of [124], where such a hard clustering is also used.
If we want to optimise the relevance assignment, R_i, and cluster assignment, $\boldsymbol{Z}_i$, of vertex i, we have to deal with two cases: The first case is, that i is relevant and therefore $\rho_i = 1$ holds. In the case of $\rho_i = 1$, we can use the update equation for the cluster assignment of relevant vertices eqn. 13.16 of Proposition 10 for the update of $\boldsymbol{Q}_i$ in a

straightforward way.
If on the other hand $\rho_i = 0$ holds, we have to be careful with the update of $\boldsymbol{Q}_i$. In this case, it follows that $Q_{ik} = \frac{1}{K}$, $\forall k \in \{1,\ldots,K\}$ which also gives biased results for the update of $\boldsymbol{Q}_a$ with $\forall a \neq i$. It also affects the update of ρ_i, which is given in eqn. 13.18 and 13.19 of Proposition 11. Moreover, from the perspective of the SBMIV (section 12.2), irrelevant vertices are not assigned to any cluster. This leads us to the conclusion that we should set $Q_{ik} \equiv 0$, $\forall k \in \{1,\ldots,K\}$.
If we set the cluster partition matrix entries of the irrelevant vertices to zero, we can see that the update of ρ_i only depends on the last for terms of eqn. 13.19. Thus, the update is dominated by the term $\mathbb{E}(\gamma)\sum_{i\neq a}^{N}\rho_i(A_{ia}+A_{ai})$. We found in numerical tests, that this leads automatically to the update $\rho_i = 1$, which is obviously wrong. So, we propose to calculate a cluster assignment $Q_{il};\forall l \in \{1,\ldots,K\}$ for the purpose of finding an unbiased relevance update first. There are different possibilities to assign the vertex i to a cluster in the case of $\rho_i = 0$. We could set $\rho_i = 1$ and calculate the updates of $Q_{il};\forall l \in \{1,\ldots,K\}$ according to eqn. 13.16 in Proposition 10. But with this approach, we would merge the irrelevant vertex i with relevant vertices in a relevant cluster. This irrelevant vertex and those relevant vertices were separated in previous iterations of the algorithm. Nevertheless this approach worked for all tests.
Our numerical tests showed us, that a better way is to limit the optimisation of the cluster assignment to the assignment to irrelevant status or to a newly introduced extra cluster. We provide the details in section 13.3.3. We conclude that this approach fits best to the aims of our SBMIV model, and it returned the best results in numerical tests. So, in all cases, we calculate or set a preliminary cluster assignment of the vertex which is currently optimised to get a relevance assignment. With this algorithmic approach, we can update ρ_i without the bias of missing terms because of $Q_{il} = 0;\forall l \in \{1,\ldots,K\}$ or biased cluster assignments because of $Q_{il} = \frac{1}{K};\forall l \in \{1,\ldots,K\}$. If the update yields $\rho_i^\star = 1$, we keep the updated cluster assignment $Q_{il}^\star$, on the other hand, if $\rho_i^\star = 0$ holds, we set $Q_{il}^\star = 0;\forall l \in \{1,\ldots,K\}$.
We conclude that we have to begin with the update of the cluster assignment $\boldsymbol{Q}_a$, with ρ_a set to one in all cases, and then we can proceed with the update of ρ_a dependent on the outcome of the update of $\boldsymbol{Q}_a$. After these two updates, we adjust $\boldsymbol{Q}_a$: If $\rho_a^\star = 1$ holds we keep the update of $\boldsymbol{Q}_a^\star$, if otherwise $\rho_a^\star = 0$ we set $Q_{al}^\star = 0;\forall l \in \{1,\ldots,K\}$. This way, the update of ρ_a is unbiased by missing terms of $\boldsymbol{Q}_a$ in the case of $\rho_a = 0$.
With these preparations we can state the E–step of the SBMIV, which consists of two parts: The optimisation with respect to the cluster assignment of vertex i

$$\{q^{(t+1)}(\boldsymbol{Z}_i)\} = \underset{\{q(\boldsymbol{Z}_i)\}}{\arg\min}\, F\left(q^{(t)}(\boldsymbol{Z}), q^{(t)}(\boldsymbol{R}), q^{(t+1)}(\boldsymbol{\Theta})\right), \tag{13.20}$$

and the relevance assignment of vertex i, given by

$$\{q^{(t+1)}(\boldsymbol{R}_i)\} = \underset{\{q(\boldsymbol{R}_i)\}}{\arg\min}\, F\left(q^{(t+1)}(\boldsymbol{Z}), q^{(t)}(\boldsymbol{R}), q^{(t+1)}(\boldsymbol{\Theta})\right). \tag{13.21}$$

Our VBEM algorithm now consists of the iterations of the update equations 13.7, 13.20 and 13.21 in the E– and M–step until the maximum number of iterations is reached or the free energy has converged,

$$F[q^{(t)}(\boldsymbol{Z}), q^{(t)}(\boldsymbol{R}), q^{(t)}(\boldsymbol{\Theta})] - F[q^{(t+1)}(\boldsymbol{Z}), q^{(t+1)}(\boldsymbol{R}), q^{(t+1)}(\boldsymbol{\vartheta})] < T, \tag{13.22}$$

where T is a predefined threshold.

We conclude that the VBEM inference algorithm for the SBMIV we presented above is an embedded algorithm, because the inference of the relevance and cluster assignment of each vertex is calculated together in the E–step ([44]).
The Propositions we proposed above together with our VBEM algorithm for the SBMIV are for a batch algorithm. We saw in the previous chapters of this thesis, that we should rather employ an inference approach which is restricted to subsets like our combination of our Blockloading algorithm of chapter 8 together with our BlockVB or BlockVB++ algorithms (see sections 4.2, 5.1).

13.1.1 Relevance BlockVB algorithm

We provide our **Relevance BlockVB algorithm** for the SBMIV which is a VBEM algorithm optimised for subsets in appendix C.2. Our Relevance BlockVB algorithm provides an embedded inference algorithm which we described in the last section. We call this version the embedded Relevance BlockVB algorithm.
Moreover, we also provide a the *filtering* Relevance BlockVB algorithm in appendix C.2. Our Blockloading framework from chapter 8 together with the filtering version of our Relevance BlockVB algorithm allows us to calculate the relevant vertices in a separate filtering step, where we skip the calculation of the assignment to the relevant clusters. Then, we continue the inference of the cluster assignment for the relevant vertices and skip the inference for the irrelevant vertices. Thus, the cluster assignments of the relevant vertices and the determination of the relevancy of the vertices are performed in separate steps. This is a filtering algorithm where the inference of relevance and cluster assignment of the vertices are separated [44]. We will describe both inference schemes in detail in appendix C.2 below. We will show how we can use our Blockloading framework together with both algorithmic approaches in section 13.3 below.
In order to start the inference process of the VBEM in the M–step (or alternatively in the E–step), we need a hard assignment of the relevance, ρ_i, for each vertex i and a fuzzy or hard cluster assignment, Q_i, for each relevant vertex i. We can also use randomly initialised assignments. In this case, we need appropriate informative hyperparameters for the prior distributions. The quality of the results is highly dependent on a good choice of these start values. We will address the issue of finding informative hyperparameters for the relevance assignments in section 13.3.2 below.
We can use our adaptive informative hyperparameters, we introduced in chapter 5, together with randomly initialised start values for the cluster assignments of the relevant vertices. We call this algorithmic variant the Relevance BlockVB++ algorithm.

13.2 Review of the Blockloading algorithm for the SBMIV

We have shown with our numerical experiments in chapter 10, that the divisive approach with subset based optimisation of our Blockloading algorithms greatly outperforms the batch algorithm, we reviewed in chapter 4 and section 6.1. For a review of other important subset optimisation and split-merge inference algorithms for clustering, we refer to chapter 7.

So, we also need a divisive algorithm which provides optimisation of subsets for optimised inference of the SBMIV in addition to the general VBEM inference algorithm of the SBMIV we proposed in section 13.1 above.
We will propose our Relevance Blockloading algorithm for the SBMIV in section 13.3. It will provide such a divisive algorithmic approach with subset optimisation. The Relevance Blockloading algorithm is based on our Blockloading algorithm [118, 116]. For a detailed discussion of the Blockloading algorithm and its properties we refer to chapter 8. For convenience, we shortly recall the Blockloading algorithm and its terminology before the introduction of the Relevance Blockloading algorithm.
We start with a cluster partition where all vertices are in one cluster and calculate the reference free energy, $F^{(ref)}$, of this cluster. The free energy after convergence [50] or Integrated variational Bayes (ILvB) [73] is a model selection criterion for the optimal number of clusters and cluster assignments of the vertices of the SBM. We reviewed important model selection criteria in chapter 6.
We expand the cluster partition matrix to two clusters by applying the VBEM BlockVB algorithm for the Poisson SBM, we presented in chapter 4, to two clusters. If the converged free energy of the resulting partition, $F^{(trial)}$, is lower than $F^{(ref)}$, e.g. improves $F^{(ref)}$, we update the reference cluster partition matrix and the parameters with the new results for two clusters.
After the initialisation of the algorithm, we choose the *active cluster* of the reference partition. The choice of the active cluster can affect the outcome of the calculation for networks with sparsely connected vertices [116] (and chapter 10), if the BlockVB algorithm is used. We discussed the max-probabilities-method in section 8.3, which lets us select the cluster $\max_l \sum_{l=1}^{K} \mathbb{E}(\lambda_{la}) + \sum_{\substack{l=1 \\ l\neq a}}^{K} \mathbb{E}(\lambda_{al}) = \sum_{l=1}^{K} \frac{\alpha_{al}}{\beta_{al}} + \sum_{\substack{l=1 \\ l\neq a}}^{K} \frac{\alpha_{la}}{\beta_{la}}, \forall l \in \{1,\ldots,K\} \equiv l_a$ as the active cluster. For other ways to select the active clusters and an in depth discussion, we refer to section 8.3.
When we employ the max–probabilities–method, sparsely connected vertices are grouped in one cluster for the first iterations of the Blockloading algorithm [116]. The result is, that sparsely connected vertices with low edge weights are kept in one extra cluster of the SBM. This property of the Blockloading algorithm leads easily to an adaption of the Blockloading inference scheme to the SBMIV, where this extra cluster of sparsely connected vertices is modelled explicitly.
In the Refinement Step, we check if vertices of the active cluster can be assigned to other clusters of the existing reference partition to improve the converged free energy. A detailed description of the Refinement Step can be found in section 8.1.
After the Refinement Step, we determine the active cluster again. We then try to split the active cluster into two new clusters to lower the reference free energy, like in the initialisation of the algorithm. If no improvement in either the Expansion or the Refinement Step was reached for all clusters of the existing partition, the Blockloading algorithm has converged. We sum up the Blockloading algorithm in the following overview:

Blockloading algorithm:
Input.–Adjacency matrix $\boldsymbol{A}$, model type.
Result–Cluster partition matrix $\boldsymbol{Q}^{(ref)}$, number of clusters $K^{(ref)}$ and parameters $\boldsymbol{\vartheta}^{(ref)}$.
(i) Blockloading Initialisation.

Main Loop.
(ii) Refinement Step.

(iii) Expansion Step.
(iv) Check for Convergence of all clusters.

One of the advantages of Blockloading compared to other variational inference methods for the SBM is, that the existing optimal partition for lower number of clusters beginning with one cluster is reused as start value partition in the following iterations of expansion and refinement. Therefore local optima are inferred one by one by the algorithm. An additional reason for the greatly improved performance is our algorithmic design of the Blockloading algorithm which focuses the inference on the efficient identification of all favourable local optima and thus a global optimum. We defined favourable local optima in Definition 4 in section 8.2.2. We also provided examples for favourable local optima in section 8.2.2.

13.3 The Relevance Blockloading Algorithm

In order to expand our Blockloading algorithm to the SBMIV, we need a model selection criterion to evaluate the outcome of the calculation of the Initialisation, Expansion and Refinement Step.
There exist three well established model selection criteria for the normal Poisson SBM: The asymptotic Integrated-completed-likelihood (ICL) of [18, 35, 84], the variational Integrated Likelihood variational Bayes (ILvb) criterion of [50, 73] and the non–asymptotic exact ICL of [29]. We also discussed these model selection criteria for the Poisson SBM in chapter 6.
The ILvb is the value of the free energy of the SBM with non-informative priors after convergence [73]. Then the optimal number of clusters is chosen for the result with the optimal value of the free energy after convergence. We also tried this approach with the free energy of the SBMIV of Proposition 12 in appendix C.1. We found in numerical tests that this approach gives biased results and leads to results where all vertices are considered irrelevant.
In these tests, we noticed that a model selection criterion for the SBMIV has to take into account, that vertices can enter or leave the relevant part of the cluster partition during the inference process. This fluctuation of vertices between relevant and irrelevant state also affects the calculation of the variational bound of the SBMIV and renders it inconsistent as a model selection criterion for the SBMIV.
We will also need an algorithm to initialise the cluster of irrelevant vertices for our VBEM algorithm of the SBMIV. This initialisation should be done early in the inference process to save computational time. We present the Initialisation of the Relevance Blockloading algorithm in section 13.3.2.
After the convergence of the Blockloading algorithm for the clusters of relevant vertices of the SBMIV, we check if the set of relevant vertices can be increased by changing the status of irrelevant vertices to relevant.
We propose the Relevance Expansion Step in section 13.3.2 for the Relevance Blockloading algorithm, where we check if the cluster of irrelevant vertices can be divided into an additional relevant cluster and an irrelevant cluster with a diminished number of vertices.

13.3.1 Model Selection for the SBMIV

We explained above that we can not use the converged free energy of the SBMIV because the optimal value of this converged free energy corresponds to a partition where all vertices are considered as irrelevant. We found an original way to use the ILvb (free energy) of [50, 73] for the vertex partitions returned by the VBEM inference with the Blockloading algorithm.
We consider the set of irrelevant vertices, $\rho_i = 0; \forall i \in \{1,\ldots,N\}$ as a special cluster and build the **combined cluster partition matrix** with the cluster assignments of the relevant vertices and a vector $\mathscr{R}$ which indicates the irrelevant vertices with $\mathscr{R}_i = 1-\rho_i; \forall i \in \{1,\ldots,N\}$. This combined cluster partition matrix is a $\boldsymbol{Q}^{(c)} \in \mathbb{R}^{N\times K^{(ref)}+1}$ matrix.
Then we calculate the Poisson free energy (ILvb) of Proposition 1 in section 4.1 for the combined cluster partition matrix $\boldsymbol{Q}^{(c)}$. This free energy is the reference free energy, $F^{(ref)}$.
The Blockloading framework is now used to check for the possibility of Refinement and Expansion of the combined cluster partition $\boldsymbol{Q}^{(c)}$ measured by the converged free energy of the combined cluster partition. This is also true if we do the Expansion Step of the cluster of the irrelevant vertices (see section 13.3.2). Before we do the Expansion step for the cluster of irrelevant vertices, we calculate the converged free energy of the current combined partition, $\boldsymbol{Q}^{(c)}$. Then we expand the cluster of the irrelevant vertices if possible with the help of the embedded BlockVB algorithm of appendix C.2. We conclude that the irrelevant vertices influence the model selection criterion, if we use this procedure.

13.3.2 Initialisation and Relevance Hyperparameters

We need start values for the expected relevance assignments of the vertices, $\boldsymbol{\rho}$, and a start cluster assignment of the relevant vertices $\boldsymbol{Q}^{(start)}$. In section 8.3.1, we explained that by starting the optimisation with the clusters with the overall highest density of edge connections first, we can lock sparsely and irregularly connected vertices within one cluster. So, this maximum-probabilities method, excludes vertices which we expect to be irrelevant according to the SBMIV. In section 10.7.1, the max-probabilities method lead to the best results for the tested earthquake network if we applied the Blockloading algorithm together with the BlockVB algorithm.
We want to transfer this approach to the inference algorithm of the SBMIV and therefore exclude irrelevant vertices at the beginning of the inference process, preferably in the first iteration of the inference algorithm.
We start the Blockloading algorithm with all vertices in one cluster and all vertices are set to relevant, e.g. $\rho_i = 1; \forall i \in \{1,\ldots,N\}$. We calculate the reference free energy, $F^{(ref)}$, of this partition, which for this case is the Poisson free energy (ILvb) in Proposition 1, section 4.1.
We recall that we aim to identify sparsely connected vertices with an uniform connection to the relevant part of the network. These irrelevant vertices are modelled by a SBM where the irrelevant vertices are connected with the same rate to all other vertices of the network (see section 12.2). We set special hyperparameters for the Gamma$(\gamma;\alpha_\gamma^0,\beta_\gamma^0)$ prior distribution to model this specific edge connection profile we assume of irrelevant vertices. We note that informative priors to set the variance of a Beta distribution for identifying relevant local and global features in a Variational Bayesian framework were used in [44] for their feature selection algorithm.

