Universität des Saarlandes

Doctoral Thesis

Structural Building Blocks in Graph Data

Characterised by Hyperbolic Communities and Uncovered by Boolean Tensor Clustering

Saskia Metzler

Dissertation
zur Erlangung des Doktorgrades
der Ingenieurwissenschaften
der Fakultät für Mathematik und Informatik
der Universität des Saarlandes

Saarbrücken, 2020

Tag des Kolloquiums:	24. Februar 2021
Dekan der Fakultät:	Prof. Dr. Thomas Schuster
Mitglieder des Prüfungsausschusses:	Prof. Dr. Kurt Mehlhorn (Vorsitzender)
	Prof. Dr. Pauli Miettinen (Erstgutachter)
	Prof. Dr. Gerhard Weikum (Zweitgutachter)
	Prof. Dr. Stephan Günnemann (Drittgutachter)
	Dr. Koninika Pal (Beisitzer)

Abstract

Graph data nowadays easily become so large that it is infeasible to study the underlying structures manually. Thus, computational methods are needed to uncover large-scale structural information. In this thesis, we present methods to understand and summarise large networks.

We propose the *hyperbolic community model* to describe groups of more densely connected nodes within networks using very intuitive parameters. The model accounts for a frequent connectivity pattern in real data: a few community members are highly interconnected; most members mainly have ties to this core. Our model fits real data much better than previously-proposed models. Our corresponding random graph generator, HyGen, creates graphs with realistic intra-community structure.

Using the hyperbolic model, we conduct a large-scale study of the temporal evolution of communities on online question–answer sites. We observe that the user activity within a community is constant with respect to its size throughout its lifetime, and a small group of users is responsible for the majority of the social interactions.

We propose an approach for *Boolean tensor clustering*. This special tensor factorisation is restricted to binary data and assumes that one of the tensor directions has only non-overlapping factors. These assumptions – valid for many real-world data, in particular time-evolving networks – enable the use of bitwise operators and lift much of the computational complexity from the task.

Kurzfassung

Netzwerke sind heutzutage oft so groß und unübersichtlich, dass manuelle Analysen nicht reichen, um sie zu verstehen. Um zugrundeliegende Strukturen im großen Maßstab zu identifizieren, bedarf es computergestützter Methoden.

Unser Modell für *hyperbolische Gemeinschaften* beschreibt die innere Struktur eng verknüpfter Knotengruppen in Netzwerken mit sehr eingängigen Parametern. Es basiert auf der Beobachtung, dass oft ein kleiner Teil der Knoten einer Gruppe eng miteinander verknüpft ist und die Mehrheit der Gruppenmitglieder nur Verbindungen zu diesem Zentrum aufweist. Unser Modell bildet echte Daten besser ab als bisherige Modelle. Der entsprechende Zufallsgraphgenerator, HyGen, erzeugt Graphen mit realistischen innergemeinschaftlichen Strukturen.

Anhand unseres Modells analysieren wir die Bildung von Gemeinschaften in online Frage-und-Antwort-Netzwerken. Wir beobachten, dass die Aktivität der Mitglieder über die Zeit konstant ist, bezogen auf die Größe der jeweiligen Gemeinschaft. Außerdem ist stets eine kleine Gruppe von Mitgliedern verantwortlich für den Großteil der Aktivität.

Wir schlagen eine Methode für *Boolesches Tensor Clustering* vor. Diese spezielle Tensorfaktorisierung ist beschränkt auf binäre Daten und wir nehmen an, dass es entlang einer Richtung des Tensors keinen nennenswerten Überlapp der Faktoren gibt. Diese Annahmen ermöglichen die Nutzung von Bitoperationen, mindern den Rechenaufwand erheblich und passen gut zu dem, was in echten Daten zu beobachten ist.

Extended Abstract

People these days frequently interact with large amounts of network data and thereby contribute further to their growth. Online social networks, discussion forums, question–answer sites – the Internet offers numerous options for users to communicate with their peers through social media. This communication leaves massive traces of data and thereby opens opportunities to acquire a better understanding of social communities. The collected user interaction data, however, easily become so large that it is infeasible to study the underlying structures by manual inspection. Thus, computational methods are needed to uncover large-scale structural information.

In this thesis, we present methods to understand and summarise large networks. We propose a model for realistically representing communities in networks, study its use and its properties, develop a corresponding graph generator, and conduct a large-scale study of user interactions in question-answering forums on its basis. Furthermore, we propose an approach for Boolean tensor clustering. While this approach is for binary tensor data in general, we demonstrate its use in particular for graph data that can be presented as a binary tensor.

Binary information is the reoccurring scheme throughout this work. Our methods are all targeted at $\{0, 1\}$ data, meaning that a link between two objects is either present or not. Friend relations in online social networks, for instance, follow such a structure: there is a link between two persons if they know each other, and no link if they don't. While our methods are for networks in general, many ideas are intuitively best understood at the example of networks of people.

This dissertation makes the following research contributions:

Hyperbolic Community Model

Our *hyperbolic community model* accounts for the specific intra-community connectivity patterns frequently observed in real-world data: a small portion of the community members form the highly interconnected core, whereas the remaining members mainly have ties to the core. In other words, in the adjacency matrix, the nodes can be ordered in such a way that (almost) all edges in the community lie below a hyperbola. Our model fits to real-world data much better than traditional block models or previously-proposed hyperbolic models and can be expressed by means of very intuitive parameters, enabling us to summarise the shapes of the communities in graphs effectively.

Hyperbolic Random Graph Generator

Random graph generators are necessary tools for many network science applications. For example, the evaluation of graph analysis algorithms requires methods for generating realistic synthetic graphs. Typically random graph generators are generating graphs that satisfy certain global criteria, such as degree distribution or diameter. If the generated graph is to be used to evaluate community detection and mining algorithms, however, the generator must produce realistic community structure, as well. Our random graph generator, HyGen, is based on the hyperbolic community model. It is designed to preserve the community structure of real networks, especially the commonly observed hyperbolic intra-community connectivity structure. Our generator can also be used to accurately model time-evolving communities.

Study of Question-Answering Communities

We conduct a large-scale assessment of volunteer effort in online social communities. To that end, we study the temporal evolution of communities on online question–answer sites using the hyperbolic community model. The model parameters reflect the connectivity patterns within the network. Our primary observation is that the user activity within a community is constant with respect to its size throughout its lifetime, and a small group of users is responsible for the majority of the social interactions.

Boolean Tensor Clustering

We propose a scalable tensor factorisation approach, targeted at binary data. Similarly to analysing the adjacency matrix of a graph using a matrix factorisation, we can analyse the tensor by factorising it. While tensor factorisations in general are computationally hard problems, much of that hardness in case of Boolean tensor factorisations comes from the possibility of overlapping components. It can be lifted by imposing the constraint that factors must be non-overlapping in one of the tensor directions. This assumption is realistic for many real-world data. For instance, graphs – such as friendship networks – that evolve over time are naturally represented as binary tensors: the time direction cannot have overlaps, since time acts as a property of the interaction between linked entities. The factorisation along that direction then amounts to clustering. Our algorithm for Boolean tensor clustering achieves high scalability, high similarity towards the input, and good generalisation to unseen data with both synthetic and real-world data sets.

Acknowledgements

Although this chapter marks the begin of a large piece of writing, it corresponds to closing a chapter in life – a good moment to look back and thank those without whose support this thesis would have never been written.

First and foremost I would like to thank my supervisor, Pauli Miettinen, for his diligent guidance throughout my doctoral studies. While offering the necessary freedom to pursue my own ideas, I could always rely on his support and insightful feedback. Accommodating my advancement during the course of my doctoral studies, he shared his scientific expertise and also made sure to open the doors to the research community for me. In addition, he broadened my horizon not least by sharing facts about Finland and teaching how to finish a paper over night.

I would like to express my deep gratitude to Gerhard Weikum for giving me the opportunity to pursue my doctoral degree at the Databases and Information Systems group of the Max-Planck Institute for Informatics.

I would like to thank Stephan Günnemann for being an inspiring collaborator, and also for being a reviewer of my dissertation. Without his insights, this thesis would likely be on a different topic.

I would like to thank Aristides Gionis for insightful discussions during my research visit at Aalto University.

I am grateful for the privileged environment the Max-Planck Institute offers, which allowed me to fully commit myself to my doctoral studies. Of course, this environment would be nothing without great colleagues. I am especially thankful to my office mates, my lunch break companions, and the daily coffee machine meetup. Without those regular encounters, PhD life would have been truly different.

I would also like to recognise the university's Graduate School of Computer Science not only for its financial support in the beginning of my graduate studies, but also for creating such a stimulating environment by bringing together gifted people from all over the world. I have been given the chance to take part in this inspiring community. Learning together and from one another offered a unique opportunity to get to know foreign cultures and to grow my own identity.

Finally, I could not have completed this dissertation without the support of my friends and family. Especially, I want to thank Frederik Heber for being there for me throughout the entire journey of my dissertation.

Contents

1 Introduction

Large amounts of data can be collected without difficulty these days. They however easily become so large that manually exploring the collected data hardly reveals interesting information. To uncover underlying patterns and structures we thus need computational methods. *Data mining* is the sub-field of computer science that deals with the extraction of useful information from large data sets. To aid humans with the interpretation of large portions of data, data mining makes use of various concepts and techniques from statistics, machine learning and database systems. Typically employed techniques are data classification, prediction, data clustering, outlier analysis, association rule mining, and regression analysis [1]. Their common goal is to uncover the interesting, useful knowledge hidden in the data. But what exactly is *interesting* or *useful*? The exact definition of the objective is often the truly challenging part in data mining. What constitutes interestingness is usually data- and field-dependent, and, not uncommonly, highly subjective. In addition, the exact definition of interestingness might reveal only after knowing what is in the data. Prior to analysis, it might not even be obvious at which level we can expect to find the interesting pieces inside the data: Sometimes, it might be useful to regard the data as a whole; novel insight then might be gained through summarisation and explanation of the overall structure. In other situations, the search for useful patterns within a given data set is best implemented using a local approach [2].

[1]: Gopalan et al. (2009), *Data Mining: Techniques and Trends*

[2]: Mannila (2002), 'Local and Global Methods in Data Mining: Basic Techniques and Open Problems'

It would be presumptuous to claim that this thesis could help with the decision which method is preferable for a given set of data, or that it could be of any help to determine the most appropriate measure of interestingness, given a task. Instead we offer even more options to choose from. The methods we propose add to the repertoire of methods for handling large volumes of data and discovering the useful information therein. As always, it is left to the user to pick the right tool for each task.

We present two approaches, very dissimilar in their nature, yet both targeted at revealing patterns in *graph data*, and both acting rather on the global view of the data. Their common theme is to rely on the adjacency matrix representation of the graphs under study, which (in here) is of pure binary character. The adjacency matrix representation facilitates the use of linear algebra operations. Binary matrices, in particular, allow for the use of a Boolean algebra. Analysing data in the Boolean domain where $1 + 1 = 1$ enhances

interpretability – since the output is of the same type as the input – and allows for efficient computations, that scale to very large graphs.

The first approach is concerned with a characterisation of more densely connected structures within graphs, so-called *communities*. We devise a new model to concisely describe community structure within graphs and thereby allow for creating a conceivable summaries of the graphs under study. With these summaries, we can obtain an intuitive understanding of the community structures in large question–answer sites, such as Reddit and StackExchange, as we demonstrate in a large-scale study. Furthermore, we generate artificial random graphs with similar structural properties as their observed real counterparts.

The second approach also aids the summarisation of graphs, however in a different way. Its motivation originates from the endeavour to efficiently factorise binary tensor data. The computational effort of factorising the multi-way generalisations of matrices (*i.e.* tensors) can easily become overwhelming once its dimension grows. To alleviate the computational effort, we make the simplifying assumption that factors are non-overlapping in one of the directions. This takes away much of the hardness of the task and at the same time allows to interpret our approach as a clustering and summarisation technique for vertex-aligned graphs.

1.1 Thesis Structure

We start out with a basic introduction to networks, graphs, and related terminology in Chapter 2. After that, the thesis subdivides into two major parts.

The first part is concerned with hyperbolic communities and encompasses Chapters 3 to 9. We start with an introduction to hyperbolic communities in graphs and provide the necessary background of related work in Chapter 3. Chapters 4 and 5 offer theory and definitions. The former introduces the hyperbolic community model, the latter defines the random graph generator. The following three chapters are of experimental character. We test and evaluate the hyperbolic community model in Chapter 6. We examine the HyGen random graph generator in Chapter 7. And we present a large-scale study of communities in question–answer sites in Chapter 8. Finally, we end the first part with a discussion of future directions in Chapter 9.

The second part addresses Boolean tensor clustering in Chapters 10 to 13. We first introduce the problem and discuss preliminaries as well as related work in Chapter 10. We present the formal

definitions and theoretical results for Boolean tensor clustering in Chapter 11. We evaluate our method in Chapter 12; our experiments encompass synthetic and real-world data. Finally, in Chapter 13, we give concluding remarks and discuss resulting future research questions.

Before concluding the thesis, we outline an idea for combining the two lines of work in Chapter 14. A final summary in Chapter 15 marks the end of the thesis.

1.2 Contributions and Prior Publications

This thesis contributes novel ideas to understandably summarise large networks. Making use of the dualism between graphs and matrices, we discuss two separate approaches for uncovering patterns in graph data: (1) the description of communities by means of a hyperbolic model, and (2) the discovery of patterns using Boolean tensor clustering. As it is common practice in Computer Science, the contents of this thesis are largely based on previously published work by the author.

In the following, we list the covered prior publications in the order they occur in the thesis. Alongside, we summarise their scientific contributions. We outline which parts of each work are attributed to the author. And we depict how the works are incorporated in this thesis.

Hyperbolic community model. Commonly, communities are modelled as non-overlapping blocks of even density. Real graphs however often show overlaps between communities, and uneven density within them. We propose the *hyperbolic community model* which accounts for the specific intra-community connectivity patterns:

Saskia Metzler, Stephan Günnemann, and Pauli Miettinen. 'Hyperbolae are No Hyperbole: Modelling Communities That are Not Cliques'. In: *Proceedings of the 16th IEEE International Conference on Data Mining*. (ICDM'16 in Barcelona, Spain). Edited by Francesco Bonchi, Josep Domingo-Ferrer, Ricardo A. Baeza-Yates, Zhi-Hua Zhou, and Xindong Wu. Los Alamitos: IEEE Computer Society, 2016

Our major contributions are:

- We present three different but equivalent models for capturing non-uniform edge distributions inside communities.

- We show how to fit the models to the communities, and how to fit a model to a full graph composed of communities.
- We demonstrate that our approach explains real graphs much better than traditional quasi-clique based models and also improves the model of Araujo et al. [3].

[3]: Araujo et al. (2014), 'Beyond blocks: Hyperbolic community detection'

Saskia Metzler (SM), Stephan Günnemann (SG), and Pauli Miettinen (PM) jointly developed the idea. Every author had a special focus for the theory: SM deduced the constraints for the parameters. SG especially focussed on the log-likelihood formulation. PM studied the computational complexity. Implementation of the algorithm and experimentation were in responsibility of SM. The paper was written jointly by SM, SG, and PM.

With minor changes, we present the theoretic results in Chapter 4 and the experimental evaluation in Chapter 6.

HyGen. Random graph generators are frequently employed for the testing of community detection algorithms. Towards the improvement of such algorithms, random graph generators need to produce realistic community structure. We propose HyGen, the first random graph generator based on the hyperbolic model:

Saskia Metzler and Pauli Miettinen. 'Random Graph Generators for Hyperbolic Community Structures'. In: *Proceedings of the 7th International Conference on Complex Networks and Their Applications*. (COMPLEX NETWORKS'18 in Cambridge, UK). Edited by Luca Maria Aiello, Chantal Cherifi, Hocine Cherifi, Renaud Lambiotte, Pietro Lió, and Luis M. Rocha. Volume 812. Studies in Computational Intelligence. Cham, Switzerland: Springer Nature, 2019

Our main contributions are:

- We propose a generator for modular networks with realistic intra-community structures, using parameter distributions derived from observations in real graphs.
- We derive suggestions for the parameter distributions.
- We demonstrate the quality of the generator in extensive experiments.

We extend this line of work in:

Saskia Metzler and Pauli Miettinen. 'HyGen: Generating Random Graphs with Hyperbolic Communities'. In: *Applied Network Science* 4.53 (2019)

The additional contributions in this journal article are:

- We present an alternative formulation of the HyGen model as a graphon.
- We show that the graphon model is particularly suitable for modelling time-evolving communities.
- We show empirically that existing random graph generators are not suitable for generating hyperbolic community structure.

For both publications, SM and PM jointly came up with the idea. SM explored the options of existing algorithms to yield graphs with hyperbolic communities, reflected in the literature review as well as in the experiments. SM explored preserved properties regarding the graph structure. PM devised the graphon representation and conducted the related experiments. SM implemented the algorithm and conducted all other the experiments. SM and PM jointly wrote the paper.

With minor changes, we present most theoretic results in Chapter 5. We already discuss the graphon view of the hyperbolic model in Chapter 4 for improved continuity. With minor changes, we present the experimental evaluation in Chapter 7.

Volunteer effort in question-answering communities. Social science is concerned with the study of communities of people and their interactions. Using our hyperbolic community model, we study volunteer effort in online social communities at a large scale, examining the community structure on online question–answer sites:

Saskia Metzler, Stephan Günnemann, and Pauli Miettinen. 'Stability and Dynamics of Communities on Online Question-Answer Sites'. In: *Social Networks* 58 (2019)

The major contributions are:

- We propose a hyperbolic structure for online question–answer forums.
- We show that the ratio of active members per community remains constant over time.
- We observe that this constancy shows irrespective of community size and in all datasets, and contrasts what is usually assumed to happen in online social communities.

SM, SG, and PM jointly developed the idea for the study. SM reviewed related studies, conducted the experiments and analysis and took part of writing. SG and PM helped with the analysis and writing.

Largely unchanged, this study is presented in Chapter 8.

Chapter 3, being a common introduction to the so far mentioned works, builds on the lines of argumentation used in each of the introductions, also drawing on the reviews of related work from those publications.

Boolean tensor clustering. Tensor factorisations are used to uncover the structural building blocks of multiply-related data. Their computational complexity hampers scaling to large data. For *Boolean tensor clustering*, we restrict ourselves to binary data and assume one of the tensor directions has only non-overlapping factors – those assumptions are valid for many real-world data. The first restriction enables the use of bitwise operators, the latter assumption lifts much of the computational complexity from the task:

> Saskia Metzler and Pauli Miettinen. 'Clustering Boolean Tensors'. In: *Data Mining and Knowledge Discovery* 29.5 (2015)

We contribute:

- We present a new algorithm for the clustering problem, resulting from the imposed constraints.
- We investigate the consequences of our constraints for the computational complexity of Boolean tensor factorisations.
- We analyse our algorithm with the goal of maximising the similarity and argue that this is more meaningful than minimising the dissimilarity.
- We obtain a PTAS and an efficient 0.828-approximation algorithm for rank-1 binary factorisations.
- We demonstrate that our algorithm for Boolean tensor clustering achieves high scalability, high similarity, and good generalisation to unseen data with both synthetic and real-world data sets.