Sparsely and irregularly connected vertices in the earthquake network we will present in section 14 have of course some variance of the probabilities of the edge existence. Therefore, we set the first hyperparameter to $\alpha_\gamma^0 = 1$. A Gamma$(\gamma; 1, \beta_\gamma^0)$–distribution has the form of an exponential distribution [19], which covers a wide range of possible values for the parameter γ. To link this Gamma prior distribution to the constant rate parameter of the irrelevant part of the SBMIV, we calculate the parameter β_γ^0 so that the expectation value of γ, $\mathbb{E}[\gamma]$, is equal to the expected value of the same parameter for an SBM with all vertices in one cluster. So, we calculate the SBM with all vertices in one cluster which yields the parameters of the edge rate, α_{ER} and β_{ER}, which we calculate according to Proposition 2.
The Poisson SBM with all vertices in one cluster is a special case of the Erdös–Rényi–Graph (ER–Graph) [98], so we gave the parameters the suffix ER. With the help of these two parameters we see that

$$\frac{1}{\beta_\gamma^0} = \frac{\alpha_{ER}}{\beta_{ER}} \Rightarrow \beta_\gamma^0 = \frac{\beta_{ER}}{\alpha_{ER}}. \tag{13.23}$$

Thus, we have a $\gamma \sim$ Gamma$(\gamma; 1, \frac{\beta_{ER}}{\alpha_{ER}})$–prior–distribution which has the form of an exponential distribution and it holds that $\mathbb{E}[\gamma] = \frac{\beta_{ER}}{\alpha_{ER}}$.
Now that we have calculated the prior distributions for the irrelevant vertices, we can run our Relevance BlockVB algorithm for the SBMIV (see appendix C.2) for all vertices set to relevant and and all vertices in one cluster with the relevance hyperparameters $\left(1, \beta_\gamma^0\right)$ calculated above.
We note that because all vertices are in one cluster we do not need to calculate a cluster assignment during the E-step. Our relevance hyperparameters play the role of start values. We refer also to chapter 5 for a discussion of informative hyperparameters for the SBM.
The Relevance Initialisation returns the relevant vertices, which are grouped together in one cluster, and the irrelevant vertices.
Now, we calculate the converged free energy, $F^{(trial)}$, of the combined cluster partition matrix, $\boldsymbol{Q}^{(c)} \in \mathbb{R}^{N\times 2}$, which we introduced in section 13.3.1. If $F^{(trial)} < F^{(ref)}$ holds, we apply the Blockloading algorithm (section 8.1) to the relevant vertices returned by our Relevance BlockVB algorithm.

Another algorithmic approach is to apply the normal BlockVB algorithm for the Poisson SBM of section 4.2, to the active clusters of the relevant vertices. This algorithmic variant leads to the **Filtering Relevance Blockloading algorithm** we will introduce below.
We emphasise that we do not need repeated initialisations for different start values with this initialisation approach. We remark that this is a tremendous advantage compared to all other variational algorithms we are aware of, which all need repeated initialisations with different start values to some extent. This is especially important for large networks. We sum up our initialisation in the following algorithm:

Relevance Initialisation
Input.–Adjacency matrix $\boldsymbol{A}$. Cluster partition matrix with all vertices in one cluster.
Result–Initialisation of relevant and irrelevant vertices.

(i) Calculate the hyperparameters for all vertices assigned to one cluster.
(ii) Calculate the relevance hyperparameters.

(iii) Apply the Relevance BlockVB algorithm of appendix C.2 with the relevance hyperparameters.

We always calculate the reference free energy, $F^{(ref)}$, and the trial free energy, $F^{(trial)}$, of the combined partition matrix, $\boldsymbol{Q}^{(c)} \in \mathbb{R}^{N \times K^{(ref)}+1}$. When the Blockloading algorithm for the relevant vertices has converged, we can check if the set of relevant vertices can be expanded. To do this we use our newly introduced *Relevance Expansion Step* in section 13.3.3.

13.3.3 Relevance Expansion Step and Convergence

After the convergence of all active clusters with relevant vertices we proceed by setting the irrelevant vertices to active. For the expansion of the set of relevant vertices, we use the same framework as in the Relevance Initialisation Step with some adaptions. The main idea stays the same of setting back all vertices back to relevant status and grouping them in a new cluster.

Relevance Expansion Step We build the combined partition matrix, $\boldsymbol{Q}^{(c)}$, and set all vertices to relevant. Then we use the current reference parameters of the irrelevant cluster, $\left(\alpha_\gamma^{(ref)}, \beta_\gamma^{(ref)}\right)$, returned by the last iteration of the Blockloading inference for the relevant vertices, to determine the relevance hyperparameters following eqn. 13.23. We set all vertices in the irrelevant clusters to active and start the inference with the Relevance BlockVB algorithm for the combined partition matrix, $\boldsymbol{Q}^{(c)}$, and the relevance hyperparameters.
We only set irrelevant vertices to the added irrelevant cluster of the combined cluster partition matrix, $\boldsymbol{Q}^{(c)}$, in the E-step. We do not perform a full optimisation of the cluster partition matrix in the E–Step.
We found in numerical tests, that this procedure has better separation properties which means that less vertices are re–labeled from relevant to irrelevant, than by doing a full optimisation with respect to all relevant clusters.
A full E–step where all vertices in the active cluster could be assigned to any of the relevant clusters can lead to the merging of clusters which were separated in the iterations before.

Convergence Like in the initialisation step, we calculate the trial free energy, $F^{(trial)}$, for the returned combined trial cluster partition matrix, $\boldsymbol{Q}^{(c)}_{(trial)}$. If the reference free energy was improved, e.g. if $F^{(trial)} < F^{(ref)}$ holds, we update all parameters and hidden variables, $\left(\boldsymbol{\rho}^{(ref)}, \boldsymbol{Q}^{(ref)}, \boldsymbol{\Theta}^{(ref)}\right)$. Then we restart the Relevance Blockloading algorithm for the now updated set of relevant vertices.
We remark, that a relevant vertex may be found as irrelevant during the optimisation of relevant clusters, following the embedded E–Step of section 13.1, but a vertex can only enter the set of relevant vertices from irrelevant status during the Relevance Expansion Step where the set of irrelevant vertices is active.
If otherwise $F^{(trial)} \geq F^{(ref)}$ holds, the Relevance Blockloading algorithm has converged. We sum up the whole Embedded Relevance Algorithm:

Embedded Relevance Blockloading Algorithm
Input.–Adjacency matrix $\boldsymbol{A}$.
Result–Cluster partition matrix, $\boldsymbol{Q}^{(ref)}$, number of clusters, $K^{(ref)}$, parameters, $\boldsymbol{\Theta}^{(ref)}$ and the relevance assignment of vertices, $\boldsymbol{\rho}$.
(i) Relevance Initialisation Step.

Main Loop.
(ii) Embedded Refinement Step with Relevance BlockVB for active cluster.
(iii) Embedded Expansion Step with Relevance BlockVB for active cluster.
(iv) Check for Convergence of relevant clusters.
(v) Relevance Expansion Step.
(vi) Check for Convergence of the irrelevant cluster.

For the Embedded Relevance Blockloading algorithm we use the Relevance BlockVB algorithm presented in appendix C.2 in all cases. The irrelevant vertices do not influence the inference process of the cluster assignment of the relevant vertices at all. This feature of Embedded Relevance Blockloading is comparable to the Gibbs sampling procedure in [54]. So we say that the ERB is an algorithm **without noise influence**.

Filtering Relevance Blockloading Algorithm For the filtering variant of the Relevance Blockloading algorithm, we replace steps ***(ii)*** and ***(iii)*** of the Embedded Relevance Blockloading algorithm above with the BlockVB Refinement and Expansion Step proposed in section 4.2. We can also use our BlockVB++ of section 5.1 instead. We sum up the **Filtering Relevance Blockloading (FRB) algorithm**:

Filtering Relevance Blockloading Algorithm
Input.–Adjacency matrix $\boldsymbol{A}$.
Result–Cluster partition matrix, $\boldsymbol{Q}^{(ref)}$, number of clusters, $K^{(ref)}$, parameters, $\boldsymbol{\Theta}^{(ref)}$ and the relevance assignment of vertices, $\boldsymbol{\rho}$.
(i) Relevance Initialisation Step with Relevance BlockVB.

Main Loop.
(ii) Refinement Step with BlockVB or BlockVB++ for relevant active cluster.
(iii) Expansion Step with BlockVB or BlockVB++ for relevant active cluster.
(iv) Check for Convergence of relevant clusters.
(v) Relevance Expansion Step with Filtering variant of Relevance BlockVB.
(vi) Check for Convergence of the irrelevant cluster.

The Refinement and Expansion Step for the relevant vertices of the FRB algorithm are applied to the combined cluster partition matrix $\boldsymbol{Q}^{(c)}$. The difference to the Blockloading algorithm is, that the cluster of irrelevant vertices is only active in the Relevance Expansion Step (RE–Step).
We remark, that in the Refinement Step, vertices can leave the set of relevant vertices and become irrelevant but a vertex can only become relevant in the RE–Step. The presence of the irrelevant vertices in a separate cluster influences the inference process for the relevant vertices. This feature separates our FRB algorithm from the algorithm proposed in [54]. Therefore the FRB is an algorithm **with noise influence**.

We will compare both the Filtering and the Embedded Relevance Blockloading algorithm with numerical tests in section 14.

13.3.4 Successive Filtering with the Relevance BlockVB Algorithm

The Relevance Expansion Step of section 13.3.3 can be applied to a given adjacency matrix of a network repeatedly without inference of the relevant clusters of the model. We propose the following filtering procedure to cluster a network independently of different start values. We call this algorithm the ***Successive Filtering*** algorithm. It allows us to divide the cluster partition matrix into different macro–clusters consisting of several clusters which then can be further refined and expanded in parallel.
For a given adjacency matrix, we start with the Relevance Initialisation Step of section 13.3.2, where we start with a cluster partition of all vertices assigned to one cluster and set to relevant. We proceed by calculating the reference free energy, $F^{(ref)}$, and the relevance hyperparameters. The Relevance Expansion Step (RE–Step) yields a cluster of relevant vertices and the cluster of irrelevant vertices. We calculate the trial free energy of the combined cluster partition matrix.
As described in section 13.3.2, we check if the trial free energy, $F^{(trial)}$, is lower than the reference free energy, $F^{(ref)}$. If the $F^{(ref)}$ could be improved, we apply the Relevance Expansion Step to the new cluster of irrelevant vertices. We continue this procedure as long as $F^{(ref)}$ can be improved.
This algorithm does not need several re–initialisation with different vertices and converges very fast. It provides us with separated parts of the network. With the help of this algorithm we can extract subnetworks which consist of clusters with homogeneous connection profiles when compared to the rest of the network. On each of the returned subnetwork we perform the Relevance Expansion Step (RE–Step).

Chapter 14

Numerical Experiments of the SBMIV

14.1 Earthquake Network

The earthquake network, which was introduced in [1], maps the spatial and temporal succession of earthquakes of a chosen region to a network. We recall our short exposition of the construction of the earthquake network and some important facts about the dataset of [104]. Important statistical properties of earthquake catalogue data are inherited by the earthquake network [1, 3].
One important finding presented in [1, 3] is, that the degree distribution of the vertices of the earthquake networks under survey follows a heavy tails or scale free distribution. This also applies to earthquake network of the Southern California area, we will use as an example below [1, 3]. The consequence is, that the majority of vertices is sparsely and irregularly connected to other vertices of networks. There is a huge variance of the edge weights connecting the vertices. In [116] we estimated the Poisson SBM for the network of the Southern California Area (details presented below) with our Blockloading algorithm.
The earthquake network is constructed for a chosen geographical area and time span. A square grid is put on the area of interest [4]. The earthquake network unfolds in the following way:

(i) Place a vertex in the first square where seismic activity occurs at the start of the observation interval.
(ii) Place a second vertex where the next time seismic activity occurs and place a (directed) edge between the last two vertices of seismic activity pointing to the latest vertex of activity.
(iii) Continue until the end of observation.

We constructed the earthquake network of the Southern California area (32s, 37n; 122w, 114w) for the time interval from January 1, 1984 to December 31, 2013. We chose a square length of $10\,km$ for the grid and did not include depth information of the earthquake catalog contrary to [1]. This results in 4256 squares. We used the earthquake catalogue data from the Southern California Earthquake Data Center (SCEDC) [104].
Earthquake catalogues have a minimum magnitude of completeness (see e.g. [90] or

Table 14.1: Results of the Fully Bayesian (BlockVB++ algorithm) Filtering Relevance Blockloading algorithm with noise influence for the Poisson SBMIV for the weighted earthquake network. Normalised Mutual Information (NMI) calculated in comparison to the best result of all tests for the combined matrix $\boldsymbol{Q}^c$ and for the irrelevant vertices (IV). Results were ordered according to the difference to the reference free energy ΔF_{ref}. Number of clusters K. Best result of all tests performed on the earthquake network in the first entry of the left side.

ΔF_{ref}	0	61	88	92	133	172	210	302	406	480
NMI $\boldsymbol{Q}^c$	1	0.94	0.93	0.93	0.89	0.92	0.93	0.91	0.93	0.94
NMI IV	1	0.94	0.92	0.95	0.93	0.89	0.92	0.91	0.95	0.89
K	52	50	51	52	50	51	51	50	51	50
no. of IV	1051	1049	1053	1053	1103	1059	1068	1064	1051	1063

[53]). The earthquake catalogue is expected to list every earthquake with magnitude equal or higher than the magnitude of completeness [90]. It was shown in [53] that the SCEDC catalogue is complete for a magnitude of $M \geq 1.8$ on the Richter Scale from January 1, 1984 onwards. We used only earthquakes with magnitude $M \geq 1.8$ for the construction of the earthquake network.
We set the entries on the diagonal of the adjacency matrix of the earthquake network to zero. We discussed the removal of loops in section 10.6. These entries represent aftershocks in the earthquake network. The resulting adjacency matrix of the earthquake network has $N = 2324$ vertices and 58718 edges. The highest edge weight of the earthquake network was 240 and the lowest 1 (and 0 if there is no edge between the two vertices).
We evaluate and compare the numerical tests with the same principles as in [116]. So, in order to compare the values of the converged free energy F in the following tests, we take the best of all values of the converged free energy F, F_{ref}, and calculate the difference $\Delta F_{ref} = F - F_{ref} \geq 0$. We compare the converged free energy of all tests and algorithms with the overall best converged free energy F_{ref}^{best} of all tests by calculating $\Delta F_{ref}^{best} = F - F_{ref}^{best}$.
To compare different cluster partitions $\boldsymbol{Q}$ and $\boldsymbol{Q}_0$ we use the Normalised Mutual Information (NMI) (e.g. [112]). A NMI($\boldsymbol{Q}_0$,$\boldsymbol{Q}$) of 1 means that both partitions $\boldsymbol{Q}$ and $\boldsymbol{Q}_0$ are identical. The NMI is zero, when no information about $\boldsymbol{Q}_0$ can be deduced from $\boldsymbol{Q}$. We reviewed the NMI in section 10.2.

We tested the three versions of our Relevance Blockloading algorithm for the SBMIV presented in section 13.3: The Filtering Relevance Blockloading algorithm with noise influence and the BlockVB algorithm for the relevant vertices, the fully Bayesian Filtering Relevance Blockloading algorithm with the BlockVB++ algorithm and the Embedded Relevance Blockloading algorithm without noise influence and the BlockVB algorithm.
All algorithms were initialised for ten times with different start values. We used the relevance priors in all algorithms for the Initialisation and the Relevance Expansion Step presented in section 13.3.2.

The best result, measured by the converged free energy of the combined partition ma-

Table 14.2: Results of the Filtering Relevance Blockloading algorithm (BlockVB version) with noise influence for the Poisson SBMIV for the weighted earthquake network. Normalised Mutual Information (NMI) calculated in comparison to the best result of all tests for the combined matrix $\boldsymbol{Q}^c$ and for the irrelevant vertices (IV). Results were ordered according to the difference to the reference free energy ΔF_{ref}. The difference of the converged free energy of each test, F, when compared to the best reference free energy of all experiments, F_{ref}^{best},for the test of the earthquake network is denoted by $\Delta F_{ref}^{(best)}$. A positive difference signals a suboptimal value of the converged free energy F of the test. Number of clusters K.

ΔF_{ref}	0	130	272	299	301	333	523	537	617	635
ΔF_{ref}^{best}	36	166	308	335	337	369	559	573	654	672
NMI $\boldsymbol{Q}^c$	1	0.94	0.95	0.93	0.93	0.96	0.95	0.96	0.94	0.95
NMI IV	1	0.93	0.93	0.94	0.93	0.96	0.95	0.96	0.94	0.95
NMI best $\boldsymbol{Q}^c$	0.95	0.94	0.93	0.94	0.92	0.93	0.92	0.93	0.93	0.92
NMI best IV	0.91	0.94	0.94	0.95	0.95	0.9	0.92	0.95	0.93	0.89
K	49	47	47	46	45	48	45	45	46	45
no. of IV	1020	1039	1034	1038	1040	1016	1024	1037	1030	1010

Table 14.3: Results of the Embedded Relevance Blockloading algorithm without noise influence for the Poisson SBMIV for the weighted earthquake network. Normalised Mutual Information (NMI) calculated in comparison to the best result of all tests for the combined matrix $\boldsymbol{Q}^c$ and for the irrelevant vertices (IV). Results were ordered according to the difference to the reference free energy ΔF_{ref}. The difference of the converged free energy of each test, F, when compared to the best reference free energy of all experiments, F_{ref}^{best}, for the test of the earthquake network is denoted by $\Delta F_{ref}^{(best)}$. A positive difference signals a suboptimal value of the converged free energy F of the test. Number of clusters K.

ΔF_{ref}	0	134	175	242	382	412	438	687	875	1026
ΔF_{ref}^{best}	287	422	462	529	670	699	726	974	1162	1313
NMI $\boldsymbol{Q}^c$	1	0.83	0.81	0.82	0.77	0.82	0.95	0.8	0.77	0.93
NMI IV	1	0.66	0.4	0.7	0.4	0.66	0.95	0.65	0.4	0.95
NMI best $\boldsymbol{Q}^c$	0.82	0.85	0.79	0.85	0.8	0.83	0.82	0.84	0.78	0.81
NMI best IV	0.63	0.76	0.56	0.73	0.56	0.77	0.64	0.77	0.56	0.63
K	50	48	49	47	47	47	47	47	46	46
no. of IV	837	1023	1350	997	1348	1025	841	1034	1345	834

Table 14.4: Results of the Blockloading algorithm with non–informative priors for the Poisson SBM (BlockVB algorithm) applied to the weighted earthquake network. Normalised Mutual Information (NMI) calculated in comparison to the best result of all tests for the combined matrix $\boldsymbol{Q}^c$ and for the proxy cluster of irrelevant vertices (proxy IV). Results were ordered according to the difference to the reference free energy ΔF_{ref}. The difference of the free energy of each test, F, when compared to the best reference free energy of all experiments, F_{ref}^{best},for the test of the earthquake network is denoted by ΔF_{ref}^{best}. A positive difference signals a suboptimal value of the converged free energy F of the test. Number of clusters K. Comparison with the relevance matrix, ΔF_{ref} and the combined matrix of the best result of all tests measured by the converged free energy.

ΔF_{ref}	0	20	132	163	198	228	276	304	346	517
ΔF_{ref}^{best}	322	342	454	486	520	550	598	626	668	839
NMI $\boldsymbol{Q}^c$	1	0.95	0.91	0.9	0.91	0.92	0.92	0.81	0.92	0.92
NMI proxy IV	1	0.96	0.97	0.95	0.91	0.94	0.99	0.63	0.98	0.98
NMI best $\boldsymbol{Q}^c$	0.85	0.86	0.84	0.85	0.86	0.86	0.85	0.87	0.84	0.85
NMI best IV	0.76	0.78	0.78	0.78	0.81	0.79	0.77	0.75	0.76	0.76
K	46	46	46	48	48	48	48	45	44	46
no. of proxy IV	1157	1144	1149	1146	1120	1140	1156	931	1161	1163

trix, $\boldsymbol{Q}^{(c)}$, was returned by the fully Bayesian version of the Filtering Relevance Blockloading algorithm with the BlockVB++ algorithm with a converged free energy of $F_{ref}^{best} = 133414$ and $K_{ref} = 52$ clusters. We introduced the fully Bayesian BlockVB++ algorithm in chapter 5. The results of the tests for this algorithm are presented in table 14.1.

We calculated the Normalised Mutual Information (NMI) in comparison to the best result of all tests for each algorithm for the combined cluster partition matrix, $\boldsymbol{Q}^{(c)}$. This measure shows how reliably an algorithm finds the same combined cluster partition for different start values.

In order to calculate the reliability of the inference of the relevant and irrelevant vertices we build a partition matrix where all relevant vertices are assigned to one column of the matrix and the irrelevant vertices to the other column. Then we compare this relevance partition matrix with the relevance partition matrix of the best result, restricted to each tested algorithm, by calculating the NMI of theses matrices.

We repeat these calculations of the NMI with respect to the overall best result, F_{ref}^{best}, measured by the free energy, of all tested algorithms for the earthquake network.

A close second best result was returned by the Filtering Relevance Blockloading algorithm without noise influence and the BlockVB algorithm (section 4.2) for the relevant part of the model with a converged free energy of $F_{ref} = 133449$ ($\Delta F_{ref}^{best} = 36$) and $K = 49$ clusters. The variance of the converged free energy for this test was higher than for the Filtering Relevance Blockloading algorithm with BlockVB++. The highest difference of the converged free energy of Filtering Relevance Blockloading with BlockVB was $\Delta F_{ref} = 672$ and with $K = 45$ clusters.

In tables 14.1 and 14.2 we can see that the difference of the converged free energy ΔF_{ref} differs less for the fully Bayesian version of the Filtering Relevance algorithm (table 14.1), but that in general all results of both algorithms have a high degree of similarity. The mixed approach version seems to be bit more reliable whereas the fully Bayesian version returns the best results of all tests measured by the converged free

energy with non-informative priors. Both algorithms identify mostly the same vertices as irrelevant.
The results of the Embedded Relevance Blockloading algorithm without noise influence are less reliable than the those of the Filtering Blockloading algorithms measured by the similarity of the combined cluster partition matrices, relevance partition matrices and ΔF_{ref}. The best converged free energy of the Embedded Relevance algorithm was $\Delta F_{ref}^{best} = 287$ with $K = 50$ clusters and the worst $\Delta F_{ref}^{best} = 1313$ with $K = 46$ clusters. We also see in table 14.3 that the Embedded Relevance algorithm returns vastly differing numbers of irrelevant vertices. The combined cluster partition matrices returned by the Embedded Relevance algorithm have a sub par NMI of the reliability and the overall best result of all algorithms.
These results show, that for the Variational Bayesian Blockloading algorithm the Filtering approach works better contrary to the findings for the Gibbs Sampling algorithm of [54]. In [54] the Gibbs Sampler was applied to the closely related SIRM in [54], where the joint sampling of the relevant clusters and the relevance assignment were proposed as the best inference method for such a model with irrelevant vertices.

For comparison, we state the results for the normal Blockloading algorithm with the mixed approach (BlockVB algorithm) for the Poisson SBM in table 14.4. We remark that these results differ to those presented in [116] because we repeated the tests with an updated version of our code and we found a minor bug in our code for the calculation of the NMI. The Blockloading algorithm for the SBM does not explicitly model irrelevant vertices. We take the cluster with the lowest summed expected edge existence rates (see section 8.3.2) as a proxy for the irrelevant vertices. In all tests, this proxy cluster of irrelevant vertices also was the cluster with the highest number of vertices.
This observation and the built–in noise suppression with the max-prob-strategy for the choice of the active cluster presented in section 8.3.1, justify the choice of these clusters as a proxy for the cluster of irrelevant vertices.

The best result of the Blockloading algorithm for the Poisson SBM was $\Delta F_{ref}^{best} = 322$ and $K = 46$ clusters and the worst result was $\Delta F_{ref}^{best} = 839$ also with $K = 46$ clusters. The Blockloading algorithm for the Poisson SBM is reliable but less so than the Filtering Blockloading algorithm with non–informative priors (BlockVB algorithm). We show in table 14.4 that the proxy relevance cluster of the normal Blockloading algorithm has a similarity of less than $NMI = 0.8$ (with one exception) for all results. In comparison, the relevance clusters returned by the Filtering Blockloading algorithm have, with one exception, similarities of the proxy relevance partitions higher than $NMI = 0.9$.

We conclude that the best choice for an inference algorithm for the SBMIV of all tested algorithms is the fully Bayesian Filtering Relevance Blockloading algorithm with noise influence which also improves on the results of the Blockloading algorithm of [116] (repeated and extended in table 14.4) for the the normal SBM. The Filtering Relevance Blockloading algorithm with non-informative priors for the relevant clusters is also a viable choice and takes a close second place with respect to the the quality of the best results and a first with respect to general reliability. Both Filtering Relevance Blockloading algorithms, the fully Bayesian (BlockVB++) - and the non-informative BlockVB version, clearly return better results than the Blockloading algorithm for the normal Poisson SBM of [116].

Summary

The aim of this thesis was the development of new algorithms for the analysis of large and complex networks with applications to earthquake networks. Earthquake networks are directed and weighted networks which model the temporal and spatial succession of earthquakes. The mapping of the interdependency between the vertices of the networks representing the location of the earthquake and the temporal succession and occurrence of earthquakes through edges between the vertices leads to a considerable complexity. Assigning (clustering) the vertices with similar edge connection profiles to the same class (cluster) opens the possibility of a better understanding of the generation and topology of the network.
One model offering such a cluster assignment is the Stochastic Block Model with Poisson distributions (SBM). It is a special feature of the SBM to consider the possible interdependence of each vertex to all other vertices of the network and thus capturing the complexity of the network. This network interdependency is a major challenge for the inference of a specific SBM for a given network. The application of the Variational Bayesian EM (VBEM) algorithm was proposed as an option for the solution to this problem some years ago. The estimation of the SBM with the VBEM algorithm leads to a non-convex optimisation problem. The objective is to find the optimal number of clusters and assignments of the vertices to the clusters corresponding to a global optimum of the model selection criterion. In order to solve this optimisation problem existing VBEM algorithms often only allow the separate initialisation for different numbers of clusters. Using this batch approach, the optimisation has to be performed each time with respect to all vertices in the network and for each number of clusters. For earthquake networks or comparable large and complex networks this approach leads to unsatisfactory results with respect to quality and computation time.
At this point the present thesis comes in. A new type of algorithms for VBEM inference, called the Blockloading algorithms, is proposed. These algorithms combine the search of the optimal cluster assignment of the vertices and the number of clusters according to the model selection criterion in a divisive algorithm. Starting with the assignment of all vertices to a single cluster the search for an optimum is performed by splitting and refining the existing clusters. As this optimisation only involves subsets, two new, especially adapted VBEM algorithms, the BlockVB and BlockVB++ algorithms, were developed to meet this objective. The design of the Blockloading algorithms follows the observation, that the successful search for a global optimum with means of a divisive algorithm can be performed by the identification of several favourable local optima. These local optima are similar in the sense that by correctly splitting the right clusters they can be transformed into a global optimum. The Blockloading, automatic Blockloading, no reset Blockloading and Blockloading++ algorithms were designed with regard to this observation which lead to an efficient algorithmic design which was optimised for finding favourable local optima.
As large and complex networks often exhibit irregularly and sparsely connected vertices the Stochastic Block Model with irrelevant Vertices (SBMIV) was developed which explicitly models these irrelevant vertices. Building on our Blockloading algorithms, the Relevance Blockloading algorithm together with the Relevance BlockVB EM algorithm was introduced for the estimation of the SBMIV. Numerical tests of the newly developed methods were performed on synthetic networks, generated by the SBM, and an earthquake network. The newly developed Blockloading algorithms reached a previously unattained quality of the results and computation time for the clustering of the earthquake network and the synthetic networks when compared to comparable methods for the estimation of the SBM. It was demonstrated for an earthquake network that inference with the Relevance Blockloading algorithm can reliably identify irrelevant vertices. This estimation lead to an even better value of the model selection criterion than the inference of the comparable Poisson SBM which demonstrates the superiority of this approach.

Zusammenfassung

Ziel der Arbeit war die Entwicklung neuer Algorithmen zur Analyse von großen und komplexen Netzwerken mit Anwendung auf Erdbebennetzwerke. Erdbebennetzwerke sind gerichtete und gewichtete Netzwerke, welche die zeitliche und räumliche Abfolge von Erdbeben abbilden. Die Abbildung der wechselseitigen Abhängigkeit zwischen den Knoten des Netzwerks, welche den Ort des Erdbebens beschreiben, und der zeitlichen Abfolge und dem Auftreten von Erdbeben durch Kanten zwischen den Knoten führt zu erheblicher Komplexität. Um die Entstehung und Topologie des Netzwerks besser zu verstehen, bietet sich das Zusammenfassen (Clustern) von Knoten mit ähnlichen Kantenverbindungsprofilen in derselben Klasse (Cluster) an.
Ein Modell für solche Zusammenfassungen ist das Stochastische Block Modell mit Poisson Verteilungen (SBM). Das SBM hat die Besonderheit, dass es die mögliche wechselseitige Abhängigkeit von jedem Knoten zu allen anderen Knoten berücksichtigt, und so die Komplexität des Netzwerks erfasst. Diese wechselseitige Abhängigkeit ist gleichzeitig eine besondere Herausforderung für ein Lösungsverfahren zur Schätzung des konkreten SBMs für ein gegebenes Netzwerk. Vor einigen Jahren wurde der Variational Bayesian EM (VBEM) Algorithmus als Option zur Lösung dieses Problems vorgestellt. Die Schätzung eines SBM mit dem VBEM Algorithmus führt auf ein nicht-konvexes Optimierungsproblem. Dabei ist die optimale Anzahl der Cluster und Zuordnung der Knoten zu den Clustern zu finden, die einem globalen Optimum des Modellauswahlkriteriums entsprechen. Zur Lösung dieses Optimierungsproblems erlaubten bisherige VBEM Algorithmen allerdings häufig nur die getrennte Anwendung für verschiedene Anzahlen von Clustern. Dabei wird die Optimierung jedesmal in Abhängigkeit von allen Knoten des gesamten Netzwerks und für jede Clusterzahl neu durchgeführt. Dieses Vorgehen führt jedoch für Erdbebennetzwerke oder vergleichbar große und komplexe Netzwerke zu unzureichenden Ergebnissen in Bezug auf die Qualität und Rechengeschwindigkeit.
Genau an dieser Stelle setzt die vorliegende Arbeit an. Es wird eine neue Art von Algorithmen zur Anwendung von VBEM Algorithmen, sogenannte Blockloading Algorithmen vorgestellt, welche die Suche nach der optimalen Clusterzuordnung der Knoten und der Anzahl der Cluster gemäß des Modellauswahlkriteriums in einem Unterteilungalgorithmus kombinieren. Ausgehend von der Zuordnung aller Knoten zu einem einzigen Cluster wird ein Optimum durch das Aufteilen und Verfeinern der bestehenden Cluster gesucht. Da in dieser Optimierung nur Teilmengen der Knoten berücksichtigt werden, wurden mit den BlockVB und BlockVB++ Algorithmen zwei neue, dafür speziell angepasste VBEM Algorithmen entwickelt. Das Design der Blockloading Algorithmen folgt der Beobachtung, dass die erfolgreiche Suche nach einem globalen Optimum mit einem Unterteilungsalgorithmus durch die Identifizierung mehrerer günstiger lokaler Optima erfolgen kann. Diese lokalen Optima sind sich in dem Sinne ähnlich, dass sie alle durch das richtige Aufteilen der richtigen Cluster in ein globales Optimum überführt werden können. Mit den Blockloading, automatic Blockloading, no reset Blockloading und Blockloading++ Algorithmen wird diese Beobachtung durch ein effizientes algorithmisches Design berücksichtigt, welches für das Auffinden geeigneter lokaler Optima optimiert wurde.

Da komplexe und große Netzwerke häufig unregelmäßig und wenig verbundene Knoten aufweisen, wurde zusätzlich das Stochastische Block Modell mit irrelevanten Knoten (SBMIV) entwickelt, welches irrelevante Knoten gesondert zu berücksichtigen erlaubt. Aufbauend auf den Blockloading und BlockVB Algorithmen wurde der Relevance Blockloading Algorithmus mit dem Relevance BlockVB EM Algorithmus zur Schätzung des SBMIV vorgestellt. Anhand von synthetischen Netzwerken, die mit dem SBM erzeugt wurden, und einem Erdbebennetzwerk wurden numerische Tests der entwickelten Methoden durchgeführt. Im Vergleich zu vergleichbaren Berechnungsmethoden zur Schätzung des SBMs waren die neu entwickelten Blockloading Algorithmen in der Lage, sowohl das Erdbebennetzwerk als auch die synthetische Netzwerke in bisher nicht erreichter Qualität und Geschwindigkeit zu clustern. Anhand des Erdbebennetzwerks konnte zusätzlich gezeigt werden, dass der neue Relevance Blockloading Algorithmus bei der Schätzung eines SBMIV zuverlässig irrelevante Knoten identifizieren kann. Dies führte sogar zu einem besseren Wert des Modellauswahlkriteriums als die Schätzung des vergleichbaren Poisson-SBM, was die Überlegenheit dieses Ansatzes demonstriert.