PM came up with the original idea of the paper. The theory and the algorithm were devised jointly by PM and SM. PM explored the complexity of the algorithm. SM implemented the algorithm and conducted the experiments. PM and SM wrote the paper jointly.

This article, with minor changes, constitutes the second part of this thesis, Chapters 10 to 13.

2 Graphs and Networks

Before we proceed to the details, this chapter is devoted to the basics. We discuss the foundations of network studies and the relevant elements of graph theory. Incidentally, we highlight the commonalities of the two major lines of work that constitute this thesis.

2.1 Foundations

Graphs are structures to model pairwise relations between objects. They are defined to consist of two sets, the set of vertices V and the set of edges E. We define a graph G as $G = (V, E)$ such that $E \subseteq V \times V$. Thus the elements of E are two-element subsets of V [4]. The number of vertices and edges in G we denote by $|V|$ and $|E|$, respectively.

[4]: Diestel (2017), *Graph Theory*

With this mathematical definition, nothing is said about the nature of the vertices or edges. In data mining, we often refer to graphs as *networks* with *nodes* and *links*. The subtle difference is that networks often refer to real systems while graphs solely denote the abstract structural concept [5]. Examples of networks are the Internet as a network of documents linked by URLs, society as a network of individuals, road maps that link cities, metabolic networks, or communication networks. We typically study the graph structure in conjunction with the network content, so the terms are mostly used interchangeably throughout this work.

[5]: Barabási et al. (2016), *Network Science*

2.1.1 Variations of Networks

For some networks, the direction of the links is important. In metabolic reaction networks, for instance, some reactions might be reversible, while others occur only in one direction. In communication networks, it might be of interest who called to whom, instead of just knowing the pairs of people in contact with each other. If all links of a network are directed, the network is called *directed*. If all links are undirected, the network is said to be *undirected*. The model we introduce subsequently (see Chapter 3) is targeted at undirected networks.

Some networks account for the strength of ties between nodes by equipping each link with a unique *weight*. An example would be a phone call network where links are weighted by the total time people talked to each other on the phone. In an email communication network, weights could be derived from how often people where in contact with each other within a certain time frame. While such information can be an interesting addition to the pure link structure, we focus on unweighted networks in this work. As we shall see shortly, unweighted networks may be represented through a binary matrix. One prime focus of our work is to explore the computational advantages of such a binary representation.

In some networks, nodes and edges are annotated. Annotations can supply additional information. In a network of people, this could be information such as the age or gender of a person. Annotations can also be used to define the *type of nodes*, for instance in a network that records who read which document. Nodes in that case either refer to people or documents and the network will thus have a *bipartite* nature, meaning that links only occur between nodes of different type. Formally, we say G is bipartite if the vertices of G partition into two classes U and V, and every edge has its ends in different classes.

Furthermore, edge annotations can define the *type of an interaction*. In the network of people and documents, the introduction of edge types, for instance, allows to distinguish whether someone commented, or authored, or just read a certain document. Since our work is focussed on exploiting advantages of the pure binary representation of networks, such additional information does not play a role for the subsequently introduced methods. Notice however, that it is often possible to turn edge annotations into separate graphs. For the example above, this would mean that we analyse three networks instead of one:[1] the network of comments, the network of reads, and the network of authorships.

1: Anticipating what will be discussed later in this chapter, this amounts to a three-way binary tensor.

2.1.2 Network Properties

Another fine difference between the notion of graphs and networks is their (perceived) size. Graphs often appear unaffected by their size. Graph-theoretical concepts, while valid for graphs of arbitrary size, can typically be understood at the example of neat small graphs. Networks, on the other hand, often seem large and unwieldy. To understand their contents, it is not enough to examine a small substitute, since their properties only show when studying the whole network. There exist numerous statistical measures that help to grasp the nature of a network under study by providing a characteristic summary. Examples of such measures [5] are

[5]: Barabási et al. (2016), *Network Science*

average path length — how many edges need to be traversed to reach from node i to j for any pair (i, j);

diameter — largest distance between any pair of nodes;

average degree — how many edges every vertex has;

clustering coefficient — to what extent the neighbours of a given node link to each other.

A further important characteristic is the *degree distribution*. By counting the number of edges incident to every node of a given network, we can empirically observe the degree distribution. The degree distribution provides the probability that a randomly selected node in the network has degree k. For real networks, the degree distribution can often be approximated by a power law function [5]. As we shall see in Chapter 4, empirical degree distributions play an important role for the model we propose.

Large networks are often *sparse*, meaning that the number of connections is less than what it could maximally be. While the exact definition of *less* is rarely ever mentioned, the networks we observe in practice have only a few percent of the potential edges present. From the coinciding observation that power law functions often well describe the degree distributions in real networks, we can gain more intuition about what sparseness refers to.

Networks whose degree distribution follows a power law are called *scale-free* networks [5]. Their nodes degrees are such that each fraction $P(k)$ of nodes with degree k obeys $P(k) \sim k^{-\gamma}$, where $\gamma \geq 0$ is the power-law exponent. Power law distributions are long-tailed and thus scale-free graphs are characterised by a small number of highly connected nodes and a large number of nodes with only few connections. This characteristic shape, as depicted in Figure 2.1, implies an upper bound for the number of possible edges: in a scale-free network, the number of nodes with degree of $O(1)$ increases linearly with $|V|$; two nodes, if $0 \leq \gamma \leq 2$, or vanishingly few nodes, otherwise, have degree $O(|V|)$ [6].[2]

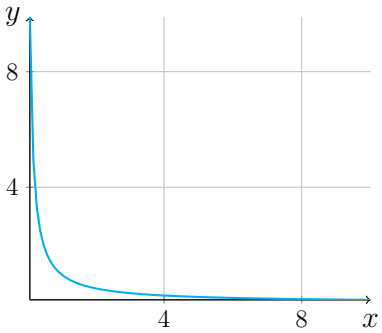


Figure 2.1: Plot of the power law function $f(x) = x^{-1}$.

[6]: Genio et al. (2011), 'All Scale-Free Networks Are Sparse'

2: Genio, Gross, and Bassler [6] find no guarantees for sparsity if $\gamma < 0$. But in that case, also the characteristic properties of scale-free networks vanish, hence we exclude them already by definition.

One way to represent graphs in a machine readable format is by means of an *adjacency matrix*. The adjacency matrix $\mathbf{A} = (a_{ij})$ of a graph G is the $|V|$-by-$|V|$ matrix where $a_{ij} = 1$ if there exists an edge between vertices i and j, and $a_{ij} = 0$ otherwise.[3] For undirected edges, we set $a_{ij} = 1$ and $a_{ji} = 1$. Thus, $\mathbf{A}$ is symmetric if G is undirected. The restriction of $\mathbf{A}$ to $\{0, 1\}$ facilitates Boolean algebra operations, and enables efficient storage of large graphs, as we shall see later.

3: We indicate matrices as bold uppercase letters. Element (i, j) of $\mathbf{M}$ is denoted as m_{ij}. See Sections 4.1 and 11.1, respectively, for more details regarding notation.

If a graph G is bipartite, the adjacency matrix can be expressed in a special way: Assuming we first write all nodes of class U and then of class V, the adjacency matrix $\mathbf{A}$ has two all-zero blocks along the diagonal since there cannot be any links between nodes of the same

type. The off-diagonal block $\mathbf{B}$ of size $|U|$-by-$|V|$ is the *biadjacency matrix*. With $\mathbf{B}$, we obtain a non-redundant representation of the graph, since $\mathbf{A} = \left(\begin{smallmatrix} 0 & \mathbf{B} \\ \mathbf{B}^T & 0 \end{smallmatrix}\right)$.

2.1.3 Graph Patterns

Networks, the larger they are, the more complex it gets to characterise their nature as a whole. It is often advantageous to study their constituents, or patterns they are composed of. Besides network measures of the kind discussed above, which offer a global view by counting properties of individual nodes, we can increase the scale of granularity and explore the organisation of groups of nodes. Commonly, intermediate, or mesoscopic, structure refers to the organisation of nodes into *modules* or *communities* [7]. While classically, this means that groups of nodes are more tightly connected to each other than they are to the rest to of the network, also internal connectivity patterns within the modules may be observed. Examples of intermediate scale structures are

[7]: Mucha et al. (2010), 'Community structure in time-dependent, multi-scale, and multiplex networks'

cliques — a set of fully interconnected nodes;
bipartite cores — non-empty, non-intersecting set of nodes;[4]
star — bipartite core with a single node in one partition;[5]
chain — list of nodes with links from one node to the next.

4: Edges exist only *between* two subsets of nodes, not *within*.

5: Such a single node is called *hub*.

Such a vocabulary for subsets of nodes is especially useful for the summarisation of graphs from an information theoretic point of view [8]. For the present work, it is a major focus to realistically model communities including their internal structure, thereby helping to gain an intuition about the nature of large networks under study.

[8]: Koutra et al. (2014), 'VoG: Summarizing and Understanding Large Graphs'

2.1.4 Community Structure

Networks with *community structure* have (potentially) overlapping subsets of nodes that are more densely connected among each other than to the rest of the graph. Social networks, in particular, frequently exhibit community structure. Using friendship networks as an example, we can easily deduce the underlying reasons why people form communities: individuals might know each other from school, the same place of origin, their occupation, a common hobby, and so on. Clearly, communities might be overlapping: the same person can have ties to former schoolmates, to colleagues at work, and be part of a social circle of musicians who know each other from orchestra rehearsals. Each of the three mentioned groups forms communities of mutual friends. One person can participate in several of them.

Furthermore, their members might not be equally well interconnected: someone who just started at a new work place probably knows the immediate colleagues and the boss, but most likely has not yet connected to everybody else; if the company is large enough, the person might never connect to all the colleagues. People, like the boss of a company, that are known by (almost) everybody else in a community, we call the *core* of the community. The rest of the members in the community we refer to as *tail*.[6]

In this social context, the intuitive notion of communities is easy to grasp. Yet, the whole branch of *Social science* is devoted to the study of social communities and the relationships among individuals therein because their deeper understanding is all but simple. Community structure is also observed in other contexts, for instance in protein interaction networks [10] and transportation networks [11]. Formal definitions to describe communities, models for their emergence, and their detection within graphs must trade-off between simplicity and the degree of realism.

With a focus on community detection, a variety of methods have been introduced in the literature [12]. Even though the proposed approaches seem highly diverse, their vast amount have been either explicitly or implicitly aimed at detecting block-shaped areas of uniform density in the adjacency matrix. This includes prominent techniques such as stochastic block-models [13, 14], affiliation network models [15], the detection of quasi-cliques [16, 17], and cross-associations [18]. Araujo et al. [3] capture that the density within communities is not uniformly distributed – some nodes show stronger connectivity among its peers. Inspired by their work, we propose a more general community model that is simple and at the same time highly realistic, as we will see in Chapter 4.

2.1.5 Network Generation

Models that reproduce network properties aid our understanding of large and complex networks. They reflect important characteristics we deem typical for the networks under study. To produce models of real networks, we need to define a suitable *random network model*, referring to a probability distribution or a random process which generates the network.

There are two classic approaches to define a random network [5]: in Erdős–Rényi graphs, L randomly placed links connect N nodes [19]; Gilbert's model assumes that each pair out of N nodes is connected with probability p [20]. Either model has advantages for the calculation of network characteristics. Real networks, however, typically exhibit the *small-world property*, meaning that they have

6: For similar structures, Borgatti and Everett [9] coined the terms *core* and *periphery*. We deviate from this terminology since the model we propose (see Chapter 4) allows for more shape variations than the term periphery implies: we will see subsequently that *tails* may get progressively thinner; nodes in the *periphery* are assumed to be evenly connected to the core.

[9]: Borgatti et al. (1999), 'Models of core/periphery structures'

[10]: Girvan et al. (2002), 'Community structure in social and biological networks'

[11]: Leo et al. (2013), 'Community core detection in transportation networks'

[12]: Javed et al. (2018), 'Community detection in networks'

[13]: Nowicki et al. (2001), 'Estimation and prediction for stochastic block-structures'

[14]: Airoldi et al. (2008), 'Mixed Membership Stochastic Blockmodels'

[15]: Yang et al. (2012), 'Community-affiliation graph model for overlapping network community detection'

[16]: Günnemann et al. (2014), 'GAMer: a synthesis of subspace clustering and dense subgraph mining'

[17]: Boden et al. (2012), 'Mining coherent subgraphs in multi-layer graphs with edge labels'

[18]: Chakrabarti et al. (2004), 'Fully Automatic Cross-Associations'

[3]: Araujo et al. (2014), 'Beyond blocks: Hyperbolic community detection'

[5]: Barabási et al. (2016), *Network Science*

[19]: Erdős et al. (1959), 'On random graphs I'

[20]: Gilbert (1959), 'Random Graphs'

a much higher clustering coefficient than the aforementioned models produce and at the same time a small average shortest path length. The coexistence of these properties is addressed in a third well-known model, the Watts–Strogatz model [21]. With a focus on realistic degree distributions,[7] the Barabási–Albert model [22] incorporates further important principles observed in real networks: *growth* and *preferential attachment*. The latter means that well-connected nodes are more likely to receive further connections.

[21]: Watts et al. (1998), 'Collective dynamics of 'small-world' networks'

7: The Erdős–Rényi model and the Watts–Strogatz model do not exhibit the scale-free property.

[22]: Albert et al. (2002), 'Statistical mechanics of complex networks'

Capturing network growth helps to understand the processes that assemble a network. It allows for hypotheses how a certain system came into being, and also enables to make predictions on the future evolution of certain networks. Bringing together small-world and scale-free phenomena in a time evolution-based modelling approach, the Forest fire model [23] has gained considerable attention: contrasting previous assumptions, the Forest fire model suggests that real graphs have out-degrees that grow over time, while the diameters have a tendency to decrease.

[23]: Leskovec et al. (2005), 'Graphs over Time: Densification Laws, Shrinking Diameters and Possible Explanations'

All of these approaches focus on imitating global properties of the graphs, like its degree distribution, clustering coefficient, or effective diameter. In contrast, the random graph generator we propose in Chapter 5 generates modular random graphs with special focus on realistic intra-community structures. To that end, it uses parameter distributions derived from observations on real graphs. Models for the realistic generation of community structures are especially important for the validation of community detection algorithms: Testing community detection requires large amounts of training data with reliable labels. Manual and automated labelling both come with the drawback that the developed detection approach is compared to the approach used to generate the labels. With a realistic random graph generator which accounts for community structure, we can test algorithms on trustworthy ground-truth information.

2.1.6 Tensors from Graphs

The networks we often encounter are all but static. Time-evolving networks, for instance, record communication, social interaction, or traffic. Their particular growth patterns only show if we study the networks at various points in time. To that end, we need to represent them in a machine readable form. One option is to record their adjacency matrix at every point in time. Those matrices, stacked on top of each other, can be regarded as a *three-way tensor*. Tensors are multi-way generalisations of matrices. While every entry of a matrix is identified by two indices, indicating the respective row

and column, in a three-way tensor, we additionally need to specify the layer we refer to. Generally, tensors can be N-way, taking N indices to refer to a particular entry.[8]

8: We formally introduce tensor terminology in Chapter 10.

Representing a time-evolving graph as a tensor implies that we need to know its dimensionality beforehand. Resizing the whole tensor as new nodes join is computationally costly. This might be a drawback when analysing networks in real time. It is however unproblematic if we are interested in the past of a network up until a certain time point. The advantage of the tensor representation is that we can make full use of tensor methods for the analysis.

The tensor constructed from the time-evolving graph as described above has no cross links along the time dimension. Assuming non-overlappingness in one of the modes facilitates a scalable binary tensor factorisation approach: *Boolean tensor clustering (BTC)* applies clustering along one tensor dimension to derive a partitioning, and presents a common representative of each partition that captures the essence of its structure, as we shall see in Chapter 10. Of course, binary tensors not need to originate from time-evolving graphs. Another example would be the network of people and documents with typed edges discussed above. A more involved example are RDF graphs interpreted as a tensor [24].

[24]: Metzler et al. (2015), 'Join Size Estimation on Boolean Tensors of RDF Data'

Hyperbolic Communities in Graphs

Modelling Community Structure 3

This chapter marks the begin of the first part of this thesis. We give an introductory overview to prepare for the introduction of the hyperbolic model: We motivate the need for incorporating details about the community structure in network models. We explain our approach to modelling community structure in networks. And we place our contributions in the context of related work.

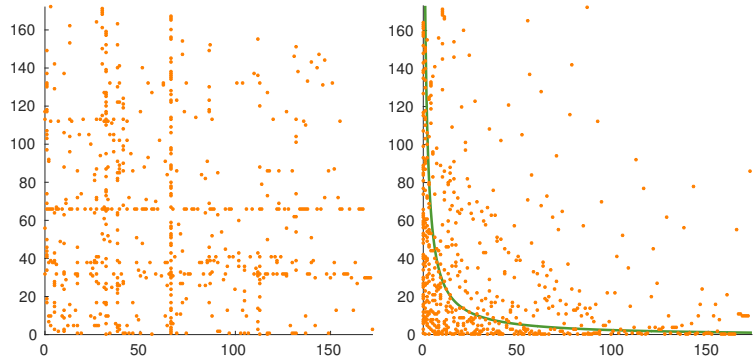
3.1 Our Contributions

The networks that typically come to mind when thinking of communities, are networks of people: friends, grouped by their home town, or the school they went to; people liking or following others; people calling each other on the phone. Also in this work, people are the entities in most data we study. It is probably a frequent choice for testing new graph methods or models, since (1) due to online communication, large amounts of data are freely available on the Internet and (2) understanding the meaning of links is very intuitive. At the same time human interaction data are so interesting – and so difficult to truly understand – that a whole branch of science is devoted to their study: social sciences are about studying communities and the relationships among individuals therein.

Our work adds options for large-scale analysis of social interaction data. Based on the observation that it is often an oversimplifying assumption that entities within communities have evenly strong connections to each other, we develop a model that describes the intra-community connectivity pattern in a realistic way: we account for the frequent observation that a community is dominated by a few highly inter-connected entities, and the remaining entities mainly have ties to those and hardly connect to one another (see Figure 3.1 for an illustration).