Appendix A

Bernoulli BlockVB and BlockVB++ Algorithm

We propose our BlockVB and BlockVB++ algorithms for optimised inference of subsets of the vertices of a network according to the directed and unweighted Bernoulli SBM. Both algorithms work analogously to the BlockVB (see section 4.2) and BlockVB++ (see section 5.1) for the Poisson SBM.

Input: Start partition matrix $\boldsymbol{Q}^{(start)}$, active Cluster c and adjacency matrix $\boldsymbol{A}$.

Initialisation: Find indices I of vertices in the active cluster, $i \in c$.
Initialise the prior hyper parameters $\alpha_{kl}^0 = 1/2$, $\beta_{kl}^0 = 1/2\ \forall (k,l) \in \{1,\dots,K\}^2$ for the Beta prior distributions and $\delta_k^0 = 1 \forall k \in \{1,\dots,K\}$ for the Dirichlet prior distribution with non informative prior hyper parametes [61, 50, 73].

To save computational time in the (Maximisation Step) M–Step, we prepare the update formulas dependent on the vertices in the active cluster, $i \in I$: $S_{\alpha_{xy}} = \sum_{i \neq j}^{N} Q_{ix} Q_{jy} A_{ij}$, $S_{\beta_{xy}} = \sum_{i \neq j}^{N} Q_{ix} Q_{jy} (1 - A_{ij})$, $S_{\delta_k} = \sum_{i=1}^{N} Q_{ik}$.

If BlockVB++ is used:
Set fraction of the adaptive prior parameters: $f_p = 4$. Initialise adaptive informative prior hyper parameters, $\alpha_{cc}^0 = \frac{S_{\alpha_{cc}}}{f_p}$.

Calculate

$$S_{\alpha_{kl}}^I = \sum_{\substack{i=1 \\ i \neq j}}^{N} \sum_{\substack{j \in I \\ i \neq j}} Q_{ik} Q_{jl} A_{ij} + \sum_{\substack{i \in I \\ i \neq j}} \sum_{\substack{j \notin I \\ j=1}}^{N} Q_{ik} Q_{jl} A_{ij}, \tag{A.1}$$

$$S_{\beta_{kl}}^I = \sum_{\substack{i=1 \\ i \neq j}}^{N} \sum_{\substack{j \in I \\ i \neq j}} Q_{ik} Q_{jl} (1 - A_{ij}) + \sum_{\substack{i \in I \\ i \neq j}} \sum_{\substack{j \notin I \\ j=1}}^{N} Q_{ik} Q_{jl} (1 - A_{ij}), \tag{A.2}$$

$$S_{\delta_k}^I = \sum_{i \in I} Q_{ik}. \tag{A.3}$$

Calculate the update of the parameters $\forall k,l$:

$$\alpha_{kl} = S_{\alpha_{kl}} + \alpha^0_{kl}, \tag{A.4}$$
$$\beta_{kl} = S_{\beta_{kl}} + \beta^0_{kl}, \tag{A.5}$$
$$\delta_k = \sum_{i=1}^{N} Q_{ik} + \delta^0_k. \tag{A.6}$$
$$\tag{A.7}$$

Main Loop: Until convergence of the free energy $F[q(\cdot)]$ or the maximum number of iterations:

Expectation Step: Until convergence of the matrix entries Q_{lv}, $\forall (l,v) \in I \times \{1,\dots,K\}$.
Update of the matrix entries Q_{lv}, $\forall (l,v) \in I \times \{1,\dots,K\}$:

$$Q_{lv} \propto \exp\left(-\sum_{y=1}^{K} \sum_{\substack{j \in I \\ j \neq l}} A_{lj} Q_{vy} C_{vy} - \sum_{y=1}^{K} \sum_{\substack{j \in I \\ j \neq l}} A_{jl} Q_{jy} C_{yv} + D_{vy} \sum_{\substack{j \in I \\ j \neq l}} Q_{jy} \right), \tag{A.8}$$

where $C_{vy} = \psi(\beta_{vy}) - \psi(\alpha_{vy})$ and $D_{vy} = \psi(\beta_{vy}) - \psi(\beta_{yv}) - \psi(\alpha_{vy} + \beta_{vy}) - \psi(\alpha_{yv} + \beta_{yv})$ and $\psi(\cdot)$ is the Digamma function (see e.g. [19]).
Normalise the entries $Q^\star_{ik} = Q_{ik} / \left(\sum_{k=1}^{K} Q_{ik}\right)$ $\forall k$ of the matrix row i.
Check for convergence of the matrix entries Q_{ik}, $\forall (i,k) \in I \times \{1,\dots,K\}$.

M–Step: Update of the parameters of the variational distributions.
Prepare the updates of the parameters dependent on the active vertices $i \in I$:

$$S^{I_{old}}_{\alpha_{kl}} = S^{I}_{\alpha_{kl}}, \tag{A.9}$$
$$S^{I_{old}}_{\beta_{kl}} = S^{I}_{\beta_{kl}}, \tag{A.10}$$
$$S^{I_{old}}_{\delta_k} = S^{I}_{\delta_k}. \tag{A.11}$$

Calculate $S^I_{\alpha_{kl}}$, $S^I_{\beta_{kl}}$ and $S^I_{\delta_k}$ for all $(k,l) \in \{1,\dots,K\}^2$ like in the equations A.1, A.2 and A.3 above.

If BlockVB++ is used: Update of the adaptive informative prior hyper parameters analogously to the BlockVB++ algorithm (Algorithm 5) of the Poisson SBM in section 5.1.

Do the updates of the parameters α_{kl}, β_{kl} and δ_k $\forall k,l$:

$$\alpha_{kl} = S_{\alpha_{kl}} - S^{I_{old}}_{\alpha_{kl}} + S^{I}_{\alpha_{kl}} + \alpha^0_{kl}, \tag{A.12}$$
$$\beta_{kl} = S_{\beta_{kl}} - S^{I_{old}}_{\beta_{kl}} + S^{I}_{\beta_{kl}} + \beta^0_{kl}, \tag{A.13}$$
$$\delta_k = S_{\delta_k} - S^{I_{old}}_{\delta_k} + S^{I}_{\delta_k} + \delta^0_k. \tag{A.14}$$
$$\tag{A.15}$$

Convergence: Calculate the free energy F:

$$F\left[\boldsymbol{Q},\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\delta}\right] = \ln\left(\frac{\Gamma\left(\sum_{x=1}^{K}\delta_x\right)\prod_{x=1}^{K}\Gamma(\delta_x^0)}{\Gamma\left(\sum_{x=1}^{K}\delta_x^0\right)\prod_{x=1}^{K}\Gamma(\delta_x)}\right) + \sum_{x,y}^{K}\ln\left(\frac{B(\alpha_{xy},\beta_{xy})}{B(\alpha_{xy}^0,\beta_{xy}^0)}\right) + \sum_{i=1}^{N}\sum_{x=1}^{K}Q_{ix}\ln Q_{ix}. \tag{A.16}$$

Check for the convergence of F.

Appendix B

Proofs and Propositions for VBEM inference of the Poisson Stochastic Block Model

Proof of **Proposition 4**.

Proof. The terms of the upper bound F dependent on $q(\boldsymbol{Z})$ are:

$$F[q(\boldsymbol{Z})] = -\mathbb{E}_{\boldsymbol{Z},\boldsymbol{\pi},\boldsymbol{\lambda}}[\ln p(\boldsymbol{A},\boldsymbol{Z},\boldsymbol{\lambda},\boldsymbol{\pi})] + \mathbb{E}_{\boldsymbol{Z}}[\ln q(\boldsymbol{Z})] + \text{const.} \tag{B.1}$$

$$= -\mathbb{E}_{\boldsymbol{Z},\boldsymbol{\lambda}}[\ln p(\boldsymbol{A}|\boldsymbol{Z},\boldsymbol{\lambda})] - \mathbb{E}_{\boldsymbol{Z},\boldsymbol{\pi}}[\ln p(\boldsymbol{Z}|\boldsymbol{\pi})] + \mathbb{E}_{\boldsymbol{Z}}[\ln q(\boldsymbol{Z})] + \text{const.} \tag{B.2}$$

$$\begin{aligned} &= \mathbb{E}_{\boldsymbol{Z}}\left(\sum_{q,l}^{K}\sum_{\substack{i,j\\ i\neq j}}^{N}\big(-Z_{iq}Z_{jl}A_{ij}\big(\psi(\alpha_{ql}) - \ln(\beta_{ql})\big) + Z_{iq}Z_{jl}\left(\frac{\alpha_{ql}}{\beta_{ql}}\right)\big)\right) \\ &- \mathbb{E}_{\boldsymbol{Z}}\left(\sum_{i=1}^{N}\sum_{q=1}^{N} Z_{iq}\left(\psi(\delta_q) - \psi\big(\sum\left(\sum_{l=1}^{K}\delta_l\right)\big)\right)\right) + \mathbb{E}_{\boldsymbol{Z}}\left(\sum_{i=1}^{N} q(\boldsymbol{Z}_i)\ln q(\boldsymbol{Z}_i)\right) \\ &+ \text{const.}. \end{aligned} \tag{B.3}$$

Variational Bayesian optimisation of F with respect to $q(\boldsymbol{Z}_a)$ yields:

$$\ln q(\boldsymbol{Z}_a) \propto \sum_{v=1}^{K} Z_{av}\left(\sum_{q=1}^{K}\sum_{\substack{i=1\\ i\neq a}}^{N} Q_{iq}A_{ai}C_{vq} + Q_{iq}A_{ia}C_{qv} - Q_{iq}D_{vq} + G_v\right). \tag{B.4}$$

Taking the exponential of eqn. (B.4) yields:

$$\begin{aligned} q(\boldsymbol{Z}_a) \propto \exp\Big(\sum_{v=1}^{K} Z_{av}\Big(&\sum_{\substack{i=1\\ i\neq a}}^{N}\sum_{q=1}^{K} A_{ai}Q_{iq}C_{vq} + \sum_{\substack{i=1\\ i\neq a}}^{N}\sum_{q=1}^{K} A_{ia}Q_{iq}C_{qv} \\ &- \sum_{\substack{i=1\\ i\neq a}}^{N}\sum_{q=1}^{K} Q_{iq}D_{vq} + G_v\Big)\Big). \end{aligned} \tag{B.5}$$

After normalisation, eqn. (B.5) shows that $q(\boldsymbol{Z}_a)$ has the functional form of a Multinomial $\mathscr{M}(\boldsymbol{Z}_a; 1, \boldsymbol{Q}_a = \{Q_{a1},\ldots,Q_{aK}\})$ distribution. □

Proof of **Proposition 2** which provides the functional form of the variational distribution $q(\lambda_{ql})\ \forall q,l$ and the update equations for the hyperparameters $(\alpha_{ql},\beta_{ql})\ \forall q,l$.

Proof. The terms of the upper bound F dependent on $\boldsymbol{\lambda}$ are:

$$F\left[q(\boldsymbol{\lambda})\right] = -\mathbb{E}_{\boldsymbol{Z},\boldsymbol{\lambda}}\left[\ln p(\boldsymbol{A}\,|\,\boldsymbol{Z},\boldsymbol{\lambda})\right] - \mathbb{E}_{\boldsymbol{\lambda}}\left[\ln p(\boldsymbol{\lambda})\right] + \mathbb{E}_{\boldsymbol{\lambda}}\left[\ln q(\boldsymbol{\lambda})\right] + const. \tag{B.6}$$

$$= \sum_{q,l}^{K}\sum_{\substack{i,j\\ i\neq j}}^{N}\left(- Q_{iq}Q_{jl}A_{ij}\mathbb{E}_{\boldsymbol{\lambda}}\left[\ln\lambda_{ql}\right] + Q_{iq}Q_{jl}\mathbb{E}_{\boldsymbol{\lambda}}\left[\lambda_{ql}\right]\right)$$

$$- \mathbb{E}_{\boldsymbol{\lambda}}\left[\ln p(\boldsymbol{\lambda})\right] + \mathbb{E}_{\boldsymbol{\lambda}}\left[\ln q(\boldsymbol{\lambda})\right] + const. \tag{B.7}$$

Terms which do not depend on λ_{ql} will be absorbed into the constant. Optimisation of F according to the Variational Bayesian framework with respect to $q(\lambda_{ql})$ leads to:

$$\ln q(\lambda_{ql}) = \sum_{\substack{i,j\\ i\neq j}}^{N}\left(Q_{iq}Q_{jl}A_{ij}\ln\lambda_{ql} - Q_{iq}Q_{jl}\lambda_{ql}\right) + \alpha_{ql}^{0}\ln(\beta_{ql}^{0}) - \ln\Gamma(\alpha_{ql}^{0})$$

$$+ (\alpha_{ql}^{0}-1)\ln\lambda_{ql} - \beta_{ql}^{0}\lambda_{ql} + const.. \tag{B.8}$$

We take the exponential of eqn. (B.8) and absorb terms not dependent on λ_{ql} into the constant. It follows that

$$q(\lambda_{ql}) \propto \exp\Big(\big(\sum_{\substack{i,j\\ i\neq j}}^{N} Q_{iq}Q_{jl}A_{ij} + \alpha_{ql}^{0} - 1\big)\ln\lambda_{ql} - \big(\sum_{\substack{i,j\\ i\neq j}}^{N} Q_{iq}Q_{jl} + \beta_{ql}^{0}\big)\lambda_{ql}\Big). \tag{B.9}$$

After normalisation of equation (B.9) we see that $q(\lambda_{ql})$ has the functional form of a Gamma$(\sum_{i\neq j}^{N} Q_{ik}Q_{jl}A_{ij} + \alpha_{kl}^{0}, \sum_{i\neq j}^{N} Q_{ik}Q_{jl} + \beta_{kl}^{0})$ distribution.

□

Proof of **Proposition 1** of the free energy of the Poisson SBM.

Proof. The free energy (variational upper bound) is given by:

$$F[q(\cdot)] = -\sum_{\boldsymbol{Z}} \int \ln\left(\frac{p(\boldsymbol{A},\boldsymbol{Z},\boldsymbol{\lambda},\boldsymbol{\pi})}{q(\boldsymbol{Z},\boldsymbol{\lambda},\boldsymbol{\pi})}\right) q(\boldsymbol{Z},\boldsymbol{\lambda},\boldsymbol{\pi})\mathrm{d}\boldsymbol{\lambda}\,\mathrm{d}\boldsymbol{\pi} \tag{B.10}$$

$$= -\mathbb{E}_{\boldsymbol{Z},\boldsymbol{\lambda},\boldsymbol{\pi}}[\ln p(\boldsymbol{A}|\boldsymbol{Z},\boldsymbol{\lambda})] - \mathbb{E}_{\boldsymbol{\lambda},\boldsymbol{\pi}}[\ln p(\boldsymbol{Z}|\boldsymbol{\pi})] - \mathbb{E}_{\boldsymbol{\lambda}}[\ln p(\boldsymbol{\lambda})]$$

$$- \mathbb{E}_{\boldsymbol{\pi}}[\ln p(\boldsymbol{\pi})] + \sum_{i=1}^{N} \mathbb{E}_{Z_i}[\ln q(Z_i)] + \mathbb{E}_{\boldsymbol{\pi}}[\ln q(\boldsymbol{\pi})] + \mathbb{E}_{\boldsymbol{\lambda}}[\ln q(\boldsymbol{\lambda})] \tag{B.11}$$

$$= \sum_{q,l}^{K} \sum_{\substack{i,j \\ i\neq j}}^{N} \big(- Q_{iq}Q_{jl}A_{ij}\left(\psi(\alpha_{ql}) - \ln(\beta_{ql})\right) + Q_{iq}Q_{jl}\left(\frac{\alpha_{ql}}{\beta_{ql}}\right)\big)$$

$$+ \sum_{q,l}^{K} -(\alpha_{ql}^0 - 1)(\psi(\alpha_{ql}) - \ln(\beta_{ql})) + \ln\left(\Gamma(\alpha_{ql}^0\right)$$

$$+ \beta_{ql}^0\left(\frac{\alpha_{ql}}{\beta_{ql}}\right) - \alpha_{ql}^0 \ln(\beta_{ql}^0) + \alpha_{ql}\ln(\beta_{ql}) - \ln\left(\Gamma(\alpha_{ql})\right)$$

$$+ (\alpha_{ql} - 1)\left(\psi(\alpha_{ql}) - \ln(\beta_{ql})\right) - \beta_{ql}\left(\frac{\alpha_{ql}}{\beta_{ql}}\right)$$

$$- \sum_{q=1}^{K}\sum_{i=1}^{N} Q_{iq}\big(\psi(\delta_q) - \psi\left(\sum_{l=1}^{K}\delta_q\right)\big) - \ln\left(\Gamma\left(\sum_{l=1}^{K}\delta_l^0\right)\right)$$

$$+ \sum_{q=1}^{K}\ln\left(\Gamma(\delta_q^0)\right) - \sum_{q=1}^{K}(\delta_q^0 - 1)\left(\psi(\delta_q) - \psi\left(\sum_{l=1}^{K}\delta_l\right)\right)$$

$$+ \ln\left(\Gamma\left(\sum_{q=1}^{K}\delta_q\right)\right) - \sum_{q=1}^{K}\ln\left(\Gamma(\delta_q)\right)$$

$$+ \sum_{q=1}^{K}(\delta_q - 1)\left(\psi(\delta_q) - \psi\left(\sum_{l=1}^{K}\delta_l\right)\right) + \sum_{i=1}^{N}\sum_{k=1}^{K} Q_{ik}\ln Q_{ik}. \tag{B.12}$$

We plug the update equations of Propositions 2 and 3 for α_{kl}, β_{kl} and $\delta_k \forall k, l \in \{1,\ldots,K\}$ into eqn. B.12. This yields equation 4.4 of Proposition 1 in chapter 4.