Primarily inspired by the observations of Araujo et al. [3], we introduce the *hyperbolic community model* to capture patterns of uneven connectivity within communities. As a special case, our model includes extreme structures such as a star, where one node is connected to everybody else but nobody has any further connection to each other, or a clique, where everybody is connected to each other. Our model furthermore includes power law connectivity,

[3]: Araujo et al. (2014), 'Beyond blocks: Hyperbolic community detection'



(a) Unordered. **(b)** Degree ordered.

Figure 3.1: An example of the kind of pattern we aim to model: the adjacency matrix of a community from the YouTube data, unordered and ordered by induced degree, with a hyperbolic model fitted (green line); while the plotted unordered adjacency matrix hardly reveals a pattern other than uniformly dense block, ordering by node degree reveals the shape of a hyperbola.

[9]: Borgatti et al. (1999), 'Models of core/periphery structures'

such as described by Araujo et al. [3]. Also the widely recognised core-periphery model [9] can be regarded as a special case of our hyperbolic community model. Unlike these preceding approaches, our model incorporates sparsity in its probabilistic formulation to account for imperfect data. Of course, even though we mostly discuss the model in the context of human interaction data here, it is valid for arbitrary graphs with community structure.

Towards the understanding (social) network organisation, we add another ingredient: we propose a random graph generator on the basis of the hyperbolic community model. The generation of realistic random graphs aids the synthesis of suitable community detection algorithms. Naturally, the quality of community detection algorithms is best tested on real-world data. This, however, requires significant amounts of testing data with reliable labels. Manual labelling is usually considered qualitatively the most adequate, yet potentially very subjective and hard to obtain in large quantities. Any automated community labelling procedure implies a comparison of the community detection algorithm under test to another procedure of detecting the communities. A favourable alternative to obtain large amounts of reliably labelled testing data is to use random graph generators that create graphs with similar properties as real-world templates. On the basis of our hyperbolic community model, we propose HyGen, a random graph generator that generates modular networks with realistic intra-community structures using parameter distributions derived from observations on real graphs.

In a hands-on study, we employ the hyperbolic community model for a broad survey of freely available question–answer data from the large popular sites `reddit.com`, `stackexchange.com`, and `healthboards.com`. This large-scale assessment of the volunteer effort in online social communities indicates that the active participation of only few community members within a group is a general organisational principle. We identify a unifying pattern present in all examined groups: *the amount of active members is a constant*

fraction of the entire community throughout its lifetime. Because of the social function of online communication [25], this result is relevant for social communities in general.

[25]: Wellman et al. (1996), 'Computer Networks as Social Networks: Collaborative Work, Telework, and Virtual Community'

In summary, in this part of the thesis, we propose a novel model for the description of communities in networks in Chapter 4 and a corresponding approach for network generation in Chapter 5. Besides experiments to validate the use of our model and our graph generator in Chapters 6 and 7, we present a large-scale case study of volunteer effort in online social networks in Chapter 8.

3.2 Related Work

Networks are frequently observed to have community structure, meaning that networks are composed of groups of densely connected nodes, sparsely connected to one another. Such modular structure occurs across various domains and has been regarded by a notable body of research. Online social networks might be the most prominent example, due to their presence in our everyday life. In addition, their data are comparatively easy to access and, to an extent, plausibility of the results can be verified using basic human intuition. The situation is very different when it comes to network data from biological systems, where data gathering and interpretation of the outcome are always a multidisciplinary endeavour: to name a few, metabolic networks [26], protein interaction networks [27], gene regulatory networks [28], and food-webs [29, 30] have been examined for their community structure. In a thorough overview, Javed et al. [12] identify further areas where modular networks have been studied: economics, e-commerce, academia and scientometrics, communication networks, healthcare, fraud and anomaly detection, link prediction, and source code design.

[26]: Ravasz et al. (2002), 'Hierarchical Organization of Modularity in Metabolic Networks'

[27]: Maslov et al. (2002), 'Specificity and Stability in Topology of Protein Networks'

[28]: Shen-Orr et al. (2002), 'Network motifs in the transcriptional regulation network of *Escherichia coli*'

[29]: Krause et al. (2003), 'Compartments revealed in food-web structure'

[30]: Guimerà et al. (2010), 'Origin of compartmentalization in food webs'

[12]: Javed et al. (2018), 'Community detection in networks'

In consequence of the numerous areas of applications, a versatile amount of methods and models exists in the literature. It would be beyond the scope of this thesis to provide a general comprehensive survey. The interested reader is referred to Javed et al. [12] for a meta-survey of relevant survey articles. In the following discussion, we focus on the three aspects we deem most relevant to our work:

- intra-community structure,
- overlap between communities,
- generation of networks with community structure.

While discussing those points, we of course also touch the area of community detection. The aspect of modelling the time evolution of communities will be considered in later, in the discussion of our results in Chapter 8.

3.2.1 Intra-Community Structure

In advance of the detection of communities in networks, it is important to define what constitutes a community. Commonly, communities are described as subsets of nodes, more densely connected to each other than to the rest of the network [31]. With this definition, nothing is said about the characteristic patterns of connectivity inside communities, and in fact, many approaches work with the underlying assumption that nodes within a community are evenly connected to each other. A community is then understood as a *block-shaped* area with *uniform density* in the adjacency matrix. This quasi-clique assumption is the basis of prominent techniques such as stochastic block-models [13, 14], affiliation network models [15], pattern-based techniques such as the detection of quasi-cliques [16, 17], and cross-associations [18]. In fact, all these techniques closely relate to the principle of Boolean matrix factorisation (BMF) [32]. The goal of BMF is to identify the binary factors that together explain given binary data. When applied to an adjacency matrix, the potentially overlapping factors correspond to sets of nodes. In other words, BMF aims to find overlapping *blocks* in binary matrices.

It has been observed, however, that communities in real networks typically display specific patterns of connectivity among their members. Especially in social networks, they show a pronounced core-tail structure: A small fraction of the community members have strong ties to each other and form the *core*. The majority of members only have ties to the core and not to each other; we denote them as *tail*. The observation of such structures has been described in diverse studies within the context of social networks: the relationships of elites and leaders in population subsystems [33, 34], the development of personal networks over time [35], volunteer effort and the civic core of society [36], communication among students [37], the relationships among Japanese macaques [38].

Borgatti and Everett [9] propose the first widely recognised formalisation of core-tail structures in networks. The *core/periphery* model classifies the nodes of a community in two subsets: a cohesive subgraph, where the participating nodes are fully interconnected, and peripheral nodes with connections to that subgraph but lacking cohesion.[1] For a continuous version of the model, Borgatti and Everett assign *core scores* to the nodes reflecting how deep a node lies inside the core. Generalising this idea, Rombach et al. [39] present a computational method for the identification of cores. While their reported performance would be insufficient for the analyses we present, it is potentially still an interesting direction of research is to explore the exact correspondence of this approach to ours.

[31]: Cherifi et al. (2019), 'On community structure in complex networks: challenges and opportunities'

[13]: Nowicki et al. (2001), 'Estimation and prediction for stochastic blockstructures'

[14]: Airoldi et al. (2008), 'Mixed Membership Stochastic Blockmodels'

[15]: Yang et al. (2012), 'Community-affiliation graph model for overlapping network community detection'

[16]: Günnemann et al. (2014), 'GAMer: a synthesis of subspace clustering and dense subgraph mining'

[17]: Boden et al. (2012), 'Mining coherent subgraphs in multi-layer graphs with edge labels'

[18]: Chakrabarti et al. (2004), 'Fully Automatic Cross-Associations'

[32]: Miettinen et al. (2008), 'The Discrete Basis Problem'

[33]: Laumann et al. (1976), *Networks of Collective Action: A Perspective on Community Influence Systems*

[34]: Alba et al. (1978), 'Elite Social Circles'

[35]: Morgan et al. (1997), 'The stability of core and peripheral networks over time'

[36]: Reed et al. (2001), 'The Civic Core in Canada: Disproportionality in Charitable Giving, Volunteering, and Civic Participation'

[37]: Panzarasa et al. (2009), 'Patterns and dynamics of users' behavior and interaction: Network analysis of an online community'

[38]: Corradino (1990), 'Proximity structure in a captive colony of Japanese monkeys (Macaca fuscata fuscata): An application of multidimensional scaling'

[9]: Borgatti et al. (1999), 'Models of core/periphery structures'

1: Unlike in our hyperbolic community model, all peripheral nodes are assumed to be evenly connected to the core.

[39]: Rombach et al. (2014), 'Core-Periphery Structure in Networks'

Our approach, the *hyperbolic model*, is inspired by an alternate line of work: Araujo et al. [3] observe uneven connectivity within communities in various data sets. They notice the reoccurring pattern that a few members of the community are highly interconnected and constitute the core. Most of the members only have ties to the core and thus form the tail. Contrasting the model of Borgatti and Everett, Araujo et al. assume that tails get progressively thinner, and the most peripheral nodes knows only a single member of the core. Such a shape in the degree-ordered adjacency matrix is well described by a power law function. In order to find and to describe those communities, Araujo et al. propose the `HyCom` algorithm.

[3]: Araujo et al. (2014), 'Beyond blocks: Hyperbolic community detection'

Our model is a generalisation of both above-described approaches. It captures the particular core-tail structure by means of simple parameters, indicating the size of the core and the size of the tail. As we shall see shortly, in Section 4.2, our model encompasses the model of Borgatti and Everett as a special case, and also encompasses power law-like connectivity as proposed by Araujo et al. It is suitably general to represent clique-like as well as star-like patterns of connectivity. Unlike `HyCom`, our probabilistic formulation enables the use with imperfect data. Araujo et al. assume a density of 100 per cent inside a community, violating the general property of sparsity. Our model, in contrast, allows varying density among the different communities, yielding clear benefits as the experimental analysis confirms.

A generalisation of our model is the concept of nested matrices [40]. Nested matrices have non-negative rounding rank 1 [41], suggesting that rounding rank decompositions could provide an approach for finding hyperbolic communities. We will return to this idea in the concluding discussion in Chapter 9.

[40]: Karaev et al. (2018), 'Logistic-Tropical Decompositions and Nested Subgraphs'

[41]: Neumann et al. (2016), 'What You Will Gain By Rounding: Theory and Algorithms for Rounding Rank'

3.2.2 Overlapping Communities

Our hyperbolic community model describes not only the structure of individual communities, but also their organisation within networks: we assume that networks may consist of multiple, potentially overlapping hyperbolic communities.

A simpler alternative is to assume communities to be disjoint. The most well-known model for networks with potentially connected, but non-overlapping communities is the *stochastic block model (SBM)* [42]. In its original formulation, communities are regarded as block-shaped areas in the adjacency matrix with uniform density. Among its numerous variations [43] are *mixed membership SBMs* [14], where a variational inference approach identifies overlapping, block-shaped communities. In an alternate line of work, Peixoto [44] presents *overlapping SBMs* which account

[42]: Holland et al. (1983), 'Stochastic blockmodels: First steps'

[43]: Lee et al. (2019), 'A review of stochastic block models and extensions for graph clustering'

[14]: Airoldi et al. (2008), 'Mixed Membership Stochastic Blockmodels'

[44]: Peixoto (2015), 'Model Selection and Hypothesis Testing for Large-Scale Network Models with Overlapping Groups'

for the co-occurrence of genuine mixed membership and single membership nodes within the same network. Further along this line, Peixoto [45] proposes a non-parametric generative process for *degree-corrected SBMs (DC-SBMs)*. DC-SBMs [46] allow for uneven intra-community densities. Yet, DC-SBMs differ from our hyperbolic model, as we will discuss subsequently in the context of graph generators. The DC-SBM formulation of Peixoto incorporates Bayesian inference and makes use of the minimum description length (MDL) principle to determine the best configuration of overlapping communities given a network. A distinction from our approach is that communities are determined in the sense of a *hierarchical partitioning* of the network. In a hierarchical partitioning, communities are embedded within other communities. Combining ideas from the work of Peixoto with our modelling approach could be an interesting direction of future research.

[45]: Peixoto (2017), 'Nonparametric Bayesian inference of the microcanonical stochastic block model'

[46]: Karrer et al. (2011), 'Stochastic blockmodels and community structure in networks'

Hierarchical organisation is considered an organisational principle of complex systems, but there is disagreement about its extent. For Palla et al. [47], the presence of communities in networks is a signature of the hierarchical nature. Lancichinetti, Fortunato, and Kertész [48] assume the co-presence of community structure and hierarchies, and propose a community detection approach for finding both overlapping communities and nested structure alongside each other. Alternatively, hierarchical organisation can simply be regarded as a special case of overlap between communities, as stressed by Yang and Leskovec [15]. This latter perspective best matches our modelling approach. Our model is valid also in case of subsumption of one community by another. The exploration of such structures, however, is not the focus of this work.[2]

[47]: Palla et al. (2005), 'Uncovering the overlapping community structure of complex networks in nature and society'

[48]: Lancichinetti et al. (2009), 'Detecting the overlapping and hierarchical community structure in complex networks'

[15]: Yang et al. (2012), 'Community-affiliation graph model for overlapping network community detection'

2: For focussing on hierarchical patterns, the algorithm presented in Section 4.5.2 would require an adaptation. In case of overlap, it uses a heuristic to ascribe nodes to either community for the log-likelihood computation. This behaviour might 'take away' all nodes from an inner community, rendering it seemingly empty.

Yang and Leskovec [15] furthermore explain non-uniform density inside communities as the result of overlap between communities. In their affiliation network model, nodes participating in multiple communities lead to areas of higher density, as edges become more likely due to the combined density of the overlapping tiles. Although one might assume the opposite, this model is not compatible with our scenario of hyperbolic communities: given a set of nodes to form a community, an affiliation network model generates edges following the same probability, leading to block-like structures in the adjacency matrix. In addition, the prior assumption of Yang and Leskovec seems inappropriate for the networks we study, since we observe non-uniform distributions of the edges in (often truly) non-overlapping communities.

For the purpose of graph compression, other graph patterns going beyond quasi-cliques have been considered. For Lim, Kang, and Faloutsos [49], graphs are a collection of hubs connecting to spokes. These hubs are recursively connected to super-hubs. Extending

[49]: Lim et al. (2014), 'SlashBurn: Graph Compression and Mining beyond Caveman Communities'

this idea, Koutra et al. [50] compress graphs by using patterns such as stars or bipartite cores. None of these works exploits the idea of hyperbolic community structure.

[50]: Koutra et al. (2015), 'Summarizing and understanding large graphs'

3.2.3 Network Generation

With an appropriate random network model, we may generate networks that mimic the structure of real graphs. The aim of many popular random graph generators is to model global properties of the graphs, such as degree distribution, clustering coefficient, or effective diameter. Hence, models such as Erdős–Rényi [19], Barabási–Albert [22], or the Forest fire model [23] do not generate community structure. Similarly, *graph expansion models* [51, 52] and *hyperbolic geometry models* [53] focus on global aspects of real-world graphs. *Kronecker graphs* [54] can be interpreted to have a community structure, but due to the recursive structure, the constructed communities have symmetries in size and shape not observed in real graphs.

[19]: Erdős et al. (1959), 'On random graphs I'

[22]: Albert et al. (2002), 'Statistical mechanics of complex networks'

[23]: Leskovec et al. (2005), 'Graphs over Time: Densification Laws, Shrinking Diameters and Possible Explanations'

[51]: Park et al. (2018), 'EvoGraph: An Effective and Efficient Graph Upscaling Method for Preserving Graph Properties'

[52]: Zhang et al. (2016), 'GSCALER: Synthetically Scaling A Given Graph'

[53]: Krioukov et al. (2010), 'Hyperbolic geometry of complex networks'

[54]: Leskovec et al. (2010), 'Kronecker Graphs: An Approach to Modeling Networks'

Further developing SBMs [42], specialised random graph generators have been proposed to generate graphs with a community structure. While the SBM is the most popular model to study community detection and clustering techniques [55], SBM has severe limitations, since it cannot incorporate variable inter-community degree distributions. To address this, DC-SBMs [46] incorporate an additional degree parameter for each vertex. This allows modelling uneven edge probabilities. As pointed out by Zhu, Yan, and Moore [56], DC-SBMs use the degrees of vertices as parameters, implying that the model cannot separate vertices based on degree even when that would be the correct partitioning. This limitation is overcome by *degree-generated SBMs* [56]. Degree-generated SBMs treat the expected degree of each vertex as generated from prior distributions, such as power laws, whose exponents vary from one community to another. Our model further differs from these, as we assume that intra- and inter-community edges have different probabilities.

[42]: Holland et al. (1983), 'Stochastic blockmodels: First steps'

[55]: Abbe (2018), 'Community Detection and Stochastic Block Models: Recent Developments'

[46]: Karrer et al. (2011), 'Stochastic blockmodels and community structure in networks'

[56]: Zhu et al. (2014), 'Oriented and degree-generated block models: generating and inferring communities with inhomogeneous degree distributions'

The *recursive matrix (R-MAT) generator* [57] is another popular model for generating random graphs with community structure. R-MAT generates graphs by recursively subdividing the adjacency matrix into four equally sized partitions and distributing the edges within the partitions according to partition-specific probabilities. This approach allows it to mimic the degree distributions of real-world graphs, but restricts the shape of the constructed communities.

[57]: Chakrabarti et al. (2004), 'R-MAT: A Recursive Model for Graph Mining'

A third popular model for testing community detection algorithms is the *Lancichinetti–Fortunato–Radicchi (LFR) benchmark* [58]. LFR is an extension of the *Girvan–Newman benchmark* [10]. Unlike SBM or

[58]: Lancichinetti et al. (2008), 'Benchmark graphs for testing community detection algorithms'

[10]: Girvan et al. (2002), 'Community structure in social and biological networks'

R-MAT, LFR can produce overlapping community structures and weighted and directed graphs. Compared to Girvan–Newman, LFR emphasises the heterogeneous distributions of node degrees and community size. Yet, LFR generates near-uniform intra-community degree distributions, violating the assumption of non-clique-like community structures, as we observe in Section 7.2.

[59]: Orman et al. (2013), 'Towards realistic artificial benchmark for community detection algorithms evaluation'

Orman, Labatut, and Cherifi [59] examine variations of LFR which achieve more realism with respect to transitivity and degree correlation in the generated graphs by choosing alternate random models for the initial step of the algorithm. In recent work, Fagnan et al. [60] propose a generalisation of LFR which follows the evolution patterns and characteristics of real networks. While the focus of our present work is the generation of individual communities with realistic structure therein, these works concentrate on modelling the overlap of multiple communities, and might therefore be a source of inspiration for future extensions of our model. An LFR variation into an alternate direction is introduced by Yang, Perotti, and Tessone [61]. Their goal is to represent hierarchical community structure within the generated networks, assuming that hierarchic organization is the prevalent principle of complex systems.