□

Appendix C

Appendix: Stochastic Block Model with irrelevant Vertices

C.1 Proofs and Propositions

Proof of **Proposition 6** in section 4:

Proof. The terms of the upper variational bound F dependent on $q(\boldsymbol{\phi})$ are:

$$F[q(\boldsymbol{\phi})] = -\mathbb{E}_{\boldsymbol{R},\boldsymbol{\phi}}[\ln p(\boldsymbol{R}|\boldsymbol{\phi})] - \mathbb{E}_{\boldsymbol{\phi}}(\ln p(\boldsymbol{\phi})) + \mathbb{E}_{\boldsymbol{\phi}}(\ln q(\boldsymbol{\phi})) + \text{const.}. \tag{C.1}$$

$$= -\sum_{i=1}^{N} \left(\rho_i \mathbb{E}_{\boldsymbol{\phi}}(\ln \phi_i) + (1-\rho_i)\mathbb{E}_{\boldsymbol{\phi}}\left(\ln(1-\phi_i)\right)\right) - \mathbb{E}_{\boldsymbol{\phi}}(\ln p(\boldsymbol{\phi})) + \mathbb{E}_{\boldsymbol{\phi}}(\ln q(\boldsymbol{\phi})) + \text{const.}. \tag{C.2}$$

We use Variational Bayesian optimisation of F with respect to $q(\phi_i)$ and absorb all terms which do not depend on ϕ_i into the constant. This yields:

$$\ln q(\phi_i) = (\zeta_i^0 - 1)\ln(\phi_i) + (\eta_i^0 - 1)\ln(1-\phi_i) + \rho_i \ln \phi_i + (1-\rho_i)\ln(1-\phi_i) + \text{const.}. \tag{C.3}$$

We take the exponential of eqn. (C.3) and it follows that

$$q(\phi_i) \propto \exp\left(\left(\rho_i + \zeta_i^0 - 1\right)\ln \phi_i + \left((1-\rho_i) + \eta_i^0 - 1\right)\ln(1-\phi_i)\right). \tag{C.4}$$

Equation C.4 shows that $q(\phi_i)$ has the functional form of a $\text{Beta}(\rho_i + \zeta_i^0, (1-\rho_i) + \eta_i^0)$ Beta distribution, so after normalisation it holds that $q(\phi_i) = \text{Beta}(\phi_i; \zeta_i, \eta_i)$, where

$$\zeta_i = \rho_i + \zeta_i^0, \tag{C.5}$$

$$\eta_i = (1-\rho_i) + \eta_i^0. \tag{C.6}$$

for $\forall i \in \{1,\ldots,N\}$. □

Proof of **Proposition 7** in section 4:

Proof. The terms of the free energy F which depend on $q(\boldsymbol{\pi})$ are:

$$F[q(\boldsymbol{\pi})] = -\mathbb{E}_{\boldsymbol{Z},\boldsymbol{R},\boldsymbol{\pi}}[\ln p(\boldsymbol{Z}|\boldsymbol{\pi},\boldsymbol{R})] - \mathbb{E}_{\boldsymbol{\pi}}[\ln p(\boldsymbol{\pi})] + \mathbb{E}_{\boldsymbol{\pi}}[\ln q(\boldsymbol{\pi})] + \text{const.} \tag{C.7}$$

$$= -\sum_{q=1}^{K}\sum_{i=1}^{N} \rho_i Q_{iq} \mathbb{E}_{\boldsymbol{\pi}}[\ln \pi_q] - \mathbb{E}_{\boldsymbol{\pi}}[\ln p(\boldsymbol{\pi})] + \mathbb{E}_{\boldsymbol{\pi}}[\ln q(\boldsymbol{\pi})] + \text{const.}. \tag{C.8}$$

Variational Bayesian optimisation of the Free Energy F with respect to $q(\boldsymbol{\pi})$ and absorbing terms which do not depend on $\boldsymbol{\pi}$ into the constant yields:

$$\ln q(\boldsymbol{\pi}) = \sum_{q=1}^{K}\sum_{i=1}^{N} \rho_i Q_{iq} \ln \pi_q + \sum_{q=1}^{K} (\delta_q^0 - 1) \ln \pi_q + \text{const.} \tag{C.9}$$

$$\Rightarrow\ q(\boldsymbol{\pi}) \propto \exp\left(\sum_{q=1}^{K}\sum_{i=1}^{N} \left(\rho_i Q_{iq} + \delta_q^0 - 1\right) \ln \pi_q \right). \tag{C.10}$$

Normalisation of eqn. (C.10) shows that $q(\boldsymbol{\pi})$ has the functional form of a $\text{Dir}(\boldsymbol{\pi};\boldsymbol{\delta})$ Dirichlet distribution with the update equations

$$\delta_k = \sum_{i=1}^{N} \rho_i Q_{ik} + \delta_k^0;\ \forall k \in \{1,\ldots,K\}, \tag{C.11}$$

for the hyper parameters. □

Proof of **Proposition 8** in section 4:

Proof. The terms of the upper bound F dependent on $q(\gamma)$ are:

$$F[q(\gamma)] = \sum_{\substack{i,j \\ i\neq j}}^{N} \Big(-(1-\rho_i\rho_j A_{ij}) \mathbb{E}_\gamma(\ln\gamma) + (1-\rho_i\rho_j)\mathbb{E}_\gamma(\gamma) \Big)$$
$$- \mathbb{E}_\gamma(\ln p(\gamma)) + \mathbb{E}_\gamma(\ln q(\gamma)) + \text{const.}. \tag{C.12}$$

We use the Variational Bayesian framework for the optimisation of F with respect to $q(\gamma)$ and absorb terms which are independent of γ into const.. This yields:

$$\ln q(\gamma) = \sum_{\substack{i,j \\ i\neq j}}^{N} \Big((1-\rho_i\rho_j) A_{ij} \ln(\gamma) - (1-\rho_i\rho_j)\gamma + (\alpha_\gamma^0 - 1)\ln(\gamma) - \beta_\gamma^0 \gamma \Big) + \text{const.}. \tag{C.13}$$

We take the exponential of eqn. (C.13). Thus, it follows that

$$q(\gamma) \propto \exp\left(\left(\left(\sum_{\substack{i,j \\ i\neq j}}^{N} (1-\rho_i\rho_j) A_{ij} \right) + \alpha_\gamma^0 - 1 \right) \ln(\gamma) - \left(\sum_{\substack{i,j \\ i\neq j}}^{N} (1-\rho_i\rho_j) + \beta_\gamma^0 \right) \gamma \right). \tag{C.14}$$

Equation C.14 shows that $q(\gamma)$ has the functional form of a $\text{Gamma}(\sum_{\substack{i,j \\ i\neq j}}^{N} (1-\rho_i\rho_j)A_{ij} + \alpha_\gamma^0, \sum_{\substack{i,j \\ i\neq j}}^{N} (1-\rho_i\rho_j) + \beta_\gamma^0)$ distribution.

□

Proof of **Proposition 9** in section 4:

Proof. We collect the terms of the upper bound F dependent on $q(\boldsymbol{\lambda})$:

$$\begin{aligned} F[q(\boldsymbol{\lambda})] =& -\mathbb{E}_{\boldsymbol{Z},\boldsymbol{R},\boldsymbol{\lambda},\gamma}[\ln p(\boldsymbol{A}|\boldsymbol{Z},\boldsymbol{R},\boldsymbol{\lambda},\gamma)] + \mathbb{E}_{\boldsymbol{\lambda}}[\ln q(\boldsymbol{\lambda})] \\ & - \mathbb{E}_{\boldsymbol{\lambda}}[\ln p(\boldsymbol{\lambda})] + \text{const.} \qquad \text{(C.15)} \\ =& \sum_{q,l}^{K}\sum_{\substack{i,j \\ i\neq j}}^{N} \big(-\rho_i Q_{iq}\rho_j Q_{jl} A_{ij}\mathbb{E}_{\boldsymbol{\lambda}}\left[\ln\lambda_{ql}\right] + \rho_i\rho_j Q_{iq}Q_{jl}\mathbb{E}_{\boldsymbol{\lambda}}\left[\lambda_{ql}\right]\big) \\ & - \mathbb{E}_{\boldsymbol{\lambda}}\left[\ln p(\boldsymbol{\lambda})\right] + \mathbb{E}_{\boldsymbol{\lambda}}\left[\ln q(\boldsymbol{\lambda})\right] + \text{const.}. \qquad \text{(C.16)} \end{aligned}$$

We apply Variational Bayesian optimisation to F with respect to $q(\lambda_{ql})$ and absorb terms not dependent on λ_{ql} into const.. This yields:

$$\begin{aligned} \ln q(\lambda_{ql}) =& \sum_{\substack{i,j \\ i\neq j}}^{N} \left(\rho_i\rho_j Q_{iq}Q_{jl}A_{ij}\ln\lambda_{ql} - \rho_i\rho_j Q_{iq}Q_{jl}\lambda_{ql}\right) \\ & - \left(\alpha^0_{ql}\ln(\beta^0_{ql}) + \ln\Gamma(\alpha^0_{ql}) + (\alpha^0_{ql}-1)\ln\lambda_{ql} - \beta^0_{ql}\lambda_{ql}\right) + \text{const.}. \qquad \text{(C.17)} \end{aligned}$$

We take the exponential of eqn. (C.17) and see that

$$q(\lambda_{ql}) \propto \exp\Big(\big(\sum_{\substack{i,j \\ i\neq j}}^{N} \rho_i\rho_j Q_{iq}Q_{jl}A_{ij} + \alpha^0_{ql} - 1\big)\ln\lambda_{ql} - \big(\sum_{\substack{i,j \\ i\neq j}}^{N} \rho_i\rho_j Q_{iq}Q_{jl} + \beta^0_{ql}\big)\lambda_{ql}\Big). \qquad \text{(C.18)}$$

Equation (C.18) shows that $q(\lambda_{ql})$ has the functional form of a Gamma$(\sum_{i\neq j}^{N}\rho_i\rho_j Q_{ik}Q_{jl}A_{ij} + \alpha^0_{kl}, \sum_{i\neq j}^{N}\rho_i\rho_j Q_{ik}Q_{jl} + \beta^0_{kl})$ distribution.

□

Proof of **Proposition 10** in section 4:

Proof. We collect the terms of the free energy F which depend on $q(\boldsymbol{Z})$:

$$\begin{aligned} F[q(\boldsymbol{Z})] =& -\mathbb{E}_{\boldsymbol{Z},\boldsymbol{R},\boldsymbol{\Theta}}[\ln p(\boldsymbol{A},\boldsymbol{Z},\boldsymbol{R},\boldsymbol{\lambda},\gamma,\boldsymbol{\pi},\boldsymbol{\phi})] + \mathbb{E}_{\boldsymbol{Z}}[\ln q(\boldsymbol{Z})] + \text{const.} \qquad \text{(C.19)} \\ =& -\mathbb{E}_{\boldsymbol{Z},\boldsymbol{R},\boldsymbol{\lambda},\gamma}[\ln p(\boldsymbol{A}|\boldsymbol{Z},\boldsymbol{R},\boldsymbol{\lambda},\gamma)] - \mathbb{E}_{\boldsymbol{R},\phi}[\ln p(\boldsymbol{R}|\boldsymbol{\phi})] - \mathbb{E}_{\boldsymbol{Z},\boldsymbol{R},\boldsymbol{\pi}}[\ln p(\boldsymbol{Z}|\boldsymbol{\pi},\boldsymbol{R})] \\ & + \mathbb{E}_{\boldsymbol{Z}}[\ln q(\boldsymbol{Z})] + \text{const.} \qquad \text{(C.20)} \\ =& \mathbb{E}_{\boldsymbol{Z}}\Big(-\sum_{\substack{i,j \\ i\neq j}}^{N}\sum_{q,l}^{K}\rho_i\rho_j Z_{iq}Z_{jl}A_{ij}\mathbb{E}(\ln\lambda_{ql}) + \sum_{\substack{i,j \\ i\neq j}}^{N}\sum_{q,l}^{K}\rho_i\rho_j Z_{ik}Z_{jl}\mathbb{E}(\lambda_{ql}) \\ & - \sum_{i=1}^{N}\sum_{q=1}^{K}\rho_i Z_{iq}\mathbb{E}(\ln\pi_q)\Big) + \mathbb{E}_{\boldsymbol{Z}}\Big(\sum_{i=1}^{N} q(\boldsymbol{Z}_i)\ln q(\boldsymbol{Z}_i)\Big) + \text{const.}. \qquad \text{(C.21)} \end{aligned}$$

We apply Variational Bayesian optimisation to F with respect to $q(\boldsymbol{Z}_a)$ and absorb terms independent of $\boldsymbol{Z}_a$ into const.. This yields:

$$\begin{aligned} \ln q(\boldsymbol{Z}_a) =& \sum_{v=1}^{K} Z_{av}\Big(\sum_{\substack{i=1 \\ i\neq a}}^{N}\sum_{q=1}^{K}\rho_a\rho_i Q_{iq}A_{ai}\mathbb{E}(\ln\lambda_{vq}) + \sum_{\substack{i=1 \\ i\neq a}}^{N}\sum_{q=1}^{K}\rho_a\rho_i Q_{iq}A_{ia}\mathbb{E}(\ln\lambda_{qv}) \\ & - \sum_{\substack{i=1 \\ i\neq a}}^{N}\sum_{q=1}^{K}\rho_a\rho_i Q_{iq}\left(\mathbb{E}(\lambda_{vq}) + \mathbb{E}(\lambda_{qv})\right) + \rho_a\mathbb{E}(\ln\pi_v)\Big) + const.. \qquad \text{(C.22)} \end{aligned}$$

We take the exponential of eqn. C.22 which leads to:

$$q(\boldsymbol{Z}_a) \propto \exp\Big(\sum_{v=1}^{K} Z_{av}\Big(\sum_{\substack{i=1\\ i\neq a}}^{N}\sum_{q=1}^{K}\rho_a\rho_i Q_{iq}A_{ai}\Big(\psi(\alpha_{vq})-\ln(\beta_{vq})\Big)+\sum_{\substack{i=1\\ i\neq a}}^{N}\sum_{q=1}^{K}\rho_a\rho_i Q_{iq}A_{ia}\Big(\psi(\alpha_{qv})-\ln(\beta_{qv})\Big)$$
$$-\sum_{\substack{i=1\\ i\neq a}}^{N}\sum_{q=1}^{K}\rho_a\rho_i Q_{iq}\left(\frac{\alpha_{vq}}{\beta_{vq}}+\frac{\alpha_{qv}}{\beta_{qv}}\right)+\rho_a\Big(\psi(\delta_q)-\psi(\sum\left(\sum_{l=1}^{K}\delta_l\right))\Big)\Big)\Big), \tag{C.23}$$

where $\psi(\cdot)$ is the Digamma function. After normalisation of eqn. C.23, we see that $q(Z_a)$ has the functional form of a $\mathscr{M}(\boldsymbol{Z}_a; 1, \boldsymbol{Q}_a = \{Q_{a1},\ldots,Q_{aK}\})$ Multinomial distribution. □