[60]: Fagnan et al. (2018), 'Modular Networks for Validating Community Detection Algorithms'

[61]: Yang et al. (2017), 'Hierarchical benchmark graphs for testing community detection algorithms'

To the best of our knowledge, no existing random graph generator is designed to model graphs consisting of hyperbolic communities. As many real-world graphs seem to follow a hyperbolic or core–tail model, we introduce a novel random graph generator to fill this gap. HyGen generates modular networks with realistic intra-community structures using parameter distributions derived from observations on real graphs.[3]

3: We introduce the random graph generator HyGen in Chapter 5.

The Hyperbolic Community Model

4

In this chapter, we introduce the hyperbolic community model for capturing non-uniform edge distributions. We discuss three equivalent formulations of the proposed model, each with a different intuition behind. We show their equivalence and discuss the advantages of the different parametrisations. Additionally, we present a graphon formulation of the model. We show how to fit the hyperbolic models to observed communities, and how to model an entire graph consisting of multiple communities.

4.1 Model Definition

Let our input be an undirected graph $G = (V, E)$ with n nodes and m edges. We will assign a number from $\{0, \dots, n-1\}$ to the vertices and use (u, v) to denote both a pair of vertices and the (potential) undirected edge between them. We will represent the graph using its *adjacency matrix* $\mathbf{A} = (a_{uv}) \in \{0,1\}^{n \times n}$.

Definition 4.1.1 *A* community *C is a tuple (V_C, π_C, Θ_C). The set $V_C \subseteq V$ contains the nodes in the community, and we write $n_C = |V_C|$. $\pi : V_C \to \{0, \dots, n_C - 1\}$ is a permutation of the nodes that maps the nodes of a community to a set of indices $\{0, \dots, n_C - 1\}$, and Θ_C is a set of parameters that defines the* shape *of the community.*

We assume that the permutation π_C orders the nodes in descending order according to their intra-community degree $\deg_C(v) = |\{u \in V_C : (v, u) \in E\}|$.

The key feature of our model is that not all edges between the nodes in V_C is necessarily part of the community – that assumption would make all of our communities quasi-cliques. Which edges are part of the community is defined using a binary decision function.

Definition 4.1.2 *The* model *of the community is given by the decision function*

$$f : \{0, \dots, n_C - 1\} \times \{0, \dots, n_C - 1\} \times \Theta_C \to \{0, 1\}.$$

The function f takes a pair of permuted node indices and the set of parameters for the community, and decides whether the edge between the nodes is part of the community.

Definition 4.1.3 *An edge (u, v) is a part of the community C if*

1. *$u \in V_C$ and $v \in V_C$; and*
2. *$f\left(\pi_C(u), \pi_C(v), \Theta_C\right) = 1$.*

For brevity, we will mostly omit writing the permutation. Instead, when it is clear from the context, we will denote vertices by their indices in the permutation; for instance, v, u, and w could be denoted by i, j, and h, with the meaning that $i = \pi_C(v)$, $j = \pi_C(u)$, and $h = \pi_C(w)$. Notice that the function f only gets the indices relative to the subgraph, not to the full graph, that is, to test a pair $(i, j) \in V_C \times V_C$, we need to compute $f\left(\pi_C(i), \pi_C(j), \Theta_C\right)$.

Every community is associated with two sets of edges: the *area* of the community, A_C, is defined as

$$A_C = \left\{(i, j) \in V_C \times V_C \colon f(i, j, \Theta_C) = 1\right\} \ , \tag{4.1}$$

and the *edges* of it, $E_C = E \cap A_C$. For notational convenience, we also define their complements (with respect to the community and the area, respectively):

$$\overline{A_C} = (V_C \times V_C) \setminus A_C \tag{4.2}$$

$$\overline{E_C} = A_C \setminus E \ . \tag{4.3}$$

Subsequently, we will discuss several definitions of the function f.

Probabilistic view. In practice the communities are rarely, if ever, exact. That is, some edges $(i, j) \in A_C$ are not in E_C, and some edges $(i, j) \in E$ that go between the nodes of the community are not in A_C. To model these imperfect communities, we consider a *probabilistic model* of the community. Given a community C, we assume that edges $(i, j) \in V_C \times V_C$ are drawn from a Bernoulli distribution, $a_{i,j} \sim \text{Bernoulli}(p_*)$, where $\mathbf{A} = (a_{i,j})$ is the adjacency matrix of the graph, and p_* is the *density* of the area that the edge belongs to. For a single community, we have two kinds of areas: the area of the community A_C and its complement $\overline{A_C}$. We denote the density of the area of the community by

$$d_C = \frac{|E_C|}{|A_C|} \ , \tag{4.4}$$

and the density of the area outside the community by

$$d_O = \frac{|E \cap \overline{A_C}|}{|\overline{A_C}|} \ . \tag{4.5}$$

These densities correspond to the maximum-likelihood solutions of the variables p_* for the edges that are inside or outside of the community. We can now consider the likelihood of the subgraph induced by the community $G|_{V_C}$ given the community C, $L(G|_{V_C} \mid C)$. The likelihood of an edge that is in community C is d_C; the likelihood of a pair (i, j) that is in the area of C but that is not an edge of G is $1 - d_C$; the likelihood of an edge that is not in the community is d_O; and the likelihood of a pair (i, j) that is not in the community's area and is not an edge of G is $1 - d_O$.

Definition 4.1.4 *The* log-likelihood $G|_{V_C}$ *of the subgraph induced by the community* $G|_{V_C}$ *given the community* C *is*

$$\begin{aligned}\log L(G|_{V_C} \mid C) = {} & |E_C| \log(d_C) + |\overline{E_C}| \log(1 - d_C) \\ & + |E \cap \overline{A_C}| \log(d_O) \\ & + |\overline{A_C} \setminus E| \log(1 - d_O) \ .\end{aligned}$$

4.1.1 Area Under a Hyperbola

`hyperbolic`(p, θ) is derived from the hyperbola equation.

From observations on real-world data, we notice that when the nodes of a community are ordered in the induced degree order, the edges lie under a hyperbolic curve, like shown in the example in Figure 3.1b. To define the hyperbola we identify the vertex indices of the community as points in x and y axes. We use i and j instead of x and y to emphasise this connection. We will only consider the area $[0, n_C - 1] \times [0, n_C - 1]$ from the non-negative quadrant, as that is where the values important to our community are. The equation for a (rectangular) hyperbola is

$$(i + p)(j + p) = \theta \ , \tag{4.6}$$

with the centre at $(-p, -p)$. Following the model, an edge (i, j) is considered to be in the community if $(i + p)(j + p) \leq \theta$. We call this model `hyperbolic`$(p, \theta)$ and write $(i, j) \in$ `hyperbolic`(p, θ) if $(i + p)(j + p) \leq \theta$.

From Figure 4.1 we can gain some intuition to the parameters p and θ: different values of p will yield different shapes of the gradient (the coloured background in Figure 4.1), while different values of θ will move the line away from the origin.

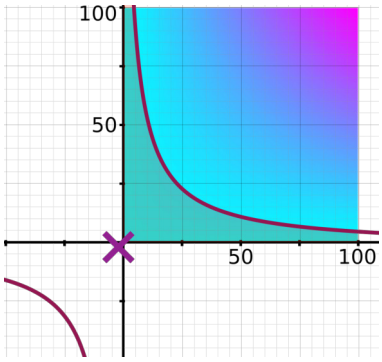


Figure 4.1: A hyperbola with $p = 2.06$ and $\theta = 673$ (dark red lines). The centre $(-p, -p)$ is marked with a cross. The colours in the background of the non-negative quadrant indicate the values of $(i+p)(j+p)$ for $i, j \in [0, 99]$, with higher values moving from cyan to magenta. The area of the community is the solid-coloured area under the curve.

Valid range of parameters. The values Equation 4.6 assigns to elements (i, j) attain their minimum at the centre $(-p, -p)$. To make sure that all elements $(i, j) \in \mathbb{N} \times \mathbb{N}$ that are under the curve Equation 4.6 are always in the community, we must bound p. A

simple boundary is to enforce that $p \geq 0$, though in the next section we will derive a more relaxed boundary. Other than this boundary, p and θ can be any values.

Definition 4.1.5 *The decision function for* `hyperbolic`(p, θ) *is*

$$f_{\texttt{hyperbolic}}(i, j, p, \theta) = \left[(i + p)(j + p) \leq \theta\right]$$

with the parameter set $\Theta = \{p, \theta\}$, *and* $p \geq 0$ *and* $\theta \in \mathbb{R}$.

We use the Iverson bracket notation, meaning that $[P] = 1$ if proposition P is true and $[P] = 0$ otherwise.

4.1.2 Fixed Points in Curve

`fixed`(γ, H) describes the community shape in an easy to interpret way.

The shape of the hyperbola is not easy to interpret from Equation 4.6, and hence it is not easy to say, by just looking at the parameters p and θ, whether the community is 'fat' or 'skinny'. To make the model parameters more interpretable, we can consider two points in the curve: the point at which it crosses the diagonal (*i.e.* when $i = j$), and the point at which the hyperbola exits the community (*i.e.* j for which $i = n_C$ or vice versa). We call the former γ and the latter H, and we can consider them as two values that define some p and θ such that

$$(\gamma + p)(\gamma + p) = \theta \tag{4.7}$$
$$(H + p)(n_C - 1 + p) = \theta\ . \tag{4.8}$$

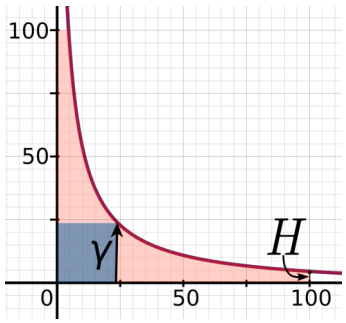


Figure 4.2: The parameter γ explains the size of the core of the community (dark-shaded box), while H explains the height of the tail at the end of the community.

To interpret the parameters, it is helpful to divide every community into two parts: *core* and *tail*. The core consists of nodes that form a densely connected (quasi-) clique, while the tail is formed by the least-connected members of the community that are mainly connected to the core. This is illustrated in Figure 4.2 where the core is shaded in dark blue and the tail in light red. The parameter γ is the size of the core (minus 1 to account for zero-based indexing) – the larger γ, the larger clique the community has – while the parameter H tells how 'fat' the tails are. A (quasi-) clique would have large γ and H, while a star would have $\gamma = H = 0$. We will call this model `fixed`(γ, H).

We can solve Equations 4.7 and 4.8 for p and θ to obtain an alternate decision function only dependant on γ, H, and n_C (the size of the community),

$$p = \frac{\gamma^2 - (n_C - 1)H}{(n_C - 1) + H - 2\gamma} \tag{4.9}$$

$$\theta = \frac{(\gamma - H)^2(\gamma - n_C - 1)^2}{(n_C - 1 + H - 2\gamma)^2} \,. \tag{4.10}$$

Definition 4.1.6 *The decision function for* `fixed`(γ, H) *is*

$$f_{\texttt{fixed}}(i, j, \gamma, H) = \left[\left(i + \frac{\gamma^2 - (n_C - 1)H}{n_C - 1 + H - 2\gamma}\right)\left(j + \frac{\gamma^2 - (n_C - 1)H}{n_C - 1 + H - 2\gamma}\right) \leq \frac{(\gamma - H)^2\left(\gamma - (n_C - 1)\right)^2}{(n_C - 1 + H - 2\gamma)^2}\right]$$

with the parameter set $\Theta = \{\gamma, H\}$ *and the community size* n_C.

Equivalence. Given Equations 4.7 and 4.8, it is hardly surprising that `fixed` is equivalent to `hyperbolic`. Similarly to Equations 4.9 and 4.10, we can work out the equations for γ and H given p and θ (and n_C), demonstrating the opposite direction of the equivalence relation:

$$\gamma = -p \pm \sqrt{\theta} \tag{4.11}$$

$$H = -\frac{p^2 + (n_C - 1)p - \theta}{n_C - 1 + p} \,. \tag{4.12}$$

Constraints of parameters. Not every γ and H yield valid communities in Equations 4.9 and 4.10, so we have to pay particular attention on the constraints of these parameters. Clearly both γ and H need to be non-negative. As the hyperbola is monotonic, it must be that $H \leq \gamma$, and as the hyperbola is convex, it must be that

$$\gamma \leq \frac{n_C - 1 + H}{2} \,. \tag{4.13}$$

Recall that in Section 4.1.1 we restricted p to be non-negative so that it cannot happen that $(0+p)(0+p) > \theta$ if $(i+p)(j+p) \leq \theta$ for some $(i, j) \in \mathbb{N} \times \mathbb{N}$. The motivation for this was that we want element $(0, 0)$ to be included in every non-empty community. But we can relax the constraint $p \geq 0$ to

$$\begin{aligned} p^2 \leq \theta \Leftrightarrow \quad & p^2 \leq (\gamma + p)^2 \\ \Leftrightarrow \quad & p \ \geq -\frac{\gamma}{2}\,, \end{aligned} \tag{4.14}$$

assuming $\gamma > 0$. Here, the first equivalence follows by substituting Equation 4.7 to θ. Notice that the p and γ in this inequality are bound together via Equation 4.9. This constraint also implies the above constraint that $H \leq \gamma$.

4.1.3 Line–Hyperbola Mixture

`mixture`(x, Σ) expresses the model as a mixture of index orderings.

Given the understanding of the previous two models, we now introduce a third equivalent model. Here, we consider a mixture of two restricted models: a simple hyperbola where an edge (i, j) is in the community if $i \cdot j \leq \Sigma'$ for some $\Sigma' \in \mathbb{R}$, and a simple linear model $i + j \leq \Sigma''$, with $\Sigma'' \in \mathbb{R}$. Notice that unlike above, here the hyperbola centre is fixed to the origin. Alone neither of these models is very expressive, but a mixture of these two is much more powerful: an edge (i, j) is in the community if

$$(1 - x)(i \cdot j) + x(i + j) \leq \Sigma \tag{4.15}$$

for some $x \in [0, 1]$ and $\Sigma \in \mathbb{R}$. Indeed, to allow even more flexibility, one can also consider a second mixture, which combines the hyperbola with a negative linear model

$$(1 - x)(i \cdot j) + x(-i - j) \leq \Sigma \tag{4.16}$$

for some $x \in [0, 1]$. Combining both of the above equations into a single model leads to our final definition: an edge (i, j) is in the community if for some $x \in [-1, 1]$ and $\Sigma \in \mathbb{R}$

$$(1 - |x|)(i \cdot j) + x(i + j) \leq \Sigma\,. \tag{4.17}$$

The meaning of the parameters here is slightly different to the model `hyperbolic`: the parameter Σ behaves similarly to θ in Equation 4.6, moving the line (or the hyperbola) further from the origin, while the mixture parameter x dictates how much the model looks like a line and how much like a hyperbola centred to the origin. We call this model `mixture`(x, Σ).

Definition 4.1.7 *The decision function for* `mixture`(x, Σ) *is*

$$f_{\texttt{mixture}}(i, j, x, \Sigma) = \left[(1 - |x|)(i \cdot j) + x(i + j) \leq \Sigma\right]$$

with the parameter set $\Theta = \{x, \Sigma\}$*, and* $x \in [-1, 1]$ *and* $\Sigma \in \mathbb{R}$*.*

Equivalence. We will now turn our attention to the equivalence between `hyperbolic` and `mixture`.

Proposition 4.1.1 *For any pair* (p, θ) *of valid parameters of the* `hyperbolic` *model, there exists a pair* (x, Σ) *of valid parameters of the* `mixture` *model that yields exactly the same community, and vice versa.*

Proof. We have

$$\begin{aligned} & (i + p)(j + p) \leq \theta \\ \Leftrightarrow\ & \frac{(i + j)p}{1 + |p|} + \frac{ij}{1 + |p|} \leq \frac{\theta - p^2}{1 + |p|} \\ \Leftrightarrow\ & (i + j)\frac{p}{1 + |p|} + ij\left(1 - \left|\frac{p}{1 + |p|}\right|\right) \leq \frac{\theta - p^2}{1 + |p|}\ , \end{aligned} \tag{4.18}$$

where the first equivalence is by expanding the left-hand side and re-arranging the terms and the second is by noting that

$$\frac{1}{1 + |p|} = \frac{1 + |p| - |p|}{1 + |p|} = 1 - \frac{|p|}{1 + |p|} = 1 - \left|\frac{p}{1 + |p|}\right|$$

If we now write $x = \frac{p}{1+|p|}$ and $\Sigma = \frac{\theta - p^2}{1+|p|}$, we get

$$(i + j)x + ij(1 - |x|) \leq \Sigma\ , \tag{4.19}$$

concluding the proof. □

4.1.4 Graphon Representation

A *graphon* [62] is a measurable function $g\colon [0, 1] \times [0, 1] \to [0, 1]$. Graphon g can be seen as a model for random graphs: To sample a graph G from graphon g, one first samples n vertices by sampling n points from the unit interval $[0, 1]$ uniformly at random (that is, $V \subset [0, 1]$). One then samples the edges so that G has edge (u, v) with probability $g(u, v)$. For undirected G, g has to be symmetric, that is, $g(u, v) = g(v, u)$, and the undirected edge between u and v is sampled as the directed edge (u, v).

[62]: Orbanz et al. (2015), 'Bayesian Models of Graphs, Arrays and Other Exchangeable Random Structures'

We now express the hyperbolic model as a graphon. The graphon is parametrised with four parameters, p_R, θ_R, d_C, and d_O. The parameters p_R and θ_R define the hyperbola in the same way as above: given two nodes $u, v \in [0, 1]$, the edge between them is part of the community[1] if $(u + p_R)(v + p_R) \leq \theta_R$. The parameters d_C and d_O are as defined in Equations 4.4 and 4.5, that is, they define the (expected) inside density and (expected) outside density. The graphon for one community is

1: An edge (u, v) is part of the community if u and v are in the intra-community area.

$$g = \begin{cases} d_C & \text{if } (u + p_R)(v + p_R) \leq \theta_R \\ d_O & \text{otherwise.} \end{cases} \tag{4.20}$$

Notice that the parameters p_R and θ_R are not the same p and θ as in the `hyperbolic` model. To understand the difference, it is easier to study the graphon equivalent of the `fixed` model. In the standard `fixed` model, the parameter γ indicates the size of the core; in the graphon version, it indicates the *relative size* of the core, that is, which *fraction* of the nodes belongs to the core. Similarly, the tail height parameter in the graphon model corresponds to the *relative thickness* of the tail. We denote these parameters as γ_R and H_R, respectively. They can be derived from the hyperbola analogously to the derivation in Section 4.1.2: We can set γ_R as the value u' where $(u' + p_R)(u' + p_R) = \theta_R$ and H_R as the value where $(u' + p_R)(1 + p_R) = \theta_R$. This gives

$$\gamma_R = -p_R \pm \sqrt{\theta_R} \tag{4.21}$$

$$H_R = -\frac{p_R^2 + p_R - \theta_R}{p_R + 1} \tag{4.22}$$

As we require that $\gamma_R, H_R \in [0, 1]$, if $p_R \geq 0$, then Equation 4.22 implies that it must be that $\theta_R \geq p_R(p_R + 1)$ and then Equation 4.21 takes from $\gamma_R = -p_R + \sqrt{\theta_R}$. Similar conditions can be easily derived for $p_R < 0$.