Proof of **Proposition 11** in section 4:

Proof. The terms of the upper bound F dependent on $q(\boldsymbol{R})$ are:

$$F[q(\boldsymbol{R})] = -\mathbb{E}_{\boldsymbol{Z},\boldsymbol{R},\boldsymbol{\Theta}}[\ln p(\boldsymbol{A},\boldsymbol{Z},\boldsymbol{R},\boldsymbol{\lambda},\gamma,\boldsymbol{\pi},\boldsymbol{\phi})]+\mathbb{E}_{\boldsymbol{R}}[\ln q(\boldsymbol{R})]+\text{const.} \tag{C.24}$$
$$= -\mathbb{E}_{\boldsymbol{Z},\boldsymbol{R},\boldsymbol{\lambda},\gamma}[\ln p(\boldsymbol{A}|\boldsymbol{Z},\boldsymbol{R},\boldsymbol{\lambda},\gamma)]-\mathbb{E}_{\boldsymbol{R},\phi}[\ln p(\boldsymbol{R}|\boldsymbol{\phi})]-\mathbb{E}_{\boldsymbol{Z},\boldsymbol{R},\boldsymbol{\pi}}[\ln p(\boldsymbol{Z}|\boldsymbol{\pi},\boldsymbol{R})]$$
$$+\mathbb{E}_{\boldsymbol{R}}[\ln q(\boldsymbol{R})]+\text{const.} \tag{C.25}$$
$$= \mathbb{E}_{\boldsymbol{R}}\Big[\sum_{q,l}^{K}\sum_{\substack{i,j\\ i\neq j}}^{N}\big(-R_iR_jQ_{iq}Q_{jl}A_{ij}\mathbb{E}[\ln\lambda_{ql}]+R_iR_jQ_{iq}Q_{jl}\mathbb{E}[\lambda_{ql}]\big)$$
$$-\left(\sum_{\substack{i,j\\ i\neq j}}^{N}(1-R_iR_j)A_{ij}\mathbb{E}[\ln\gamma]-(1-R_iR_j)\mathbb{E}[\gamma]\right)$$
$$+\sum_{i=1}^{N}-R_i\mathbb{E}[\ln\phi_i]-(1-R_i)\mathbb{E}[\ln(1-\phi_i)]$$
$$-\sum_{q=1}^{K}\sum_{i=1}^{N}R_iQ_{iq}\mathbb{E}[\ln\pi_q]\Big]+\mathbb{E}_{\boldsymbol{R}}\Big(\sum_{i=1}^{N}q(R_i)\ln q(R_i)\Big)+\text{const.}. \tag{C.26}$$

Variational optimisation of F with respect to $q(R_a)$ leads to:

$$\ln q(R_a) = R_a\Big[\left(\sum_{\substack{i=1\\ i\neq a}}^{N}\sum_{q,l}^{K}\rho_iQ_{iq}Q_{al}A_{ia}\mathbb{E}(\ln\lambda_{ql})-\rho_iQ_{iq}Q_{al}\mathbb{E}(\lambda_{ql})\right)$$
$$+\left(\sum_{\substack{i=1\\ i\neq a}}^{N}\sum_{q,l}^{K}\rho_iQ_{il}Q_{aq}A_{ai}\mathbb{E}(\ln\lambda_{ql})-\rho_iQ_{il}Q_{aq}\mathbb{E}(\lambda_{ql})\right)$$
$$-\sum_{\substack{i=1\\ i\neq a}}^{N}\rho_i(A_{ia}+A_{ai})\mathbb{E}(\ln\gamma)+2\mathbb{E}(\gamma)\sum_{\substack{i=1\\ i\neq a}}^{N}\rho_i+\mathbb{E}(\ln\phi_a)$$
$$-\mathbb{E}[\ln(1-\phi_a)]+\sum_{q=1}^{K}Q_{aq}\mathbb{E}(\ln\pi_q)\Big]+\text{const.}. \tag{C.27}$$

To see that, by eqn.(C.27), $q(R_a)$ has the functional form of the logarithm of a Bernoulli $\mathrm{Ber}(R_a;\rho_a)$ distribution, we use the following observation [74]:

$$\ln \mathrm{Ber}(R_a;\rho_a) = R_a \ln \rho_a + (1-R_a)\ln(1-\rho_a) = R_a \ln\left(\frac{\rho_a}{1-\rho_a}\right) + \text{const.}, \quad \text{(C.28)}$$

now we set $U_a = \ln\left(\frac{\rho_a}{1-\rho_a}\right)$ which leads to

$$\rho_a = \exp(U_a)(1-\rho_a) \Rightarrow \rho_a = \frac{1}{1+\exp(-U_a)}. \quad \text{(C.29)}$$

So, we set

$$\begin{aligned} U_a \equiv & \left(\sum_{\substack{i=1 \\ i\neq a}}^{N} \sum_{q,l}^{K} \rho_i Q_{iq} Q_{al} A_{ia} \mathbb{E}(\ln \lambda_{ql}) - \rho_i Q_{iq} Q_{al} \mathbb{E}(\lambda_{ql}) \right) \\ & + \left(\sum_{\substack{i=1 \\ i\neq a}}^{N} \sum_{q,l}^{K} \rho_i Q_{il} Q_{aq} A_{ai} \mathbb{E}(\ln \lambda_{ql}) - \rho_i Q_{il} Q_{aq} \mathbb{E}(\lambda_{ql}) \right) \\ & - \sum_{\substack{i=1 \\ i\neq a}}^{N} \rho_i (A_{ia} + A_{ai}) \mathbb{E}(\ln \gamma) + 2\mathbb{E}(\gamma) \sum_{\substack{i=1 \\ i\neq a}}^{N} \rho_i + \mathbb{E}(\ln \phi_a) \\ & - \mathbb{E}[\ln(1-\phi_a)] + \sum_{q=1}^{K} Q_{aq} \mathbb{E}(\ln \pi_q) \end{aligned} \quad \text{(C.30)}$$

which gives us together with eqn. (C.29) the optimal update, $\rho_a^\star$. □

Proposition 12. *The free energy after convergence (Integrated Likelihood variational bound) for the Poisson Stochastic Block Model with irrelevant vertices for K clusters, is given by*

$$\begin{aligned} F[q(\boldsymbol{Z},\boldsymbol{R},\boldsymbol{\Theta})] = & \sum_{i=1}^{N} \left(\ln\left(\frac{\Gamma(\zeta_i+\eta_i)}{\Gamma(\zeta_i)+\Gamma(\eta_i)}\right) - \ln\left(\frac{\Gamma(\zeta_i^0+\eta_i^0)}{\Gamma(\zeta_i^0)+\Gamma(\eta_i^0)}\right)\right) \\ & - \alpha_\gamma^0 \ln(\beta_\gamma^0) + \ln\Gamma(\alpha_\gamma^0) + \alpha_\gamma \ln(\beta_\gamma) - \ln\Gamma(\alpha_\gamma) \\ & - \sum_{q,l}^{K} \alpha_{q,l}^0 \ln(\beta_{ql}^0) + \ln\Gamma(\alpha_{ql}^0) + \sum_{q,l}^{K} \alpha_{ql} \ln(\beta_{ql}) - \ln\Gamma(\alpha_{ql}) \\ & - \ln\left(\Gamma\left(\sum_{q=1}^{K} \delta_q^0\right)\right) + \sum_{q=1}^{K} \ln(\Gamma(\delta_q^0)) + \ln\left(\Gamma\left(\sum_{q=1}^{K} \delta_q\right)\right) - \sum_{q=1}^{K} \ln(\Gamma(\delta_q)) \\ & + \sum_{q=1}^{K} \sum_{i=1}^{N} Q_{iq} \ln Q_{iq}. \end{aligned} \quad \text{(C.31)}$$

where $\boldsymbol{Q}$ is the cluster partition matrix, $\boldsymbol{A}$ the adjacency matrix, $\boldsymbol{R}$ the relevant vertices, $\boldsymbol{\Theta} = (\boldsymbol{\lambda}, \boldsymbol{\pi}, \gamma, \boldsymbol{\phi})$ the model parameters and $\boldsymbol{\vartheta} = (\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\delta}, \boldsymbol{\alpha}_\gamma, \boldsymbol{\beta}_\gamma, \zeta_i, \eta_i)$ the hyper parameters.

Proof.

$$F[q(\boldsymbol{Z},\boldsymbol{R},\boldsymbol{\Theta})] = -\sum_{\boldsymbol{Z},\boldsymbol{R}} \int \ln\left(\frac{p(\boldsymbol{A},\boldsymbol{Z},\boldsymbol{R},\boldsymbol{\Theta})}{q(\boldsymbol{Z},\boldsymbol{R},\boldsymbol{\Theta})}\right) q(\boldsymbol{Z},\boldsymbol{R},\boldsymbol{\Theta})\mathrm{d}\boldsymbol{\Theta} \tag{C.32}$$

$$\begin{aligned}
&= -\mathbb{E}_{\boldsymbol{Z},\boldsymbol{R},\boldsymbol{\Theta}}[\ln p(\boldsymbol{A}|\boldsymbol{Z},\boldsymbol{R},\boldsymbol{\lambda},\gamma)] - \mathbb{E}_{\boldsymbol{Z},\boldsymbol{R},\boldsymbol{\phi}}[\ln p(\boldsymbol{Z}|\boldsymbol{\pi},\boldsymbol{R})] - \mathbb{E}_{\boldsymbol{\lambda}}[\ln p(\boldsymbol{\lambda})] \\
&\quad - \mathbb{E}_{\gamma}[\ln p(\gamma)] - \mathbb{E}_{\boldsymbol{\pi}}[\ln p(\boldsymbol{\pi})] - \mathbb{E}_{\boldsymbol{R},\boldsymbol{\phi}}[\ln p(\boldsymbol{R}|\boldsymbol{\phi})] - \mathbb{E}_{\boldsymbol{\phi}}[\ln p(\boldsymbol{\phi})] \\
&\quad + \sum_{i=1}^{N} \mathbb{E}_{Z}[\ln q(Z_i)] + \mathbb{E}_{\boldsymbol{\pi}}[\ln q(\boldsymbol{\pi})] + \mathbb{E}_{\boldsymbol{\phi}}[\ln q(\boldsymbol{\phi})] \\
&\quad + \mathbb{E}_{\boldsymbol{\lambda}}[\ln q(\boldsymbol{\lambda})] + \mathbb{E}_{\gamma}[\ln q(\gamma)]
\end{aligned} \tag{C.33}$$

$$\begin{aligned}
&= \sum_{q,l}^{K}\sum_{\substack{i,j \\ i\neq j}}^{N} \left(-\rho_i\rho_j Q_{iq}Q_{jl}A_{ij}\left(\psi(\alpha_{ql}) - \ln(\beta_{ql})\right) + \rho_i\rho_j Q_{iq}Q_{jl}\left(\frac{\alpha_{ql}}{\beta_{ql}}\right)\right) \\
&\quad + \sum_{q,l}^{K} -(\alpha^0_{ql}-1)(\psi(\alpha_{ql}) - \ln(\beta_{ql})) + \ln\left(\Gamma(\alpha^0_{ql})\right) \\
&\quad + \sum_{q,l}^{K} \beta^0_{ql}\left(\frac{\alpha_{ql}}{\beta_{ql}}\right) - \alpha^0_{ql}\ln(\beta^0_{ql}) + \alpha_{ql}\ln(\beta_{ql}) - \ln\left(\Gamma(\alpha_{ql})\right) \\
&\quad + \sum_{q,l}^{K} (\alpha_{ql}-1)\left(\psi(\alpha_{ql}) - \ln(\beta_{ql})\right) - \beta_{ql}\left(\frac{\alpha_{ql}}{\beta_{ql}}\right) \\
&\quad - \left(\sum_{\substack{i,j \\ i\neq j}}^{N} (1-\rho_i\rho_j)A_{ij}\left(\psi(\alpha_\gamma) - \ln(\beta_\gamma)\right) - (1-\rho_i\rho_j)\frac{\alpha_\gamma}{\beta_\gamma}\right) \\
&\quad - \alpha^0_\gamma\ln(\beta^0_\gamma) + \ln\Gamma(\alpha^0_\gamma) - (\alpha^0_\gamma - 1)\left(\psi(\alpha_\gamma) - \ln(\beta_\gamma)\right) - \beta_\gamma\frac{\alpha_\gamma}{\beta_\gamma} \\
&\quad + \sum_{i=1}^{N} \Big(-\rho_i\left(\psi(\zeta_i) - \psi(\zeta_i+\eta_i)\right) - (1-\rho_i)\left(\psi(\eta_i) - \psi(\zeta_i+\eta_i)\right)\Big) \\
&\quad - \left(\sum_{i=1}^{N} (\zeta^0_i - 1)\left(\psi(\zeta_i) - \psi(\zeta_i+\eta_i)\right) + (\eta^0_i - 1)\left(\psi(\eta_i) - \psi(\zeta_i+\eta_i)\right)\right) \\
&\quad - \left(\sum_{i=1}^{N} (\zeta_i - 1)\left(\psi(\zeta_i) - \psi(\zeta_i+\eta_i)\right) + (\eta_i - 1)\left(\psi(\eta_i) - \psi(\zeta_i+\eta_i)\right)\right) \\
&\quad - \sum_{i=1}^{N} \left(\ln(\Gamma(\zeta^0_i+\eta^0_i)) - \ln(\Gamma(\zeta^0_i)) - \ln(\Gamma(\eta^0_i))\right) \\
&\quad - \sum_{i=1}^{N} \left(\ln(\Gamma(\zeta_i+\eta_i)) - \ln(\Gamma(\zeta_i)) - \ln(\Gamma(\eta_i))\right)
\end{aligned}$$

$$
\begin{aligned}
&- \sum_{q=1}^{K} \sum_{i=1}^{N} \rho_i Q_{iq} \big(\psi(\delta_q) - \psi \left(\sum_{l=1}^{K} \delta_q \right)) - \ln \left(\Gamma \left(\sum_{l=1}^{K} \delta_l^0 \right) \right) \\
&+ \sum_{q=1}^{K} \ln \big(\Gamma(\delta_q^0) \big) - \sum_{q=1}^{K} (\delta_q^0 - 1) \left(\psi(\delta_q) - \psi \left(\sum_{l=1}^{K} \delta_l \right) \right) \\
&+ \ln \left(\Gamma \left(\sum_{q=1}^{K} \delta_q \right) \right) - \sum_{q=1}^{K} \ln \big(\Gamma(\delta_q) \big) \\
&+ \sum_{q=1}^{K} (\delta_q - 1) \left(\psi(\delta_q) - \psi \left(\sum_{l=1}^{K} \delta_l \right) \right) + \sum_{i=1}^{N} \sum_{k=1}^{K} Q_{ik} \ln Q_{ik}. \qquad \text{(C.34)}
\end{aligned}
$$

We insert the update equations for the hyper parameters $\boldsymbol{\vartheta} = (\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\delta}, \boldsymbol{\alpha}_\gamma, \boldsymbol{\beta}_\gamma, \zeta_i, \eta_i)$ of Propositions 6, 7, 8 and 9 into eqn. (C.34). This yields eqn. (C.31). □

C.2 Relevance BlockVB algorithm: Filtering or Embedded Algorithm

There are two variants for the application of our Relevance BlockVB algorithm: We can choose the Filtering Relevance BlockVB algorithm where the update is determined between the cluster of irrelevant vertices and the expansion cluster of the relevant partition and the cluster of the irrelevant vertices.
The other option is the Embedded Relevance BlockVB algorithm, where the each vertex, regardless if it is relevant or not is assigned, to one of the relevant clusters or the cluster of the irrelevant vertices in the E-step.
In addition, we can use the Relevance BlockVB algorithm with the adaptive informative hyperparameters, we introduced in chapter 5, analogous to our BlockVB++ algorithm of section 5.1.

Input: Combined cluster partition matrix $\boldsymbol{Q}^{(c)}$, adjacency matrix $\boldsymbol{A}$, vector of relevant vertices $\boldsymbol{\rho}$, input for relevance hyperparameters: α_{ER} and β_{ER}, indices of active vertices I.