2: Analogously to the derivation of Equations 4.9 and 4.10.

Vice versa, we can also solve p_R and θ_R from γ_R and H_R:[2]

$$p_R = \frac{\gamma_R^2 - H_R}{-2\gamma_R + H_R + 1} \qquad \text{if } \gamma_R \neq 1 \text{ and } H_R \neq 1 \tag{4.23}$$

$$\theta_R = \frac{(\gamma_R - 1)^2(\gamma_R - H_R)^2}{(-2\gamma_R + H_R + 1)^2} \qquad \text{if } \gamma_R \neq 1 \text{ and } H_R \neq 1. \tag{4.24}$$

When $\gamma_R = H_R = 1$, we can set, for instance, $p_R = 0$ and $\theta_R = 1$.

4.2 Generalisation of Existing Models

An important consideration in our model is that we want to be able to generalise existing models. We now discuss how our model includes as a special case quasi-cliques, core-periphery models [9], and power-law models such as `HyCom` [3].

[9]: Borgatti et al. (1999), 'Models of core/periphery structures'

[3]: Araujo et al. (2014), 'Beyond blocks: Hyperbolic community detection'

Quasi-cliques. Clearly, quasi-cliques are a special case of our model. Using the `fixed` model, we can simply set H and γ to n_C.

Core-periphery. For the classic core-periphery model [9], it is assumed that all core members interconnect and peripheral nodes connect each to all core members. This resembles restricting $H = \gamma$ in our `fixed` model.

Power-law, represented by `HyCom`. Araujo et al. [3] define a community as follows: given $\alpha < 0$ and $\tau \in \mathbb{R}$, all edges (u, v) that fulfil

$$u^\alpha \cdot v^\alpha \geq \tau \tag{4.25}$$

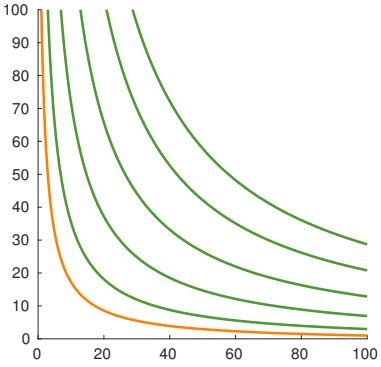
are part of the community. Note that in [3], one-based indexing is used, meaning that the indices of nodes u, v start with 1. Thus, using our zero-based indexing with respect to i, j, Equation 4.25 is equivalent to

$$\begin{aligned} &(i+1)^\alpha \cdot (j+1)^\alpha \geq \tau \\ \Leftrightarrow\ &0.5 \cdot (i \cdot j) + 0.5 \cdot (i + j) \leq \tau' \,. \end{aligned} \tag{4.26}$$

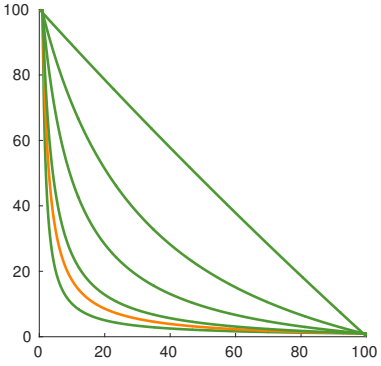
Here, we set $\tau' = 0.5 \cdot (\tau^{1/\alpha} - 1)$ and exploited that $\alpha < 0$.

Using our `mixture` model, it is now obvious that `HyCom` is a special case: it corresponds to `mixture`$(0.5, \tau')$. Indeed, while at first sight Equation 4.25 seems to have two degrees of freedom, it only has one. The parameter τ' (Σ in our notation) can be adapted, the parameter x is fixed to 0.5.

This restriction in the `HyCom` model limits the possible shapes of the communities significantly. Figure 4.3 provides an intuition of the influence of the parameters on the hyperbolic model. For the `HyCom` model, only the variation illustrated in Figure 4.3a is possible: fixing $x = 0.5$ limits the decision boundary to keep a certain curvature; as Σ varies, the boundary moves along the diagonal maintaining its shape. Figure 4.3b, shows the additional behaviour of the hyperbolic model under variation of x: the hyperbolic model attains different amounts of curvature of the decision boundary; even a straight diagonal line in the extreme case of $x = 1$.



(a) x fixed, Σ varies.



(b) Σ fixed, x varies.

Figure 4.3: Visualisation of the influence of Σ and x in the hyperbolic model. The highlighted line refers to $x = 0.5$, $\Sigma = 200$ in both plots.

Indeed, as we will show in the experiments in the next chapter, many real world datasets contain communities with x not close to 0.5 (see Figure 6.3). Thus, the additional freedom of the hyperbolic model, compared to `HyCom`, allows for a better representation. Notice also that the `HyCom` model does not offer an intuitive interpretation of the shape of the community as the exponent α can be switched to any other exponent by adjusting τ accordingly.

4.3 Full Graph Model

While the previous section focused on modelling individual communities, we now introduce a principle for describing a graph containing multiple hyperbolic communities.

When multiple communities are present, we might observe overlapping groups. When communities are modelled as quasi-cliques, there is only one type of overlap we need to consider: if the nodes of two communities overlap, so do their (implicit) edges. With our community models, however, we have to distinguish three types of overlapping behaviour: two communities C and D are

- *node-disjoint* if they do not share any nodes and hence $V_C \cap V_D = \emptyset$;
- *area-disjoint* (but *node-overlapping*) if they share nodes but no (implicit) edges, which is $V_C \cap V_D \neq \emptyset$ but $A_C \cap A_D = \emptyset$;
- *area-overlapping* (or *overlapping* for short) if also their (implicit) edges overlap, such that $A_C \cap A_D \neq \emptyset$.

Area-overlapping communities present a particular challenge to the modelling as we have to assign a likelihood to every (implicit) edge. In this work we assign each edge to at most one community and the likelihood of the edge is calculated using only that community.

For defining the quality of a set of communities, we refer to a probabilistic approach: we aim to find the set of models $\mathscr{C}$ leading to the highest *likelihood* of the input graph G. For this purpose, notice that in real graphs the communities rarely have full density (*i.e.* $|E_C| \neq |A_C|$), and that the density varies between the communities.

We use the model from Section 4.1 as the basis for modelling a community. That is, the density of a community C, d_C, is defined as in Equation 4.4, with every community having its own density. To define the outside density d_O, we have to consider not only the 'outside area' of a single community, but the whole area of the graph that does not belong to any community. If we let $A_{\mathscr{C}} = \cup_{C \in \mathscr{C}} A_C$ be the area that belongs to the communities and let $\overline{A_{\mathscr{C}}} = (V \times V) \setminus A_{\mathscr{C}}$ be its complement, then $d_O = |E \setminus A_{\mathscr{C}}| / |\overline{A_{\mathscr{C}}}|$.

We can now model the full graph similarly to how we modelled a single community to obtain the overall likelihood $L(G \mid \mathscr{C})$ of a graph G given the set of communities $\mathscr{C}$.

Definition 4.3.1 *Given a graph G and a collection of its communities $\mathscr{C}$, where every edge belongs to at most one community, the* log-likelihood $\log L(G \mid \mathscr{C})$ *is defined as*

$$\begin{aligned}\log L(G \mid \mathscr{C}) = \sum_{C \in \mathscr{C}} \Big(|E_C| \log(d_C) + |\overline{E_C}| \log(1 - d_C) \Big) \\ + |E \setminus A_{\mathscr{C}}| \log(d_O) \\ + \Big(|\overline{A_{\mathscr{C}}}| - |E \setminus A_{\mathscr{C}}| \Big) \log(1 - d_O) \, .\end{aligned}$$

4.4 Time Complexity

Now we turn our attention to the time complexity of the problems related to the modelling. Instead of dealing directly with the likelihood, in this section our target is to minimise the number of non-edges inside the communities while simultaneously maximising the number of non-edges outside.[3] This is a natural surrogate for the likelihood that allows us to avoid some issues in the analysis caused by the likelihood function.[4]

We will first consider problems involving only a single community, showing that finding the node sets for communities is hard in our model:

Proposition 4.4.1 *Given a graph $G = (V, E)$ and a pair of parameters (p, θ), it is* NP*-hard to find*

- *the largest set of nodes $V_C \subset V$ and a permutation $\pi_C \colon V_C \to \{0, \ldots, |V_C| - 1\}$ such that the area A_C defined by V_C, π_C, and* `hyperbolic`*(p, θ) is* exact*, that is $A_C = E_C$;*
- *the set of nodes $V_C \subset V$ and a permutation $\pi_C \colon V_C \to \{0, \ldots, |V_C| - 1\}$ such that $d_C \geq c$ for some given constant $c \in (0, 1)$ and d_O is minimised.*

Proof. These results follow from the fact that the clique is a special case of our model.

If (p, θ) are set so that they encode a clique, the first case is equivalent to the well-known NP-hard problem of finding the largest clique [63, Problem GT19], while the second is equivalent to the problem of finding the maximum c-quasi-clique, which is also NP-hard [64]. □

3: In other words, we aim to maximise the community density while minimising the outside-area density.

4: The likelihood model is oblivious to the 'inside' and 'outside': very sparse communities with dense outside area are also good models in likelihood's sense.

[63]: Garey et al. (1979), *Computers and intractability: A guide to the theory of NP-Completeness*

[64]: Pattillo et al. (2013), 'On the maximum quasi-clique problem'

On the other hand, if we are given the nodes (and their ordering), it is rather easy to find the best model for a single community, as the following proposition indicates:

Proposition 4.4.2 *Given a graph $G = (V, E)$, a set of vertices $V_C \subseteq V$, and a permutation $\pi_C \colon V_C \to \{0, \dots, |V_C| - 1\}$, finding the pair of parameters (p, θ) that maximises the function $d_C + (1 - d_O)$ (or the likelihood of Definition 4.1.4) is doable in time polynomial to $|V_C|$.*

Proof. We utilise here the (γ, H) parametrisation, which is equivalent to the (p, θ) parametrisation. As γ and H are non-negative integers bounded by $|V_C|$, there are at most $|V_C|^2$ different configurations of them. We can simply try every configuration exhaustively to find the optimal solution. □

We use Proposition 4.4.2 in the next section to design our algorithms. Importantly, we will show that we do not need to try all the $|V_C|^2$ different configurations.

Let us now turn our attention to the case where we are already given a collection $\mathcal{C}$ of communities (with fitted models), and we want to find a subcollection $\mathcal{S} \subseteq \mathcal{C}$ that minimises the number of edges in the outside area plus the number of non-edges inside the communities. That is, we want to minimise

$$|E \cap \overline{A_{\mathcal{S}}}| + |A_{\mathcal{S}} \setminus E| \,. \tag{4.27}$$

[65]: Miettinen (2015), 'Generalized Matrix Factorizations as a Unifying Framework for Pattern Set Mining: Complexity Beyond Blocks'

For these results, we use the general framework of Miettinen [65]. First note, that our communities are (symmetric) generalised rank-1 matrices in the sense of Miettinen: the functions f of our models define the outer product in Definition 1 of [65], while the adjacency matrix is the data matrix. For this problem, we only care about the union of the areas of the communities, and consequently, we take the element-wise disjunction of the matrices representing their areas. Propositions 6 and 10 of [65] directly provide the following results:

Proposition 4.4.3 *Given a graph $G = (V, E)$ and a collection $\mathcal{C}$ of communities of G,*

- *it is* NP*-hard to find the subcollection $\mathcal{S} \subseteq \mathcal{C}$ that minimises Equation 4.27;*
- *it is* NP*-hard to approximate Equation 4.27 to within a factor of $\Omega\left(2^{\log^{1-\varepsilon}|V|}\right)$ and quasi-*NP*-hard to approximate it within $\Omega\left(2^{(4\log|\mathcal{C}|)^{1-\varepsilon}}\right)$ for any $\varepsilon > 0$;*
- *Equation 4.27 can be approximated to within a factor of $2\sqrt{(|\mathcal{C}| + |V|)\log|V|}$ in polynomial time.*

The situation of Proposition 4.4.3 can easily arise as the consequence of the following simple idea of finding the communities: first, find many subset of nodes,[5] then fit the community models to them, and then select the final subset of communities from the potentially highly redundant set of communities. As Proposition 4.4.3 shows, the last step of this approach is computationally hard and hence we do not use it.

[5]: Subsets of nodes can be found by sampling or by enumerating all dense subgraphs.

4.5 Algorithms

Next we present an algorithm for fitting our model to a graph. We assume the input is the graph and an initial collection of sets of nodes that represent initial communities. These initial node sets can be found using any existing community-detection algorithm, for example `HyCom` [3]. We will first present the algorithm to fit the model to a single community, and then explain how to use that to fit the model for the full graph.

[3]: Araujo et al. (2014), 'Beyond blocks: Hyperbolic community detection'

4.5.1 Single Community

To model a single community, we use the `fixed` model with *integer* parameters γ and H. Given the intuition from Figure 4.2, this restriction is natural. We will also not lose too much, as the following proposition demonstrates:

Proposition 4.5.1 *Let $C = \texttt{fixed}(\gamma, H)$ be a community of n_C nodes defined by $\gamma \in \mathbb{R}$ and $H \in \mathbb{N}$, and let A_C be its area. Then there exists integer γ' such that if $D = \texttt{fixed}(\gamma', H)$ is the community defined by γ' and H, and A_D is the respective area, then $|A_C - A_D| \in \Theta\left(\gamma \ln(n_C)\right)$.*

In other words, the difference in area between integer and non-integer γ grows only linearly with γ and logarithmically with the number of nodes in the community. The proof of Proposition 4.5.1 is postponed to Appendix A.

Constraining ourselves to integer parameters would not alone solve much, as there still are $O(n_C^2)$ parameter configurations to study. Many of these configurations, however, can be pruned out as they would lead to infeasible communities and the pruned search space is small enough for exhaustive search.

To gain intuition on how much of the search space the constraints remove, let us consider Figure 4.4. The area of the plot is the area of all possible combinations of γ and H for some community size n_C. The yellow area can be ignored as in that area $H > \gamma$. But also both of the green areas can be ignored, as in those areas, the

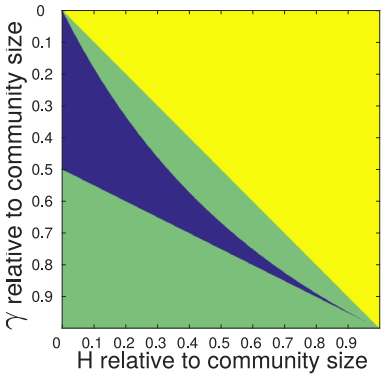


Figure 4.4: The blue area shows feasible values for parameters γ and H relative to a given community size. The green and yellow area are infeasible. The yellow part violates the trivial requirement of $H \leq \gamma$ and the green areas violate the condition $p \geq -\gamma/2$, where p is given by Equation 4.9.

constraint $p \geq -\gamma/2$ is violated (see Equation 4.14). This pruning significantly reduces the different parameter configurations we need to test in the exhaustive search.

Likelihood computation. Given a pair of parameters (γ, H), we need to evaluate its fit by computing the log-likelihood of the resulting model. According to the model proposed in Section 4.1, this requires to determine the area inside and outside the current community as well as the corresponding number of edges (and non-edges) in these areas. Obviously, testing each position in the community is not a practical solution since it would lead to a running time quadratic in the size of the community; instead we derive a solution which is linear in the number of edges.

The computation of the area can be done in time linear in the number of nodes by referring to the functional form of the hyperbola. That is, we evaluate

$$i = \frac{\theta}{j + p} - p \tag{4.28}$$

for each column j. Here we can compute p and θ from γ and H using Equations 4.9 and 4.10. Alternatively, we can approximate the area in constant time by taking the integral of this function from 0 to $n_C - 1$. Counting how many edges are inside the community requires a pass over the edges. Thus, this step dominates the time complexity.

It is easy to optimise this procedure further: First, we can compute the area faster by noticing that at the bottom we have a rectangle of size n_C-by-H. Second, when we test a succession of parameter values, we can re-use part of the information about the edges: by increasing the values of H or γ only edges previously outside the community need to be evaluated. All remaining edges will still be located within the community.

4.5.2 Full Graph

To obtain a model for a graph consisting of multiple potentially overlapping communities, we aim to optimise the log-likelihood $L(G \mid \mathscr{C})$ for the full graph, given in Definition 4.3.1. Our main problem is to determine how to deal with overlapping communities; indeed, if there are no node-overlapping communities, we can simply optimise every community separately using the above approach. If the communities do overlap, however, we do need to decide the order of the communities so that we can assign every edge to at most one community.

Algorithm 1: Algorithm to fit the `fixed` model to a graph.

Data: Undirected graph $G = (V, E)$, a collection of sets of nodes $\mathcal{V} = \{V_i \subset V\}$ describing the initial communities
Result: Ordered set of communities $\mathcal{C}$

1 **for** every set $V_i \in \mathcal{V}$ **do**
2 $C \leftarrow$ best model describing $G|_{V_i}$
3 $\mathcal{C} \leftarrow \mathcal{C} \cup \{C\}$
4 Order $\mathcal{C}$ based on the likelihoods of the communities
5 Compute the global outside density d_O
6 $\mathcal{F} \leftarrow \mathcal{C}$
7 **repeat**
8 $\mathcal{T} \leftarrow \mathcal{F}$
9 $\mathcal{F} \leftarrow \emptyset$
10 $M \leftarrow \emptyset$
11 **forall** $C \in \mathcal{T}$ in decreasing likelihood **do**
12 Update the model of C ignoring areas in M
13 $M \leftarrow M \cup A_C$
14 **if** the likelihood of C improved **then**
15 Update d_O
16 $\mathcal{F} \leftarrow \mathcal{F} \cup \{D \in \mathcal{C} : V_D \cap V_C \neq \emptyset\}$
17 Update the position of C in $\mathcal{C}$
18 **until** $\mathcal{F} = \emptyset$;
19 **return** $\mathcal{C}$

As computing the log-likelihood for the full graph for each possible order of the communities is infeasible, we optimise the log-likelihoods of each community individually following an alternating optimisation strategy. When optimising one community, we keep track of the area that was already covered by other communities, and ignore that area in the computations of the subsequent communities. In concordance with the log-likelihood we want to optimise for, we consider the global density for the whole outside community area (see Section 4.3) in the log-likelihood computation of the model, instead of just the local outside density, as in Section 4.1.

The algorithm, shown as Algorithm 1, comprises an initialisation phase and an update phase. In the initialisation phase of the algorithm (Lines 1 to 5), we compute an individual model for each community, leaving out those edges that have already been covered in a previous step by another community. Note that during this step, each community uses its individual outside density. Next, we order the obtained models by their log-likelihood starting with the best and we compute the global outside density d_O for the further updates.

Now that we have established an order of the communities, the alternating optimisation starts (Line 7): Each time, we select a community C_i and fit a new model to it – now, not only excluding edges already covered in previous communities, but also using the global outside density to determine the true log-likelihood for each community. After fitting the new model, we update the global outside density (Line 15) if the new model is different to the old one.

All communities that have node overlap with C_i are marked; due to the update of C_i also the parameters of the overlapping communities might change. Thus, we mark the communities that overlap with C_i for a re-update (Line 16). We iterate over this process of updating the community models until there are no more communities to be updated.

The output of this algorithm is a list of models for all communities, ordered by their log-likelihoods.

5 The Graph Generator Model

In this chapter, we introduce the random graph generator, HyGen. Based on the hyperbolic community model introduced in the previous chapter, HyGen is designed to model graphs consisting of one or multiple communities with realistic internal structures. We discuss properties of HyGen and introduce an alternate sampling approach, based on graphons, that is particularly suitable for time-evolving communities.

5.1 Model Definition

Random graph models are probability distributions over networks. Our random graph model is built upon the definition of hyperbolic communities, introduced in the previous chapter. Therefore, it accounts for core-tail structure that is commonly observed in real networks. We begin with the definition of the random graph generator for a single community, and extend it to a generator for graphs consisting of multiple hyperbolic communities. Finally, we discuss a graphon version of the model.

5.1.1 Single Community

The model for a single community is straight forward and wee express it via the `fixed` model (see Section 4.1.2). Assume, for now, that the size of the community, n_C, is given. To define the community `fixed`(γ, H), we need to have the two parameters, γ and H. In our model, we assume that the relative γ, $\gamma_R = \gamma/n_C$, follows the normal distribution with some mean μ and variance σ^2. Assuming we have sampled some $\gamma = n_C\gamma_R$, we will sample H by assuming that $H/\gamma \sim \text{Exponential}(\lambda)$.[1]

After we have sampled γ and H, we can generate a perfect community C using the `fixed`(γ, H) model. The community C will have no noise, that is, no edges between the nodes in the tail of the community and no missing edges within the community. As this is an unrealistic assumption, we will apply uniform noise to remove edges from the community and add edges between tail nodes. For this, we will need two more parameters, d_C and d_O, that describe the *densities* of the community and of the area outside

1: We will motivate the chosen distributions later using experiments with real-world data sets, see Section 7.3.

the community; a fraction of $1 - d_C$ edges within the community are removed and a fraction of d_O edges outside the community[2] is added.

2: That is between the tail nodes.

5.1.2 Multiple Communities

To generate multiple communities, we use an approach similar to SBMs, and generate individual communities independently of each other. For each community, we will draw the size n_C from distribution D_{size}. Our experiments[3] suggest using the generalised extreme value distribution as the distribution of the sizes, but the power law distribution, as used by Lancichinetti, Fortunato, and Radicchi [58], is a viable alternative.

3: We study a number of distributions for the model parameters in Section 7.3.

[58]: Lancichinetti et al. (2008), 'Benchmark graphs for testing community detection algorithms'

After the community size is sampled, we can sample γ and H as explained above. Do note that the way we sample them is independent of the community size, and hence we can use the same distributions for every community.

After we have sampled a noise-free community, we will remove some of its edges to obtain the desired 'inside' density d_C and plant the community to the graph. The planting happens so that the communities do not overlap, similar to SBMs, but we can permute the order of the nodes, so that the hyperbolic shape is not immediately obvious. After all communities have been planted, we apply the 'outside' noise to achieve the d_O density among the edges that are between nodes in different communities and between nodes that are in the tail of the same community. The full process is detailed in Algorithm 2.

Notice that this model assumes that the 'inside' noise is the same in all communities, and the 'outside' noise is uniform throughout the graph.[4] We also assume the size and shape of the communities to be uncorrelated.

4: These two types of noise can be very different, in general, $d_C \gg d_O$.

Algorithm 2: HyGen algorithm

Data: Distributions D_{size}, D_γ, D_H, densities d_C, d_O, number of communities k
Result: Random graph G

```
for i = 1 : k do
    Draw size s from D_size, γ from D_γ, and H from D_H
    Scale γ according to s, and H according to γ
    Make model fixed(γ, H)
    Select edges to discard uniformly at random to reach d_C
    Plant result into G
Apply noise d_O to the outside community area of G
return G
```

5.1.3 HyGen as a Graphon

The graphon model [62] is particularly useful for modelling time-evolving communities. Therefore, we incorporate graphons in the HyGen algorithm. Recapitulating from Section 4.1.4, graphons can be seen as a model for random graphs: to sample a graph G from graphon $g\colon [0,1]\times[0,1]\to[0,1]$, one first samples n vertices by sampling n points from the unit interval $[0,1]$ uniformly at random; one then samples the edges so that G has edge (u,v) with probability $g(u,v)$. In the HyGen algorithm, we generate a graphon for each community based on its parameters and replace the community generation inside the for-loop of Algorithm 2 with sampling the community from its graphon.

[62]: Orbanz et al. (2015), 'Bayesian Models of Graphs, Arrays and Other Exchangeable Random Structures'

For modelling time-evolving communities, unlike the standard HyGen algorithm, the graphon model makes it easy to adjust the size of the community while retaining some vertices: this can be done by simply randomly discarding some of the previously-selected nodes and potentially sampling new ones. Generating a full time-evolving graph would then amount to running Algorithm 2 for every time step, but instead of generating the communities from the scratch, we adjust their sizes based on the graphon model.

Compared to the standard HyGen model, the graphon model has more randomness. The parameter γ_R, for example, does not define the actual size of the core, but the expected size of the core. The actual size of the core is a binomially distributed random variable with parameters γ_R and s, where s is the size of the community. Similarly, the density parameters d_C and d_O are only expectations.

5.2 Time Complexity

Let us analyse the time complexity of Algorithm 2. For a community C, denote by E_C^p the edges of a 'perfect' community[5] and denote by E^p be the edges in a graph consisting only perfect communities. Let E^n be the set of edges noise *adds* to the graph (inter- and intra-community). Drawing the parameters and adjusting them is a constant-time operation, creating the model takes $O(n_C)$ time, and sampling the edges to discard from the community can be done in time $O\left(|E_C^p|\right)$ [66, p. 137]. Assuming we have k communities, the total time to generate a graph with no noise is $O\left(k(1+n_c+|E_C^p|)\right) = O(k+|V|+|E^p|)$.

5: We call a community *perfect* if it has no noise.

[66]: Knuth (1997), *The Art of Computer Programming Volume 2: Seminumerical Algorithms*

To add noise, we need to do sampling without replacement over a population of $O(|V|^2 - |E_p|)$ edges, taking essentially linear time. When the graph is sparse, $|E_p|$ and $|E^n|$ are small compared to $|V|^2$ and we can sample with replacement to obtain practically the same result, taking $O(|E^n|)$ time.

In total, the full running time of Algorithm 2 is $O(k + |V| + |E^p| + |E^n|)$, which is only slightly more than $O(|V| + |E|)$.

5.3 HyGen Graphs from Predefined Model

In the above, we described HyGen model using randomly sampled shape parameters. Normally, we would fit the hyperparameters of the distributions to some real data to obtain a realistic model for the random graphs, but in some cases, we might want to obtain specific community structures.

One specific task is to generate random graphs with power law connectivity within the communities.[6] That is easy to do via the `mixture` parametrisation, introduced in Section 4.1.3. Recall from Definition 4.1.7 that two parameters, $x \in [-1, 1]$ and $\Sigma \in \mathbb{R}$, define the shape via evaluating

$$(1 - |x|)(i \cdot j) + x(i + j) \leq \Sigma \, .$$

By setting $x = 0$, it becomes clear that the model reduces to a power law; the exponent of the power law is absorbed by the parameter Σ, as setting $\Sigma' = \Sigma^{1/-\alpha}$ yields[7]

$$(ij)^{-\alpha} \leq \Sigma' \, .$$

Hence we can sample power law communities by just sampling one parameter, $\Sigma \in (0, \infty)$.

A clique-like community is even more restricted than power law community, as we can set $\gamma = H = n_C$ to obtain a clique. With such fixed settings, our model reduces to a variation of SBM.

5.4 Theoretical Results

The HyGen model is designed to preserve important graph properties. In addition to the hyperbolic community structure, HyGen also preserves the *degree distribution* and the *clustering coefficient* of the graph. These two properties measure the connectivity of networks [67], and are often studied with social networks. In this

6: The `HyCom` model [3] is an example for power law connectivity.

[3]: Araujo et al. (2014), 'Beyond blocks: Hyperbolic community detection'

7: Assuming $\Sigma > 0$.

[67]: Aggarwal et al. (2010), *Managing and Mining Graph Data*

section, we present theoretical analysis of the HyGen random graphs, proving that they preserve the degree distribution and clustering coefficients (up to noise).

In the last part of this section, we discuss how the hyperbolic community structure determines the *degree correlation*. This measure is studied to assess to what extent nodes of similar degree connect to each other.

5.4.1 Degree Distribution

The degree distribution is perhaps the most important global property of the graph and has been one of the main topics in many seminal papers [*e.g.* 21, 22, 68], and preserving the distribution (at least approximately) is one of the standard aims of random graph generators.

[22]: Albert et al. (2002), 'Statistical mechanics of complex networks'
[21]: Watts et al. (1998), 'Collective dynamics of 'small-world' networks'
[68]: Faloutsos et al. (1999), 'On power-law relationships of the Internet topology'

As HyGen graphs have disjoint communities, we will first analyse a single community. For the sake of simplicity, we will study the *complementary cumulative degree distribution* $\bar{F}\colon \{0,\dots,|V|\} \to [0,1]$, where $\bar{F}(k)$ is the fraction of vertices with a degree of *at least* k. Clearly, the cumulative degree distribution F is $F(k) = 1-\bar{F}(k)$. For a single community with no noise we have the following lemma.

Lemma 5.4.1 *Let $C = (V_C, E_C)$ be a community that follows the* `hyperbolic`(p,θ) *model perfectly. The complementary cumulative degree distribution $\bar{F}_C$ of C is determined by the parameters p and θ.*

Proof. According to the model, an edge (i,j) is in E_C if and only if $(i+p)(j+p) \le \theta$. Rewriting the equation, we obtain that $(i,j) \in E_C$ if

$$j \le \frac{\theta}{i+p} - p\ . \tag{5.1}$$

That is, vertex i has degree at least j if Equation 5.1 holds,[8] and conversely,

8: Recall that the model assumes that the vertices are sorted by degree.

$$\bar{F}_C(j) = \frac{\max\left\{i \in \mathbb{N}\colon j \le \frac{\theta}{i+p} - p\right\}}{|V_C|}\ . \tag{5.2}$$

□

Lemma 5.4.1 uses the `hyperbolic` model for the simplicity of the proof. Thanks to the equivalence relations Equations 4.9 and 4.10 derived in Section 4.1.1, we can also state the lemma using the potentially more intuitive `fixed` model and its parameters:

Corollary 5.4.2 *The complementary cumulative degree distribution of C, $\bar{F}_C$ is determined by the parameters H and γ.*

Lemma 5.4.1 shows that the degree distribution of a single community is determined by the model. To extend this statement to the full graph, it is enough to notice that as the communities are disjoint, the total degree distribution is the sum of degree distributions of the individual communities.

Lemma 5.4.3 *Given a graph $G = (V, E)$ that is a product of the HyGen model with no noise, its complementary cumulative degree distribution $\bar{F}$ is completely determined by the parameters of its communities.*

Proof. Let G consist of k communities $C_1 = (V_{C_1}, E_{C_1}), \ldots, C_k = (V_{C_k}, E_{C_k})$ and denote by $n_{C_i} = |V_{C_i}|$ the number of nodes in community i. As the communities are disjoint, and there are no vertices outside the communities,

$$n = |V| = \sum_{i=1}^{k} n_{C_i} \,. \tag{5.3}$$

That there is no noise means that (1) communities are perfect in the sense of Lemma 5.4.1, and (2) there are no inter-community edges. Hence, by Lemma 5.4.1, $\bar{F}_{C_i}$ is defined by the parameters of community C_i, and

$$\bar{F}(j) = \frac{1}{n} \sum_{i=1}^{k} n_{C_i} \bar{F}_{C_i}(j) \,, \tag{5.4}$$

that is, the number of nodes in G with degree at most j is the sum of the numbers of nodes with degrees at most j over all the communities. □

Lemmas 5.4.1 and 5.4.3 cover the case where there is no noise. Such an assumption is usually too strict for real-world graphs, and would yield to bad modelling of real-world phenomena. When we add the noise, the model parameters (for example p and θ) will not be enough to define the overall edge distribution. We can show, however, that the noise has most effect to the tails of the degree distribution.

Lemma 5.4.4 *Let G be a HyGen graph with no noise and G' the same graph with a fraction of $q \in [0, 1]$ noise applied, that is, $d_O = q$ and $d_C = 1 - q$ in G'. The* expected degree *of vertex v in G', $\mathbb{E}_{G'}[d(v)]$, is*

$$\mathbb{E}_{G'}[d(v)] = d(v) + q\big(n - 2d(v)\big) \,, \tag{5.5}$$

where $d(v)$ is the degree of v in G and n is the number of vertices in G.

Proof. A fraction of q edges connected to v will be removed due to the noise, and a fraction of q edges not connected to v will be added. Hence,

$$\mathbb{E}_{G'}[d(v)] = d(v) - qd(v) + q\big(n - d(v)\big) \,, \tag{5.6}$$

which simplifies to Equation 5.5. □

Equation 5.5 already hints that if $d(v)$ is large or small, the expected degree can have bigger relative changes. Let $\alpha(v) = {}^{d(v)}/_{n}$, that is $\alpha(v)$ is the *relative degree* of v. Then we can write Equation 5.5 as

$$\mathbb{E}_{G'}[\alpha(v)] = \frac{\mathbb{E}_{G'}[d(v)]}{n} = \alpha(i) + q\big(1 - 2\alpha(i)\big) \,, \tag{5.7}$$

showing that the noise has the most effect on nodes with $\alpha(v) \approx 0$ or $\alpha(v) \approx 1$. On the contrary, if $\alpha(v) = {}^1/_2$, the presence of noise will have no effect on the expected degree.

The above result means that communities that have many vertices with either very few or very many neighbours are most affected by the noise. An extreme example of such a community would be a star, and similarly, communities with steep power-law curves in degree distribution would also see significant changes from small amounts of noise.

5.4.2 Clustering Coefficient

In addition to the degree distribution, the *clustering coefficient* is often used to measure the connectivity of the graphs, and high clustering coefficients are associated with small-world graphs [21].

[21]: Watts et al. (1998), 'Collective dynamics of 'small-world' networks'

There exist two different versions of the clustering coefficient, the *global clustering coefficient* and the *local clustering coefficient* [67].

[67]: Aggarwal et al. (2010), *Managing and Mining Graph Data*

Definition 5.4.1 *Given a graph G, its* global clustering coefficient $\mathrm{cc}(G)$ *is defined as*

$$\mathrm{cc}(G) = \frac{\textit{number of closed triplets in } G}{\textit{number of all triplets in } G} \,. \tag{5.8}$$

Definition 5.4.2 *Given an undirected graph $G = (V, E)$ and its vertex $v \in V$, the* local clustering coefficient $\mathrm{cc}_v(G)$ *of v in G is defined as*

$$\mathrm{cc}_v(G) = \frac{2\big|\{(u,w)\colon u, w \in N(v), \{u,w\} \in E\}\big|}{d(v)\big(d(v)-1\big)} \,, \tag{5.9}$$

where $N(v)$ is the neighbourhood of v.

We can again show that, up to the effects of the noise, the clustering coefficients – both local and global – are fully determined by the model parameters. We start by analysing the local clustering coefficient in a single community.

Lemma 5.4.5 *Let $C = (V_C, E_C)$ be a community that follows the* `hyperbolic`(p, θ) *without any noise and let i be an arbitrary vertex of C. The local clustering coefficient $\mathrm{cc}_i(C)$ of i in C is determined entirely by p and θ.*

Proof. In Definition 5.4.1, three terms that determine the clustering coefficient: neighbourhood $N(i)$, set of edges E_C, and degree $d(i)$. Given that C follows `hyperbolic`(p, θ), we can use Equation 5.1 to express the neighbourhood of v as

$$N(i) = \left\{h \in V_C \colon h \leq \frac{\theta}{i+p} - p\right\} \tag{5.10}$$

and the set of edges E_C as

$$E_C = \left\{i, j \in V_C \colon j \leq \frac{\theta}{i+p} - p\right\}, \tag{5.11}$$

showing that they both depend only on p and θ. That the degree of vertex i, $d(i)$, depends only on p and θ follows from Lemma 5.4.1. □

To analyse the global clustering coefficient cc, we will first make some definitions. Define the indicator function $\chi(i, j)$ as

$$\chi(i,j) = \begin{cases} 1 & j \leq \frac{\theta}{i+p} - p \\ 0 & \text{otherwise}. \end{cases} \tag{5.12}$$

This function indicates for every pair of vertices i and j if there would be an edge between them in a community following the `hyperbolic`(p, θ) model with no noise.

The number of closed triplets in a community following noise-free `hyperbolic`(p, θ) model can be counted by testing whether all of the edges between the vertices exist. Define $T^{\mathrm{cl}} \colon \mathbb{N}^3 \to \{0, 1\}$ as

$$T^{\mathrm{cl}}(i,j,h) = \chi(i,j) \cdot \chi(i,h) \cdot \chi(j,h). \tag{5.13}$$

An open triplet (wedge) is a set of three vertices connected by exactly two edges. Define $T^{\mathrm{o}} \colon \mathbb{N}^3 \to \{0, 1\}$ as

$$T^{\mathrm{o}}(i,j,h) = \chi(i,j) \cdot \chi(i,h) \cdot \big(1 - \chi(j,h)\big) \tag{5.14}$$

to test whether (i, j, h) is an open triplet centred at i.

Lemma 5.4.6 *Let C be as in Lemma 5.4.5. The global clustering coefficient* $\mathrm{cc}(C)$ *of C is entirely determined by p and θ.*

Proof. As the value of $\chi(i,j)$ is defined by p and θ (with fixed i and j), so are the values of T^{cl} and T^{o}. We can write $\mathrm{cc}(C)$ from Definition 5.4.1 with T^{cl} and T^{o} as

$$\mathrm{cc}(C) = \frac{\sum_{i,j,h\in V_C} T^{\mathrm{cl}}(i,j,h)}{\sum_{i,j,h\in V_C}\left(T^{\mathrm{cl}}(i,j,h) + T^{\mathrm{o}}(i,j,h)\right)}\,, \tag{5.15}$$

where we always assume that i, j, and h are disjoint. □

The above Lemmas 5.4.5 and 5.4.6 can be extended to a full graph following noise-free HyGen model.