Initialisation: Set neutral [69] hyperparameters, we discussed in chapter 5, for the Gamma$(\lambda_{kl};\alpha_{kl},\beta_{kl})$ distributions: $\alpha^0_{kl}=\frac{1}{3}$; $\beta^0_{kl}=0.01\ \forall k,l\in\{1,\ldots,K\}$.
Set non informative priors of the Dirichlet prior distribution of cluster assignments: $\delta^0_k=1,\forall k\in\{1,\ldots,K\}$ [61, 50].
Set the informative relevance hyperparameters (RP), $(\alpha^0_\gamma,\beta^0_\gamma)$, (section 13.3.2) for the Gamma$(\gamma;\alpha_\gamma,\beta_\gamma)$ distributions of the irrelevant cluster:

$$\alpha^0_\gamma = 1; \tag{C.35}$$

$$\beta^0_\gamma = \frac{1}{\left(\frac{\alpha_{ER}}{\beta_{ER}}\right)}. \tag{C.36}$$

Set non informative hyperparameters [61] for the distribution of relevant and irrelevant vertices:

$$\zeta^0_i = \frac{1}{2}\forall i\in\{1,\ldots,N\}, \tag{C.37}$$

$$\eta^0_i = \frac{1}{2}\forall i\in\{1,\ldots,N\}. \tag{C.38}$$

Prepare M–Step: Calculate $\forall k,l$:

$$S^I_{\alpha_{kl}} = \sum_{\substack{i=1\\ i\neq j}}^{N}\sum_{\substack{j\in I\\ i\neq j}}\rho_i\rho_j Q_{ik}Q_{jl}A_{ij} + \sum_{\substack{i\in I\\ i\neq j}}\sum_{\substack{j\notin I\\ j=1}}^{N}\rho_i\rho_j Q_{ik}Q_{jl}A_{ij}, \tag{C.39}$$

$$S^I_{\beta_{kl}} = \sum_{\substack{i=1\\ i\neq j}}^{N}\sum_{\substack{j\in I\\ i\neq j}}\rho_i\rho_j Q_{ik}Q_{jl} + \sum_{\substack{i\in I\\ i\neq j}}\sum_{\substack{j\notin I\\ j=1}}^{N}\rho_i\rho_j Q_{ik}Q_{jl}, \tag{C.40}$$

$$S^I_{\delta_k} = \sum_{i\in I}\rho_i Q_{ik}, \tag{C.41}$$

for optimised updates of the parameters with respect to the subset of vertices $i\in I$ in

the M-step below.
Initialise the hyper parameters of the variational distributions:

$$S_{\alpha_{kl}} = \sum_{i \neq j}^{N} \rho_i \rho_j Q_{jl} Q_{ik} A_{ij}, \tag{C.42}$$

$$S_{\beta_{kl}} = \sum_{i \neq j}^{N} \rho_i \rho_j Q_{ik} Q_{jl}, \tag{C.43}$$

$$S_{\delta_k} = \sum_{i=1}^{N} \rho_i Q_{ik}, \tag{C.44}$$

and

$$\alpha_{kl} = S_{\alpha_{kl}} + \alpha^0_{kl}, \tag{C.45}$$

$$\beta_{kl} = S_{\beta_{kl}} + \beta^0_{kl}, \tag{C.46}$$

$$\delta_k = S_{\delta_k} + \delta^0_k. \tag{C.47}$$

Initialise parameters for the edge connections of the irrelevant vertices:

$$S_{\alpha_\gamma} = \sum_{\substack{i,j \\ i \neq j}}^{N} (1 - \rho_i \rho_j) A_{ij}, \tag{C.48}$$

$$\alpha_\gamma = S_{\alpha_\gamma} + \alpha^0_\gamma, \tag{C.49}$$

$$S_{\beta_\gamma} = \sum_{\substack{i,j \\ i \neq j}}^{N} (1 - \rho_i \rho_j), \tag{C.50}$$

$$\beta_\gamma = S_{\beta_\gamma} + \beta^0_\gamma. \tag{C.51}$$

Initialise parameters for the Beta$(\phi_i, \zeta_i, \eta_i)$ $\forall i$, distributions of relevant and irrelevant vertices $\forall i$:

$$\zeta_i = \rho_i + \alpha^0_i, \tag{C.52}$$

$$\eta_i = 1 - \rho_i + \beta^0_i. \tag{C.53}$$

Main Loop: Until convergence of F or maximum number of iterations.

Repeat: Update all active vertices, $\forall a \in I$.

E–Step for Q:

Embedded E-Step:
Set active vertex a to relevant $(\rho_a = 1)$ and calculate cluster assignment of a for all $v = \{1, \ldots, K\}$:

$$\begin{aligned} Q^\star_{av} \propto \exp\Big(& \sum_{\substack{i=1 \\ i \neq a}}^{N} \sum_{q,l}^{K} \rho_i \rho_a Q_{iq} A_{ai} \mathbb{E}(\ln \lambda_{vq}) + \sum_{\substack{i=1 \\ i \neq a}}^{N} \sum_{q,l}^{K} \rho_i \rho_a Q_{iq} A_{ia} \mathbb{E}(\ln \lambda_{qv}) \\ & - \sum_{\substack{i=1 \\ i \neq a}}^{N} \sum_{q=1}^{K} \rho_a \rho_i Q_{iq} \big(\mathbb{E}(\lambda_{vq}) + \mathbb{E}(\lambda_{qv}) \big) + \rho_a \mathbb{E}(\ln \pi_v) \Big). \end{aligned} \tag{C.54}$$

Normalise all updated matrix rows.

Filtering E-Step:
Alternative to Embedded E-step. Set matrix entry of expansion cluster of $\boldsymbol{Q}^c$ to 1.

E–Step for $\boldsymbol{\rho}$: Calculate relevance assignment of vertex a:

$$\begin{aligned} U_a = &\sum_{\substack{i=1\\ i\neq a}}^{N}\sum_{q,l}^{K}\rho_i Q_{iq}Q_{al}A_{ia}\mathbb{E}(\ln\lambda_{ql}) + \sum_{\substack{i=1\\ i\neq a}}^{N}\sum_{q,l}^{K}\rho_i Q_{il}Q_{aq}A_{ai}\mathbb{E}(\ln\lambda_{ql}) \\ &- \sum_{\substack{i=1\\ i\neq a}}^{N}\sum_{q,l}^{K}\rho_i Q_{iq}Q_{al}\mathbb{E}(\lambda_{ql}) - \sum_{\substack{i=1\\ i\neq a}}^{N}\sum_{q,l}^{K}\rho_i Q_{il}Q_{aq}\mathbb{E}(\lambda_{lq}) - \mathbb{E}(\ln\gamma)\sum_{\substack{i\neq a\\ i=1}}^{N}\rho_i(A_{ia}+A_{ai}) \\ &+ 2\mathbb{E}(\gamma)\sum_{\substack{i=1\\ i\neq a}}^{N}\rho_i + \mathbb{E}(\ln\theta_a) - \mathbb{E}(\ln(1-\theta_a)) + \sum_{q=1}^{K}Q_{aq}\mathbb{E}[\ln\pi_q] \end{aligned} \tag{C.55}$$

and

$$\rho_a^\star = \frac{1}{1+\exp(-U_a)}. \tag{C.56}$$

Round $\rho_a^\star$. Set

$$\rho_a^\star = \begin{cases} 1, & \text{if } \rho_a^\star \geq 0.5 \\ 0, & \text{else} \end{cases}. \tag{C.57}$$

Update the relevant entries of the partition matrix $\boldsymbol{Q}$: Set matrix row a to zero if $\rho_a^\star = 0$.

M–Step: We prepare the update of the parameters of the distributions of the relevant vertices by setting $S^{(old)}_{\alpha_{kl}} = S^I_{\alpha_{kl}}$, $S^{(old)}_{\beta_{kl}} = S^I_{\beta_{kl}}$ and $S^{(old)}_{\delta_k} = S^I_{\delta_k}$ $\forall k,l$.
Then we calculate $S^I_{\alpha_{kl}}$, $S^I_{\beta_{kl}}$ and $S^I_{\delta_k}$ $\forall k,l$ like in equations C.39, C.40 and C.41.
Now we can update the model parameters for all k,l restricted to the vertices in I, analogous to our BlockVB algorithm in section 4.2:

$$\alpha_{kl} = S_{\alpha_{kl}} - S^{(old)}_{\alpha_{kl}} + S^I_{\alpha_{kl}} + \alpha^0_{kl}, \tag{C.58}$$

$$\beta_{kl} = S_{\beta_{kl}} - S^{(old)}_{\beta_{kl}} + S^I_{\beta_{kl}} + \beta^0_{kl}, \tag{C.59}$$

$$\delta_k = S_{\delta_k} - S^{(old)}_{\delta_k} + S^I_{\delta_k} + \delta^0_k. \tag{C.60}$$

We update the parameters of the irrelevant vertices:

$$\alpha_\gamma = \sum_{\substack{i,j\\ i\neq j}}^{N}(1-\rho_i\rho_j)A_{ij} + \alpha^0_\gamma, \tag{C.61}$$

$$\beta_\gamma = \sum_{\substack{i,j\\ i\neq j}}^{N}(1-\rho_i\rho_j) + \beta^0_\gamma, \tag{C.62}$$

$$\zeta_i = \rho_i + \zeta^0_i,\ \forall i = \{1,\ldots,N\}, \tag{C.63}$$

$$\eta_i = 1-\rho_i + \eta^0_i,\ \forall i = \{1,\ldots,N\}. \tag{C.64}$$

Convergence: Check for the convergence of the variational upper bound, $F[q(\boldsymbol{Z},\boldsymbol{R},\boldsymbol{\Theta})]$:

$$\begin{aligned}
F[q(\boldsymbol{Z},\boldsymbol{R},\boldsymbol{\Theta})] = &\sum_{i=1}^{N} \ln\left(\frac{\Gamma(\zeta_i+\eta_i)}{\Gamma(\zeta_i)+\Gamma(\eta_i)}\right) - \ln\left(\frac{\Gamma(\zeta_i^0+\eta_i^0)}{\Gamma(\zeta_i^0)+\Gamma(\eta_i^0)}\right) \\
&- \alpha_\gamma^0 \ln(\beta_\gamma^0) + \ln\Gamma(\alpha_\gamma^0) + \alpha_\gamma \ln(\beta_\gamma) - \ln\Gamma(\alpha_\gamma) \\
&- \sum_{q,l}^{K} \alpha_{q,l}^0 \ln(\beta_{ql}^0) + \ln\Gamma(\alpha_{ql}^0) + \sum_{q,l}^{K} \alpha_{ql} \ln(\beta_{ql}) - \ln\Gamma(\alpha_{ql}) \\
&- \ln\left(\Gamma\left(\sum_{q=1}^{K} \delta_q^0\right)\right) + \sum_{q=1}^{K} \ln(\Gamma(\delta_q^0)) + \ln\left(\Gamma\left(\sum_{q=1}^{K} \delta_q\right)\right) - \sum_{q=1}^{K} \ln(\Gamma(\delta_q)) \\
&+ \sum_{q=1}^{K}\sum_{i=1}^{N} Q_{iq} \ln Q_{iq}. \qquad \text{(C.65)}
\end{aligned}$$

Repeat until convergence of the upper variational bound or until the maximum number of iterations is reached.

Selbständigkeitserklärung

Ich versichere hiermit, dass ich alle Hilfsmittel und Hilfen angegeben und auf dieser Grundlage die Arbeit selbständig verfasst habe. Ich habe die Arbeit nicht schon einmal in einem früheren Promotionsverfahren eingereicht.

Berlin, Februar 2017

(Christian Tobias Willenbockel)

Curriculum Vitae

Mein Lebenslauf wird aus Gründen des Datenschutzes in der elektronischen Fassung meiner Arbeit nicht veröffentlicht.

Bibliography

[1] S. Abe and N. Suzuki. Scale-free network of earthquakes. *Europhys. Lett.*, 65 (4):581–586, 2004. doi: 10.1209/epl/i2003-10108-1.

[2] S. Abe and N. Suzuki. Complex-network description of seismicity. *Nonlin. Processes in Geophys.*, 13:145–150, 2006. doi: 10.5194/npg-13-145-2006.

[3] S. Abe and N. Suzuki. Complex earthquake networks: Hierarchical organisation and assortative mixing. *Phys. Rev. E*, 74(026113), 2006. doi: 10.1103/PhysRevE.74.026113.

[4] S. Abe and N. Suzuki. Aftershocks in modern perspectives: Complex earthquake network, aging, and non-markovianity. *arXiv:1202.4394v1 [physics.geo-ph]*, 2012.

[5] S. Abe and N. Suzuki. Dynamical evolution of the community structure of complex earthquake network. *EPL*, 99:39001, 2012. doi: 10.1209/0295-5075/99/39001.

[6] S. Abe, D. Pastém, V. Munoz, and N. Suzuki. Universalities of earthquake-network characteristics. *Chinese Science Bulletin*, 56(34):3697–3701, 2011. doi: 10.1007/s11434-011-4767-6.

[7] C. Aicher. The weighted stochastic block model (wsbm) (ver. 1.2), 2014. URL `http://tuvalu.santafe.edu/~aaronc/wsbm/`.

[8] C. Aicher, A.Z. Jacobs, and A. Clauset. Adapting the stochastic block model to edge-weighted networks. *ICML Workshop on Structured Learning (SLG 2013)*, 2013.

[9] C. Aicher, A.Z. Jacobs, and A. Clauset. Learning latent block structure in weighted networks. *Journal of Complex Networks*, 3:221–248, 2015.

[10] E. M. Airoldi, D. M. Blei, S. E. Fienberg, and E. P. Xing. Mixed membership stochastic blockmodels. *J. Mach. Learn. Res.*, 9:1981–2014, 2008.

[11] H. Akaike. A new look at statistical model identification. *IEEE Transactions on Automatic Control*, 19:716–723, 1974.

[12] K.S. Ambrosen, K.J. Albers, T.B. Dyrby, M.N. Schmidt, and M. Mørup. Non-parametric bayesian clustering of structural whole brain connectivity in full image resolution. *PRNI*, 2014.

[13] D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithm*, pages 1027–1035, 2007.

[14] H. Attias. Inferring parameters and structure of latent variable models by variational bayes. In Laskey KB and Prade H, editors, *Uncertainty in artificial intelligence: proceedings of the fifteenth conference*, pages 21–30. Morgan Kaufmann, San Francisco, CA, 1999.

[15] H. Attias. A variational bayesian framework for graphical models. *Advances in neural information processing systems*, 12(1-2):209–215, 2000.

[16] M. J. Beal. *Variational Algorithms for approximate Bayesian inference*. PhD thesis, University College London, 17 Queen Square, London WC1N 3AR, 2003.

[17] H. Bengtsson, A. Jacobsen, and J. Riedy. R.matlab ver. 3.6.0: Read and write mat files and call matlab from within r, 2016. URL `https://github.com/HenrikBengtsson/R.matlab`.

[18] C. Biernacki, G. Celeux, and G. Govaert. Assessing a mixture model for clustering with the integrated completed likelihood. *IEEE Transactions Pattern Analysis and Machine Intelligence*, 7:719–25, 2000.

[19] C. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag, New York, 2006.

[20] D. Blackwell and J.B. MacQueen. Ferguson distributions via polya urn schemes. *The Annals of Statistics*, 1(2):353–355, 1973.

[21] V.D. Blondel, J.-L. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 10:10008–10020, 2008.

[22] S. Boyd and L. Vandenberghe. *Convex Optimisation*. Cambridge University Press, 2004.

[23] P. Bradley and U. Fayyad. Refining initial points for k-means clustering. In *Proceedings of the Fifteenth International Conference on Machine Learning ICML98*, pages 91–99, San Francisco, 1998. Morgan Kaufmann.

[24] J.E. Bronson, J. Fei, J.M. Hofman, R.L. Gonzalez, and C.H. Wiggins. Learning rates and states from biophysical time series: a bayesian approach to model selection and single-molecule fret data. *Biophysical journal*, 97(12):3196–3205, 2009.

[25] A. Celisse, J.-J. Daudin, and L. Pierre. Consistency of maximum-likelihood and variational estimators in the stochastic block model. *Electron. J. Statist.*, 6: 1847–1899, 2012.

[26] J. Chang and J.W. Fisher. Parallel sampling of dp mixture models using subclusters splits. In *NIPS*, 2013.

[27] A. Channarond, J.-J. Daudin, and S. Robin. Classification and estimation in the stochastic block model based on the empirical degrees. *arXiv:1110.6517v1 [math.ST]*, 2011.

[28] A. Clauset, C. Moore, and M.E.J. Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 453, 2008. doi: http://dx.doi.org/10.1038/nature06830.

[29] E. Côme and P. Latouche. Model selection and clustering in stochastic block models with the exact integrated complete data likelihood. *arXiv:1303.2962v2*, 2014.