Lemma 5.4.7 *Let $G = (V,E)$ be a graph that follows the HyGen model with no noise. The global clustering coefficient* $\mathrm{cc}(G)$ *and the local clustering coefficient* $\mathrm{cc}_v(G)$ *for any $v \in V$ are determined through the parameters of the hyperbolic communities of G.*

Proof. As there are no inter-community edges, the neighbourhood of any $v \in V$ is entirely contained in the community where v is, and Lemma 5.4.5 applies directly. The function $\chi()$, testing whether there is an edge between vertices i and j, also needs to take the communities into account. It can be re-defined as

$$\chi(u,v) = \begin{cases} 1 & \text{if } u,v \in C \text{ and } \pi_C(u) \leq \frac{\theta_C}{\pi_C(v)+p_C} - p_C \\ 0 & \text{otherwise,} \end{cases}$$

where p_C and θ_C are the parameters of the community C and π_C is the permutation associated with it. With this definition of $\chi()$, the functions T^{cl} and T^{o} will also work throughout the full graph, concluding the proof. □

Lemma 5.4.7 also shows that the *average local clustering coefficient* is determined by the community parameters.

Corollary 5.4.8 *Let G be as above. The* average local clustering coefficient $\overline{\mathrm{cc}}(G) = \frac{1}{|V|}\sum_{v\in V}\mathrm{cc}_v(G)$ *is entirely determined by the community parameters.*

It is not trivial to analyse the effects of noise to the clustering coefficient. Triangles or wedges from the inside-community area disappear, and new ones get introduced involving the outside-community area. Given the overall density of a graph, the expected number of triangles or wedges is derivable, but integrating the specific intra-community structure into this expectation remains an open problem.[9]

9: There are $\binom{\gamma+1}{3}$ triangles in the core, and every i with $d(i) \geq 2$ adds $\binom{d(i)}{2}$ more triangles because it only has connections to the core. To the best of our knowledge, there is no similar expression to determine the number of wedges.

5.4.3 Degree Correlation

The degree correlation measures whether the number of links between nodes with high degree and nodes with low degree is systematically different from what is expected in a random network. A network is called assortative if nodes of similar degree tend to link to each other, and dissassortative if the network exhibits a preference for links between nodes of dissimilar degree. Both trends of correlation may occur in real world networks. While social networks tend to be assortative, other kinds of networks, including those of question-answering portals, are typically observed to be disassortative [67].

[67]: Aggarwal et al. (2010), *Managing and Mining Graph Data*

Positive degree correlation in social networks occurs since the degree-ordered adjacency matrix typically shows a banded structure, meaning that links between nodes of similar degree occur more likely than other links. The modular character of such networks with multiple, potentially distinct, communities helps assortativity. Interestingly, for the building blocks of such modular networks, the individual communities, we observe an opposite trend of degree correlation.

Using results of prior work, we may claim that hyperbolic communities are usually disassortative: In [69], Johnson, Domínguez-García, and Muñoz study the relation between nestedness of networks and disassortivity. They show that with high probability, disassortative networks are nested and vice versa. Let $G = (V, E)$ be an undirected graph. We say G is *nested* if we can order the vertices $v \in V$ in a sequence $(v_1, v_2, \ldots, v_n)$ such that $N(v_{i+1}) \subseteq N(v_i)$ for all $i = 1, \ldots, n-1$. As discussed by Karaev, Metzler, and Miettinen [40], hyperbolic communities are a special case of nested matrices. Notice that also communities whose adjacency is described by a power law function are therefore disassortative since the power law is special case of the hyperbolic model.

[69]: Jonhson et al. (2013), 'Factors Determining Nestedness in Complex Networks'

[40]: Karaev et al. (2018), 'Logistic-Tropical Decompositions and Nested Subgraphs'

Validation of the Hyperbolic Community Model

6

In this chapter, we demonstrate the use of the hyperbolic community model. In our experimental study, we analyse a large variety of real-world datasets. We observe that our modelling approach explains real graphs much better than traditional quasi-clique models and also improves preceding approaches.

6.1 Datasets

For our experiments, we use two collections of real-world data sets. In the first set of experiments, in Sections 6.2 and 6.3, we use graphs with given ground-truth communities from the Stanford Large Network Dataset collection [70], called SNAP. Our goal is to study the shape of these communities, and the differences between our model and power law models, as well as quasi-clique models. The employed data sets originate from different large online resources where nodes of the respective network typically correspond to people and edges indicate interactions or friend relationships. In summary, the SNAP data collection comprises

[70]: Leskovec et al. (2014), *SNAP Datasets: Stanford Large Network Dataset Collection*

- **Amazon** — products bought together, grouped by product category;
- **DBLP** — author collaboration network, grouped by venue;
- **Friendster** — online gamers' network of friendships with user-defined groups;
- **LiveJournal** — bloggers' friendship network with user-defined groups;
- **Orkut** — friendship network with user-defined groups;
- **YouTube** — video sharers' network of friendships with user-defined groups.

Their size ranges from 300 000 nodes up to 65 million nodes, as outlined in Table 6.1. As neither very small nor very large communities are particularly interesting, we restrict our analysis to communities with between 100 and 1000 nodes (inclusive). We use a sample of 500 communities from each of these data sets.[1] For DBLP we also use a sample of 100 communities which we denote DBLP(100).[2] Notice that the ground-truth information for the communities is only provided for the nodes and not for the edges.

1: Notice that the YouTube network has only 129 communities within that size range; hence we use them all.

2: We discuss the reasons in Section 6.3.

Table 6.1: Sizes of the SNAP datasets used for evaluation, their total number of communities as well as the number of communities of size 100 to 1000, and the time it took to determine a hyperbolic model for a sample of size 500.

	nodes	edges	communities		time
			all	100–1000	(h)
Amazon	334 863	925 872	75 149	1 380	0.6
DBLP	317 080	1 049 866	13 477	805	27.0
Friendster	65 608 366	1 806 067 135	957 154	19 763	12.3
LiveJournal	3 997 962	34 681 189	287 512	8 769	11.3
Orkut	3 072 441	117 185 083	6 288 363	80 251	3.1
YouTube	1 134 890	2 987 624	8 385	129	0.8

Table 6.2: Datasets used for the community finding experiments. We choose k, the number of clusters, as displayed.

	nodes	edges	content	k
Email	1 133	10 902	University email network	10
Erdős	472	2 628	Erdős collaboration network	8
Jazz	198	5 481	Network of Jazz musicians	5
PolBooks	105	882	Books about US politics	6

The second collection of data, called SuiteSparse, originates from the University of Florida Sparse Matrix Collection [71].[3] As outlined in Table 6.2, the data are significantly smaller than in the SNAP collection. Also, they are not annotated with community information. We use SuiteSparse for a second group of experiments in Section 6.4 where we employ existing community-detection algorithms to find the communities, and then apply our model to the found communities. Their smaller size allows the community detection algorithms to work efficiently.

[71]: Davis et al. (2011), 'The University of Florida Sparse Matrix Collection'

3: The data collection renamed to SuiteSparse Matrix Collection.

6.2 Models from Annotated Communities

We start by studying the results of fitting our model to the SNAP networks for which ground-truth communities are given.[4]

Fitting the hyperbolic community model in its `fixed` parametrisation yields the parameters γ and H for every community. We summarise the results per data set in Figures 6.1 and 6.2. Additionally, we display the distribution of the x which we infer by converting the results into the `mixture` parametrisation (see Section 4.1.3) in Figure 6.3. The boxplots display distributions such that in each box, the central mark is the median and the edges of the box are the first and third quartiles. The whiskers extend

4: The source code for the algorithm and the scripts to run the experiments are available at `https://cs.uef.fi/~pauli/hybobo/`.

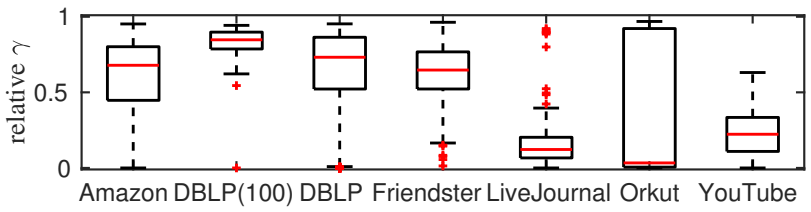


Figure 6.1: Distribution of γ relative to the community size n_C after fitting a hyperbolic model to the sampled communities.

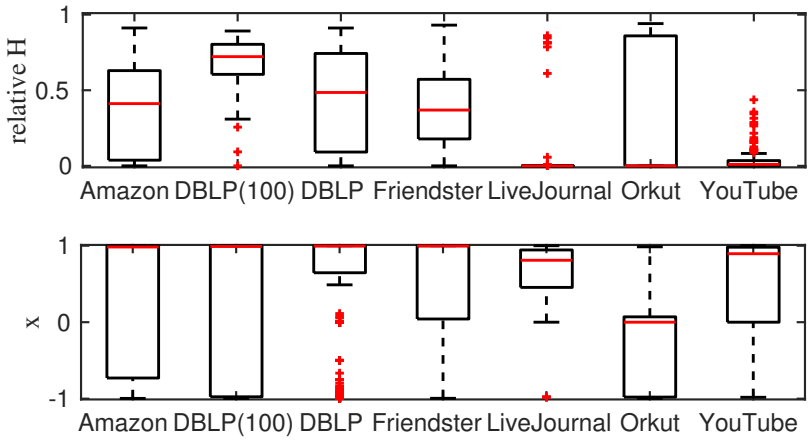


Figure 6.2: Distribution of H relative to the community size n_C after fitting a hyperbolic model to the sampled communities.

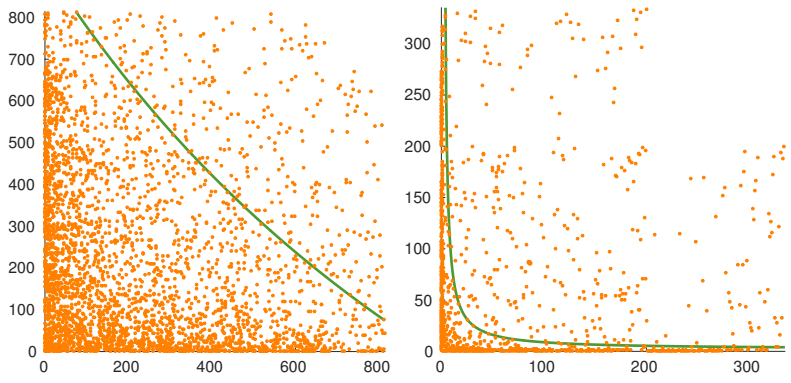
Figure 6.3: Distribution of x after fitting a hyperbolic model to the sampled communities.

to the most extreme data points that are not outliers. Points are considered outliers if they are larger (smaller) than the third (first) quantile plus (minus) 1.5 times the difference between the quantiles. Outliers are plotted individually.

The boxplots reveal a characteristic shape for each data set: Amazon, DBLP, and Friendster show rather thick communities with γ on median being more than 50 per cent of the community size. LiveJournal, Orkut, and YouTube communities mostly have thin cores. While Orkut also has communities with big cores, LiveJournal is at the other extreme and its communities mostly exhibit a star-like shape. These observations are in line with the intuitive idea about some of the data sets. DBLP, for instance, is the network of co-authorships and therefore many communities are almost quasi-cliques.

Furthermore, we observe that the obtained models for all datasets differ significantly from the `HyCom` model that would require $x = 0.5$ (see Section 4.2). As Figure 6.3 shows, none of the medians is near $x = 0.5$.

To gain intuition on how these communities look like, we give examples in Figure 6.4 (and above in Figure 3.1), and refer to Appendix B for further examples.



(a) Community from Amazon. **(b)** Community from Friendster.

Figure 6.4: Examples of hyperbolic community models, depicted as degree-ordered adjacency matrices where dots indicate edges between nodes i and j. The green curve indicates the fitted hyperbolic model.

Running time. Our algorithm is implemented in Matlab and C. It took between half an hour and a full day to compute the models on a machine with four Intel Xeon E7-4860 10-core CPUs running at 2.27 GHz and 256 GB of main memory. The exact running times are given in Table 6.1. Notice that the time depends not only on the size of the graphs, but to a great extend on the amount of overlap between the communities. Overlapping communities have to be computed in a sequential manner as the model of one community has impact on the next (see Algorithm 1). Additionally, an update to one model causes the re-computation of all communities that overlap with it. Hence, the computation for DBLP takes more than twice as much time than that for Friendster although the Friendster graph is orders of magnitude larger.

6.3 Comparison to Alternative Models

In Section 4.2, we describe special cases of our model: In the case where every community is assumed to be a quasi-clique, the community size, n_C, determines the model parameters structure, as $H = \gamma = n_C$. When the communities are assumed to follow a power law pattern, only the threshold parameter Σ may vary and, assuming HyCom [3] as the power law model, the other is fixed to $x = 0.5$.

[3]: Araujo et al. (2014), 'Beyond blocks: Hyperbolic community detection'

In this section, we assess the benefit of the additional flexibility of our model by comparing to these restricted versions. To do so, we compare the log-likelihoods (Definition 4.3.1) of our hyperbolic models[5] for the SNAP collection to the log-likelihoods of the respective HyCom models and block models. For the HyCom models, we run Algorithm 1 only admitting those (γ, H) combinations that yield $x = 0.5$. For the block models, no parameter search is necessary as there is only one admitted configuration per community.

5: We obtain the models just as in Section 6.2.

To compare the log-likelihood, we use the likelihood ratio test [72, Ch. 10.6]. In case of the block model, we test the null hypothesis H_0 that all parameters,[6] are fixed so that they create block structures versus the alternative hypothesis H_1 that the parameters are not fixed; for HyCom, we assume one free parameter. The likelihood ratio test statistics are given by

[72]: Wasserman (2004), *All of Statistics: A Concise Course in Statistical Inference*

6: All parameters refers to all H and γ of all communities.

$$\lambda = 2 \log \frac{L(\text{our model})}{L(\text{block model})}$$

and

$$\lambda = 2 \log \frac{L(\text{our model})}{L(\texttt{HyCom}\ \text{model})}$$

for block and HyCom models, respectively.

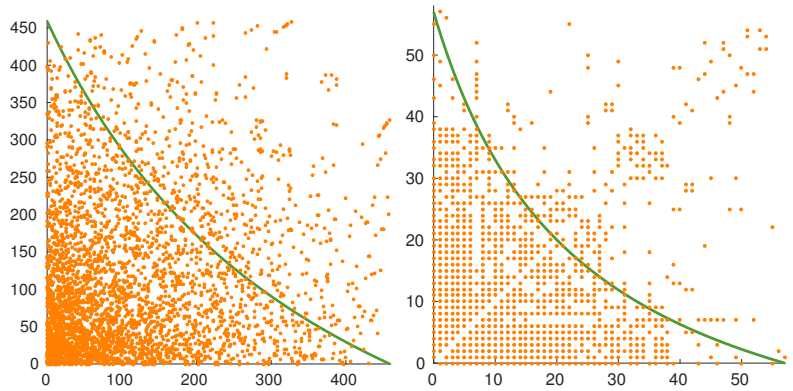
	likelihood ratio	
	block model	HyCom
Amazon	26 450.6	30 997.1
DBLP(100)	3 148.5	−788.0
DBLP	−264 974.7	17 958.1
Friendster	200 627.6	17 811.7
LiveJournal	154 982.4	22 705.8
Orkut	11 945.3	1 598.5
YouTube	75 689.6	12 660.0

Table 6.3: Test statistic of the likelihood ratio test between our models and block models, and our models and HyCom. For all datasets 500 communities were sampled. Additionally, we display the result for sampling 100 communities from the DBLP data.

Block model. We can see from Table 6.3 that, for the block model, statistic yields highly positive values, with one exception. Since the derived p-values are always essentially zero, we confirm that the hyperbolic model is statistically significantly better than the block model.[7] The exception is the 500 sample of the DBLP data. For this data set, the block model gives a better likelihood than ours. While cliques are a special case of our model, and hence we can always model each community as a clique, our iterative method to update the model parameters (see Section 4.5.2) is based on a greedy heuristic. In case of the DBLP data, the greedy heuristic has reached a local optimum that is less good than what could be obtained with pure block models. On one hand, this is partially because DBLP contains block-like communities, and on the other hand, the large overlaps between these communities might have lead the optimisation astray. To test the latter hypothesis, we also sampled just 100 communities from DBLP: this reduced the amount of overlap between the communities (from 33 % to 18 %) and also significantly improved the results of our algorithm.

7: We will see a similar result in a repetition of this experiment on different data in Section 8.3.1.

HyCom. For the comparison to the HyCom model, we obtain a similar result: with one exception, our hyperbolic model describes the data statistically significantly better than the HyCom model. The better solution for DBLP(100) found with the parameter space restricted to HyCom models is also a valid solution within our more general modelling framework. The greedy algorithm we propose, however, gives no guarantee to converge to the globally best solution and this result indicates that it depends on the data whether additional freedom in the parameter space is a benefit or hindrance *for the algorithm* to find a good solution. More importantly, as we will see in the next section, starting with HyCom as initialisation, our model is always significantly better.



(a) Community from Email data. **(b)** Community from Jazz data.

Figure 6.5: Examples of communities obtained via spectral clustering and fitted by our model.

6.4 Finding Communities

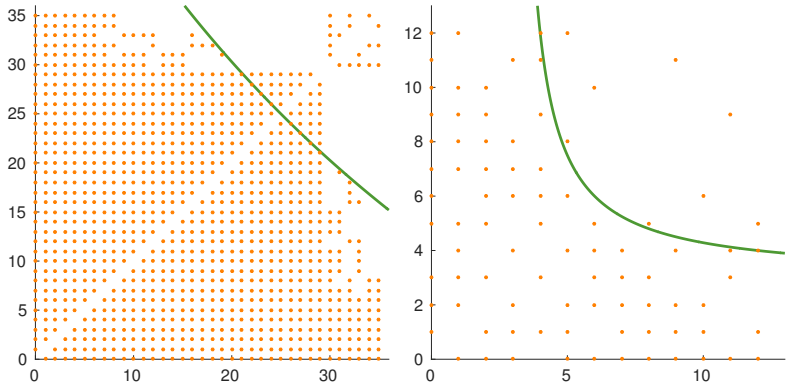
Next, we demonstrate that our model improves the description of communities returned by existing community-finding methods. To that end, we performed spectral clustering, Boolean matrix factorisation (BMF), and `HyCom` to find the communities in the SuiteSparse collection; the first two approaches look for clique-like communities while `HyCom` looks for hyperbolic shapes (see Section 4.2). For each approach, using the likelihood ratio test, we compare the original outcome to the result after fitting hyperbolic models on the outcome.