[30] E. Côme and P. Latouche. Model selection and clustering in stochastic block models with the exact integrated complete data likelihood. *Statistical Modelling*, 15(6), 2015.

[31] A. Corduneanu and C. Bishop. Variational bayesian model selection for mixture distributions. In T. Richardson and T. Jaakkola, editors, *Artificial intelligence and statistics: proceedings of the eighth conference*, pages 27–34. Morgan Kaufmann, San Francisco, CA, 2001.

[32] D.R. Cutting, D.R. Karger, J.O. Pedersen, and J. W. Tukey. Scatter/gather: A cluster-based approach to browsing large document collections. In *SIGIR 92*, pages 318–329, 1992.

[33] D. B. Dahl. An improved merge-split sampler for conjugate dirichlet process mixture model. Technical Report 1086, Department of Statistics, University of Wisconsin-Madison, 2003.

[34] L. Danon, J. Duch, A. Diaz-Guilera, and A. Arenas. Comparing community structure identification. *J. Stat. Mech.*, P09008, 2005.

[35] J.-J. Daudin, F. Picard, and S. Robin. A mixture model for random graphs. *Statist. Comput.*, 18(2):173–183, 2008. ISSN 1573-1375. doi: 10.1007/s11222-007-9046-7.

[36] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *J. Roy. Stat. Soc. B*, 39:1–38, 1977.

[37] T. Fawcett. An introduction to roc analysis. *Pattern Recognition Letters*, 27: 861?874, 2006. doi: doi:10.1016/j.patrec.2005.10.010.

[38] R. P. Feynman. *Statistical Mechanics, A Set of Lectures*. W.A. Benjamin, Reading, MA, 1972.

[39] J. Futoma. Scalable inference algorithms for clustering large networks. Bachelor thesis, Department of Mathematics Dartmouth College, 2013.

[40] S. Gazal, J.-J. Daudin, and S. Robin. Accuracy of variational estimates for random graph mixture models. *Journal of Statistical Computation and Simulation*, 82(6):849–862, 2012.

[41] P.K. Gopalan and D.M. Blei. Efficient discovery of overlapping communities in massive networks. *Proceedings of the National Academy of Sciences*, 110 (36): 14534–14539, 2013.

[42] P.K. Gopalan, D. Mimno, S. Gerrish, M. Freedman, and D.M. Blei. Scalable inference of overlapping communities. *Advances in Neural Information Processing Systems*, pages 2249–2257, 2012.

[43] G. Govaert and M. Nadif. Block clustering with bernoulli mixture models: Comparison of different approaches. *Computational Statistics and Data Analysis*, 52: 3233–3245, 2008.

[44] Y. Guan, J.G. Dy, and M.I. Jordan. A unified probabilistic model for global and local unsupervised feature selection. *Proceedings of the* 28^{th} *International Conference on Machine Learning, Bellevue, WA, USA, 2011*, 2011.

[45] M. Hinne, M. Ekman, R.J. Janssen, T. Heskes, and M.A.J. van Gerven. Probabilistic clustering of the human connectome identifies communities and hubs. *PLOS ONE*, 10(1), 2015. doi: 10.1371/ journal.pone.0117179.

[46] P.D. Hoff. Subset clustering of binary sequences, with an application to genomic abnormality data. *Technical Report no. 456, Department of Statistics, University of Washington*, 2004.

[47] P.D. Hoff. Model-based subspace clustering. *Bayesian Analysis*, 1, Number 2: 321–344, 2006.

[48] M. Hoffman, D.M. Blei, C. Wang, and J. Paisley. Stochastic variational inference. *J Mach Learn Res*, 14:1303–1307, 2013.

[49] J. M. Hofman. vbmod. https://sourceforge.net/projects/vbmod/files/, 2008.

[50] J. M. Hofman and C. H. Wiggins. Bayesian approach to network modularity. *Phys. Rev. Lett.*, 100(258701), 2008.

[51] P. W. Holland and S. Leinhardt. An exponential family of probability distributions for directed graphs. *Journal of the American Statistical Association*, 76 (373):33–50, 1981.

[52] P. W. Holland, K. B. Laskey, and S. Leinhardt. Stochastic blockmodels: First steps. *Social Networks*, 5:109–137, 1983.

[53] K. Hutton, J. Woessner, and E. Hauksson. Earthquake monitoring in southern california for seventy years. *Bulletin of the Seismological Society of America*, 100(2):423–446, 2010. doi: 10.1785/0120090130.

[54] K. Ishiguro, N. Ueda, and H. Sawada. Subset infinite relational models. *Proceedings of the* 15^{th} *International Conference on Artificial Intelligence and Statistics (AISTATS) 2012, La Palma, Canary Islands*, 2012.

[55] T. Jaakkola. *Advanced Mean Field Methods: Theory and Practice*. MIT Press, Cambridge, MA, 2000.

[56] A.K. Jain and R.C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, 1988.

[57] S. Jain and R.M. Neal. A split-merge markov chain monte carlo procedure for the dirichlet process mixture model. *Technical Report No. 2003, Department of Statistics, University of Toronto*, 2000.

[58] S. Jain and R.M. Neal. A split-merge markov chain monte carlo procedure for the dirichlet process mixture model. *Journal of Computational and Graphical Statistics*, 13:158–182, 2004.

[59] S. Jain and R.M. Neal. Splitting and merging components of a nonconjugate dirichlet process mixture model. *Bayesian Analysis*, 2(3):445–472, 2007.

[60] H. Jeffreys. Some tests of significance, treated by the theory of probability. *Mathematical Proceedings of the Cambridge Philosophical Society*, 31:203–222, 1935.

[61] H. Jeffreys. An invariant form for the prior probability in estimation problems. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 186(1007):453–461, sep 1946.

[62] M.I. Jordan, Z. Ghahramani, T. Jaakkola, and L.K. Saul. An introduction to variational methods for graphical models. *Mach. Learn.*, 37:183–233, 1999.

[63] C. F. F. Karney. Algorithms for geodesics. *Journal of Geodesy*, 87(1): 43–55, 2013. ISSN 1432-1394. doi: 10.1007/s00190-012-0578-z. URL `http://dx.doi.org/10.1007/s00190-012-0578-z`.

[64] C.F.F. Karney. Geographiclib, version 1.22. https://sourceforge.net/projects/geographiclib/files/distrib/, 2012.

[65] B. Karrer and M.E.J. Newman. Stochastic block models and community structure in networks. *Phys. Rev. E*, 83:016107, Jan 2011. doi: 10.1103/PhyReveE.83.016107.

[66] R.E. Kass and A.E. Raftery. Bayes factors. *J. Am. Stat. Assoc.*, 90:773–795, 1995.

[67] C. Kemp, T.L. Griffiths, and J.B. Tenenbaum. Discovering latent classes in relational data. *(Technical report). MIT, Massachusetts,USA*, 2004.

[68] C. Kemp, J.B. Tenenbaum, T.L. Griffiths, T. Yamada, and N. Ueda. Learning systems of concepts with an infinite relational model. In *AAAI*, volume 3, page 5. 2006.

[69] J. Kerman. Neutral non informative and informative conjugate beta and gamma prior distributions. *Electronic Journal of Statistics*, 5:1450–1470, 2011. doi: 10.1214/11-EJS648.

[70] B. Larsen and C. Aone. Fast and effective text mining using linear-time document clustering. In *KDD 99 SanDiego California*, 1999.

[71] P. Latouche. *Modèles de graphes aléatoires à structure cachée pour l'analyse des réseaux*. PhD thesis, Université d'Evry-Val-D'Essonne Statistique et Génome / Exalead, 2010.

[72] P. Latouche, E. Birmelé, and C. Ambroise. Bayesian methods for graph clustering. *Advances in data handling and business intelligence*, pages 229–239, 2009. doi: 10.1007/978-3-642-01044-6.

[73] P. Latouche, E. Birmelé, and C. Ambroise. Variational bayesian inference and complexity control for stochastic block models. *Statistical Modelling*, 12:93, 2012.

[74] P. Latouche, E. Birmelé, and C. Ambroise. Model selection in overlapping stochastic block models. *Electon. J. Statist.*, 8(1):762–794, 2014.

[75] J.-B. Leger. Wmixnet: Software for clustering the nodes of binary and valued graphs using the stochastic block model. *arXiv:1402.3410v1*, 2014.

[76] J.-B. Leger. *Modelling the topology of ecological bipartite networks with statistical models for heterogeneous random graphs*. PhD thesis, Université Paris Diderot (Paris 7) Sorbonne Paris Cité, 2014.

[77] J.-B. Leger. Package 'blockmodels', 2015. URL `https://cran.r-project.org/web/packages/blockmodels/blockmodels.pdf`.

[78] J.-B. Leger. Inra, package 'blockmodels', latent and stochastic block model estimation by a 'v-em' algorithm (ver. 1.1.1), 2015. URL `https://cran.r-project.org/web/packages/blockmodels`.

[79] J.-B. Leger. Blockmodels: A r-package for estimating in latent block model and stochastic block model, with various probability functions, with or without covariates. *arXiv:1602.07587v1 [stat.Co]*, 2016.

[80] S.P. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, IT-28(2):129–137, 1982.

[81] B.A. Logsdon, G.E. Hoffman, and J.G. Mezey. Mouse obesity network reconstruction with a variational bayes algorithm to employ aggressive false positive control. *BMC Bioinformatics*, 2012.

[82] F. Lorrain and H. C. White. Structural equivalence of individuals in social networks. *Journal of Mathematical Sociology*, 1:49–80, 1971.

[83] J.B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297. University of California Press, 1967.

[84] M. Mariadassou, S. Robin, and C. Vacher. Uncovering latent structure in valued graphs: A variational approach. *Annals of Applied Statistics*, 4:715–742, 2010.

[85] A. McDaid, T. Murphy, and F.N.N. Hurley. Improved bayesian inference for the stochastic block model with application to large networks. *Computational Statistics and Data Analysis*, 60:12–31, 2013.

[86] C.A. McGrory and D.M. Titterington. Variational approximations in bayesian model selection for finite mixture distributions. *Comput. Statist. Data Anal.*, 51: 5352–5367, 2007.

[87] M. Middendorf, E. Ziv, and C.H. Wiggins. Inferring network mechanisms: the drosophila melanogaster protein interaction network. *Proceedings of the National Academy of Sciences of the United States of America*, 102(9):3192–3197, 2005.

[88] V. Miele, C. Ambroise, F. Picard, and H. Zanghi. Technical documentation about online estimation in the ermg model (documentation of mixnet package), 2007. URL `http://www.math-evry.cnrs.fr/logiciels/mixnet`.

[89] V. Miele, F. Picard, J.-J. Daudin, M. Mariadassou, and S. Robin. Technical documentation about estimation in the ermg model (documentation of mixnet package), 2007. URL `http://www.math-evry.cnrs.fr/logiciels/mixnet`.

[90] A. Mignan and J. Woessner. Estimating the magnitude of completeness for earthquake catalogs. *Community Online Resource for Statistical Seismicity Analysis*, 2012. doi: 10.5078/corssa-00180805. URL `http://www.corssa.org`.

[91] M. Morup. Bayesian community detection (software), 2012. URL `www2.imm.dtu.dk/pubdb/views/edoc_download.php/6147/zip/imm6147.zip`.

[92] M. Morup and M. Schmidt. Bayesian community detection. *Neural Computation*, 24(9):2434–2456, 2012.

[93] R.M. Neal and G.E. Hinton. A view of the em algorithm that justifies incremental, sparse, and other variants. In M.I. Jordan, editor, *Learning in graphical models*. MIT Press, Cambridge, MA, 1999.

[94] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Phys. Rev. E*, 69:026113, 2004. doi: 10.1103/PhysRevE.69.026113.

[95] M.E.J. Newman. Fast algorithm for detecting community structure in networks. *Physical Review Letter E*, 69, 2004. doi: 0066133.

[96] L. Nouedoui and P. Latouche. Bayesian non parametric inference of discrete valued networks. *21-th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN 2013), 2013, Bruges, Belgium*, pages 291–296, 2013.

[97] K. Nowicki and T.A.B. Snijders. Estimation and prediction for stochastic blockstructures. *Journal of the American Statistical Association*, 94:1077–1087, 2001.

[98] P. Erdös and A. Rényi. On random graphs. i. *Publicationes Mathemticae*, 6: 290–297, 1959.

[99] D. Pelleg and A. Moore. X-means: Extending k-means with efficient estimation of the number of clusters. In *Proceedings of the 17th International Conf. on Machine Learning*, pages 727–734, San Francisco, CA, 2000. Morgan Kaufmann.

[100] H. Robbins and S. Monro. A stochastic approximation method. *Ann Math Stat*, 22:400–407, 1951.

[101] S. Robin. Network analysis with the stochastic block model (using variational approximations). INRA / AggroParisTech, LIP6 February 2015 , Paris (lecture slides), 2015.

[102] K. Rohe, S. Chatterjee, and B. Yu. Spectral clustering and the high-dimensional stochastic blockmodel. *The Annals of Statistics*, 39(4):1878–1915, 2011.

[103] S. M. Savaresi, D. L. Boley, S. Bittanti, and G. Gazzaniga. Cluster selection in divisive clustering algorithms. *SDM conference*, pages 299–314, 2002.

[104] SCEDC. Southern california earthquake data center. caltech. dataset. http://service.scedc.caltech.edu/ftp/catalogs/SCSN/, 2015. doi:10.7909/C3WD3xH1.

[105] M.N. Schmidt and M. Mørup. Non-parametric bayesian modeling of complex networks. *arXiv:1312.5889v1 [stat.ML]*, 2013.

[106] G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6:461–464, 1978.

[107] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on PAMI*, 22:888–905, 8 2000.

[108] T. A. Snijders and K. Nowicki. Estimation and prediction for stochastic blockmodels for graphs with latent block structure. *Journal of Classification*, 14: 75–100, 1997.

[109] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. In *In KDD Workshop on Text Mining*, 2000.

[110] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. Technical Report 00-034, University of Minnesota, 2000.

[111] M. Svensén and C. Bishop. Robust bayesian mixture modelling. *Neurocomputing*, 64:235–52, 2004.

[112] N. X. Vinh, J. Epps, and J. Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research*, 11:2837–2854, 2010.

[113] W. Wang and S. Russell. A smart-dumb/dumb-smart algorithm for efficient split-merge mcmc. In *UAI 15*, pages 902–911, 2015.

[114] D.J. Watts and S.H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, 1998.

[115] P. Wessel et al. Gmt the generic mapping tools version 4. http://gmt.soest.hawaii.edu, 2003.

[116] C.T. Willenbockel and C. Schütte. A variational bayesian algorithm for clustering of large and complex networks. Technical Report 15-25, ZIB, Takustr.7, 14195 Berlin, 2015.

[117] C.T. Willenbockel and C. Schütte. Variational bayesian inference and model selection for the stochastic block model with irrelevant vertices. Technical Report 16-01, ZIB, Takustr.7, 14195 Berlin, 2016.

[118] C.T. Willenbockel and C. Schütte. An algorithm for clustering and link prediction of large and complex networks. *Phys. Rev. E (submitted, not accepted)*, EU11374, 2014.

[119] D. K. Wind and M. Mørup. Link prediction in weighted networks. In *2012 IEEE International Workshop on Machine Learning for Signal Processing*, pages 1–6, Sept 2012. doi: 10.1109/MLSP.2012.6349745.

[120] J. Wyse, N. Friel, and P. Latouche. Inferring structure in bipartite networks using the latent blockmodel and exact icl. *arxiv:1404.2911v3 [stat.CO]*, 2015.

[121] B. Yang and X. Zhao. On the scalable learning of stochastic blockmodel. *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[122] T. Yang, Y. Chi, S. Zhu, Y. Gong, and R. Jin. Detecting communities and their evolutions in dynamic social networks - a bayesian approach. *Mach Learn*, 82: 157–189, 2011. doi: 10.1007/10994-010-5214-7.

[123] H. Zanghi. *Approches modèles pour la structuration du Web vu comme un graphe*. PhD thesis, Université d'Evry-Val-D'Essonne Statistique et Génome / Exalead, 2010.

[124] H. Zanghi, C. Ambroise, and V. Miele. Fast online graph clustering via erdös-rényi mixture. *Pattern Recognition*, 41:3592–3599, 2008.

[125] H. Zanghi, F. Picard, V. Miele, and C. Ambroise. Strategies for online inference of model-based clustering in large and growing networks. *The Annals of Applied Statistics*, 4(2):687–714, 2010.

[126] E. Ziv, M. Middendorf, and C.H. Wiggins. Information-theoretic approach to network modularity. *Physical Review E*, 71(4):046117, 2005.