[73]: Luxburg (2007), 'A Tutorial on Spectral Clustering'

Spectral clustering. We use spectral clustering with the normalised Laplacian [73] to cluster the nodes of the graph. The resulting communities are non-overlapping. We notice a significant benefit of modelling the obtained result by means of hyperbolic models, as the likelihood ratio test confirms for all examined datasets (see Table 6.4). These results yield p-values that are essentially zero, confirming that the results are statistically significant. We have chosen the number of clusters k as indicated in Table 6.2. The examples of modelled communities in Figure 6.5 show relatively large cores but thin tails, with most edges being in the lower triangular area. Our models clearly capture this phenomenon.

Table 6.4: Statistic of the likelihood ratio test between our model and block models (resulting from spectral clustering and BMF), as well as our and `HyCom` models. The communities are found using `HyCom`, spectral clustering, and BMF.

	likelihood ratio		
	spectral clustering	BMF	`HyCom`
Email	10 895.8	3 552.0	250.1
Erdős	1797.0	949.0	256.3
Jazz	3 003.8	4 435.0	3 718.5
PolBooks	648.0	303.3	228.2



(a) Community from Jazz data. **(b)** Community from PolBooks data.

Figure 6.6: Examples of communities obtained via BMF and fitted by our model.

BMF. To find overlapping communities, we use BMF and run the Asso algorithm [32] with the number of communities k as in Table 6.2. We set the threshold parameter τ of Asso to 0.6 and the weight w to 10. We again find that our models resemble the data significantly better than the corresponding block models (see Table 6.4). Figure 6.6 shows example communities from the PolBooks and Jazz data. Further examples are displayed in Appendix B.

[32]: Miettinen et al. (2008), 'The Discrete Basis Problem'

`HyCom` algorithm. We run the `HyCom` algorithm [3] on each of the data sets stopping after we found k communities, with k specified in Table 6.2. As the likelihood ratio test confirms (Table 6.4), our hyperbolic model improves the result of the `HyCom` algorithm.

[3]: Araujo et al. (2014), 'Beyond blocks: Hyperbolic community detection'

6.5 Discussion

Our experiments have verified our intuition that the communities in real graphs are better modelled using our models than the traditional quasi-clique models, and that our models are an improvement over the previously proposed `HyCom` model [3]. This holds true for a variety of data sets, both with ground-truth communities, and with communities detected with existing methods. It is important to notice that the existing methods, especially BMF, aim at finding clique-like communities. Thus, since our algorithm uses their results as the initial community candidates, any weaknesses of these algorithms will also affect our result. Still, our experiments show that our models provide statistically significantly better fit, even when we take into account the increased number of free parameters for our model.

One advantage of our hyperbolic community model is that it considers overlap between communities. As we discuss in Section 4.3, communities may overlap in three different ways; in addition to

the nodes, our model considers the edges that constitute a community. Thus communities can be node-overlapping but edge-disjoint. Exactly that case is not entirely represented in Algorithm 1. For the computation of full graph models, we have restricted edges to be part of at most one community. We use a heuristic approach to assign an edge to that community where it best improves the log-likelihood. Avoiding the need to regard partial memberships is advantageous as it simplifies the log-likelihood computation. But there might be merits of extending our algorithm to handle edge overlaps more elegantly.

Already with the present implementation, we observe that our model is not only a better fit for the data, but also provides interesting insights to the shape of the communities. The easy interpretability of the parameters γ and H means that we can simply study a summary of their distributions to gain an understanding on how the communities in a data look like, whether the cores are small or big and whether the tails are fat or skinny. This allows a data analyst to obtain a fast general understanding about the data without having to look at any particular community.

Finally, our experiments also demonstrate the scalability of our method. It had no problem of handling even the largest graph, Friendster, with approximately 65.6 million nodes and 1.8 billion edges.

Graph Generation with HyGen

7

In this chapter we conduct a series of experiments to verify the use of the hyperbolic graph generator. We start by exploring to what extent previous graph generators are able to capture hyperbolic community structure. In further experiments, we demonstrate that HyGen generates modular networks with realistic intra-community structures using parameter distributions derived from observations on real graphs.

7.1 Datasets

We begin our experimentation by studying on artificially generated data how related random graph generators work at modelling the kind of communities observed in real-world graphs in Section 7.2. For the remainder of this chapter, we employ two different collections of real-world data sets.

The first collection is SNAP. We use four networks from the Stanford Large Network Dataset collection [70]: Amazon, DBLP, Friendster, and YouTube. As detailed in Section 6.2, these real-world social networks have ground-truth community information.[1] We use the SNAP collection to explore the parameter distributions for HyGen in Section 7.3, and to test for the stability and the randomness of HyGen-generated graphs, in Sections 7.4 and 7.5.

The second collection of real-world datasets is StackExchange. In Section 7.6, we use four time-evolving communities from the collection of online question–answer sites on `stackexchange.com` [74]:[2]

- `gaming.stackexchange.com`,
- `gardening.stackexchange.com`,
- `tex.stackexchange.com`,
- `unix.stackexchange.com`.

The data sets contain a snapshot of the community at the begin of each month from the start date until November 2016. Some basic properties of these communities are listed in Table 7.1.

We base our evaluation on the HyGen parameters, since (1) up to noise, clustering coefficient as well as degree distribution are derivable from the HyGen parameters (see Section 5.3), and (2) inter-community connections such as path length, modularity, or

[70]: Leskovec et al. (2014), *SNAP Datasets: Stanford Large Network Dataset Collection*

1: An overview of the SNAP data is in Table 6.1.

[74]: Stack Exchange, Inc. (2016), *Stack Exchange Data Dump*

2: The whole StackExchange data collection is introduced in Section 8.2. In this chapter we use an excerpt of four communities to provide proof of concept of how to model time-evolving communities.

Table 7.1: Basic properties of four StackExchange communities. The average γ_R and average H_R are computed using a weighted average with the community size at each month being the weight.

	Start date	largest size	average γ_R	average H_R
`gaming`	2009-08-01	3 818	0.106	0.004
`gardening`	2010-07-01	446	0.124	0.009
`tex`	2008-08-01	3 342	0.074	0.009
`unix`	2008-08-01	5 450	0.073	0.001

degree correlation would measure only added noise, or in case of the degree correlation of individual communities, be already predetermined through the structures we allow.[3]

3: Matlab code for the graph generator and scripts for the experiments can be obtained from `https://cs.uef.fi/~pauli/hybobo/rgg/`.

7.2 Limits of Current Graph Generators

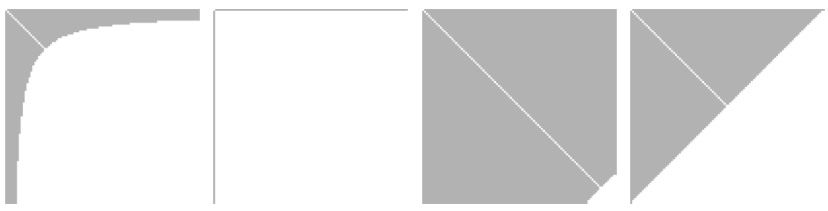
While many of the existing random graph generators cannot be used to model community structures in networks at all, there are a number of related approaches that allow for generated graphs with community structure. To the best of our knowledge however, there exists no approach of preserving the intra-community connection patterns in the modelling process.

In this section, we detail why SBM, DC-SBM,LFR, and R-MAT do not provide solutions to the modelling task we aim to solve. For this purpose, we compare the outcome of these generators when fitted to idealised hyperbolic communities. One may argue that such communities are not realistic and therefore not a fair basis of comparison for these generators that are designed to model real-world graphs, which are usually sparse and exhibit a certain level of noise. However, the purpose of this experiment is to see to what extent the *pure* hyperbolic structure can be captured at all by the different generators.

7.2.1 Experimental Setup

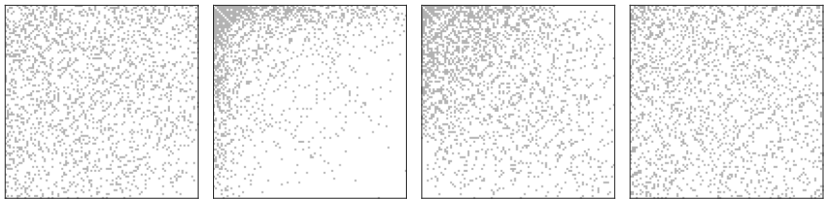
We generate single hyperbolic communities with the extremes of shapes that can be represented: a star, a near-clique, a triangular, and a stereotypical core-tail pattern (see Figure 7.1).[4] Each of these communities consists of 100 nodes. For each of the graph generators, SBM, DC-SBM, LFR, and R-MAT, we learn the respective

4: Compare with examples of real communities in Figures 3.1 and 6.4 to 6.6, as well as in Appendix B.



(a) 20 %-core **(b)** star **(c)** near-clique **(d)** triangle

Figure 7.1: Adjacency matrices of idealised hyperbolic communities. The different extremes of hyperbolic structures, range from star-like to near-clique.



(a) SBM **(b)** DC-SBM **(c)** R-MAT **(d)** LFR

Figure 7.2: How the compared graph generators create a new random community when given the 20 %-core community shown in Figure 7.1a as input. The adjacency matrices in each subfigure are degree-ordered.

parameters with these communities as input and generate new graphs according to those parameters. After that, we fit the best hyperbolic model on the newly generated graphs, measure the fitting quality in terms of log-likelihood, and compare how well the obtained model matches the original input. We repeat this experiment 50 times to counteract effects of randomness. Figure 7.3 summarises the resulting hyperbolic models that best fit the communities generated by the respective method. We overlay the 50 obtained models in light grey. Ideally, we would expect to see the exactly the structures of Figure 7.1 again.

Notice that the LFR implementation strictly requires multiple communities to be generated. Therefore, we provide graphs containing twice the same community as input (like illustrated in Figure 7.4a for the 20 %-core[5]) and use the better matching one in the evaluation of the fit quality.

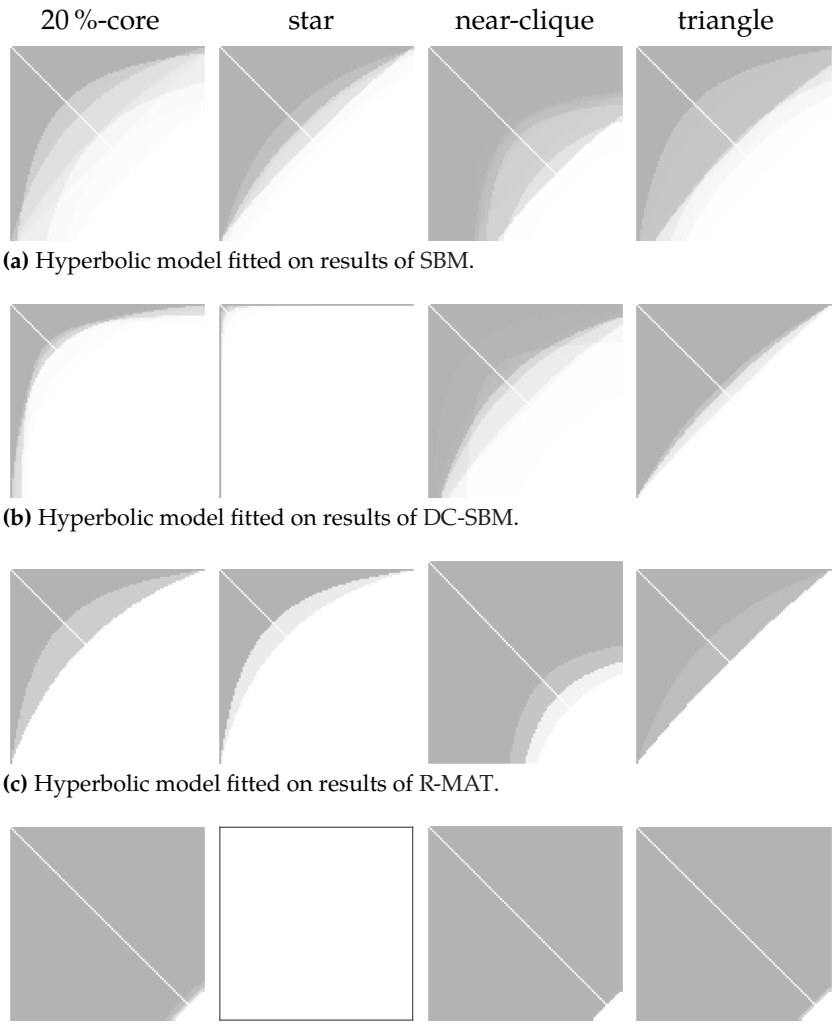
5: The core of the stereotypical core-tail community is formed by 20 % of the nodes, hence the name.

7.2.2 SBM

The standard SBM is based on the assumption that vertices within a block are stochastically equivalent. The hyperbolic model fulfils this assumption only in the extreme case of a quasi-clique, where the size of the core equals the size of the community. The typical core-tail structure of hyperbolic communities cannot be captured (see Figure 7.2a). As the generated graphs after fitting the SBM model are almost uniformly dense, the hyperbolic models fitted on these outputs exhibit a huge degree of variance and often differ a lot from the original input (see Figure 7.3a). The high negative log-likelihood scores of the fits (see Table 7.2) indicate that the fitted hyperbolic models are also not particularly good at explaining SBM-generated communities. Regarding the case of a near-clique, it is worth pointing out that a perfect clique

	20 %-core	star	near-clique	triangle
SBM	−2 354±1 919	−451±1 406	−282±1 007	−3 393±39
DC-SBM	−1 545±1 734	−212±713	−3 238±521	−2 679±768
R-MAT	−2 008±38 619	−440±958	−214±548	−2 727±230 133
LFR	−2 515±7 697	—	−1 299±17 346	−3 714±6 250

Table 7.2: Average best log-likelihood for 50 trials of generating hyperbolic communities with SBM, DC-SBM, R-MAT, and LFR, with standard deviations (SDs). Notice that HyGen would achieve the ideal value of 0 in each of the examined cases.



(a) Hyperbolic model fitted on results of SBM.

(b) Hyperbolic model fitted on results of DC-SBM.

(c) Hyperbolic model fitted on results of R-MAT.

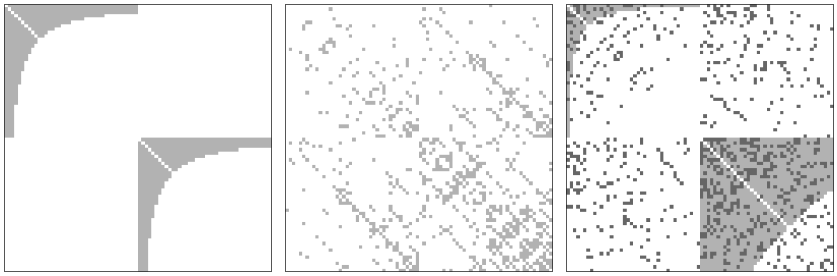
(d) Hyperbolic model fitted on results of LFR.

Figure 7.3: Hyperbolic model fitted on results of alternate generators. Each subfigure summarises 50 repetitions of using the alternative generator to fit the respective input displayed in Figure 7.1. For every sample, the adjacency matrix of the closest hyperbolic model is displayed in light grey. Shades of grey result from overlaying all samples and yield a visual summarisation of the observed shapes. The ideal result would be complete resemblance to the respective input.

would be recovered well by the SBM. The case of the near-clique (Figure 7.3a, column 3) is substantially harder: almost all nodes are connected to each other but a few miss some connections. For the SBM however, the primary aim is to match the overall density evenly. Thus the fit is such that many nodes are fully connected and some are connected to almost every other node, which yields into a substantially different looking hyperbolic model.

7.2.3 DC-SBM

DC-SBMs allow for variation of the degree within a community (see Figure 7.2b). Fitting back a hyperbolic model on the DC-SBM outcome of modelling a 20 %-core structure, a star, or a triangle are fairly accurate in the sense that the parameters γ and H are close to the original (see Figure 7.3b). Yet, the hyperbolic models fitted on the DC-SBM-generated communities leave some amount of noise to be explained otherwise (see Table 7.2). The DC-SBM expects a power-law degree distribution within the communities



(a) input graph **(b)** R-MAT result **(c)** fitted model

Figure 7.4: How R-MAT reconstructs a graph of two identical hyperbolic communities. To fit the hyperbolic models, we assume that the community sizes remained unchanged during modelling. Fitting implies degree ordering the communities.

and draws edges from that one to recover the connectivity pattern inside the community. In particular the near-clique case (see Figure 7.3b, column 3) seems to be hard to explain by a power law. The hyperbolic model is more general in the sense that it includes power-law distributions as a special case. It also has a substantially different noise model, assuming uniform density for the inside-community area and as well as for the outside.

7.2.4 R-MAT

R-MAT is designed to model the degree distribution of the input data using a recursive procedure. The results we observe for the single artificial communities are comparable to those of the DC-SBM (see Figure 7.3c). The recursive construction procedure however introduces particular structures in the data. To construct a graph, R-MAT subdivides the adjacency matrix recursively into quarters of certain density. This makes it hard to capture multiple communities, especially of unequal sizes. An additional experiment reveals that, already if R-MAT is fitted on a graph of two equally sized (hyperbolic) communities with no inter-connections, the resulting model is not capturing the this structure well (see Figure 7.4): by its definition, R-MAT models consist of four self-similar blocks. This means, the blocks with community structure are always mirrored to the off-diagonal, introducing many surplus links between the communities.

7.2.5 LFR

The LFR benchmark generates random graphs given power-law distributions for the node degree and the community size. Creating graphs that consist of a single community is not included as a special case in this approach. To still obtain comparable modelling results for the four sample hyperbolic communities (see Figure 7.1), we fit LFR on graphs consisting of twice each of those communities. The reported log-likelihood scores of fitting a hyperbolic model on

the LFR results then refer to the better of the two obtained communities. While for the star pattern, we could not find a set of valid LFR parameters to describe this pattern with the procedure suggested by Lancichinetti, Fortunato, and Radicchi [58], we observe that the remaining hyperbolic communities are modelled very similar to each other by LFR, as the best hyperbolic model to explain these communities is the same in each case (see Figure 7.3d). A closer look at the LFR-generated graphs reveals that they actually differ: the average degree per community is retained from the original communities, but the hyperbolic structure is lost.

[58]: Lancichinetti et al. (2008), 'Benchmark graphs for testing community detection algorithms'

7.3 Distributions for the Parameters

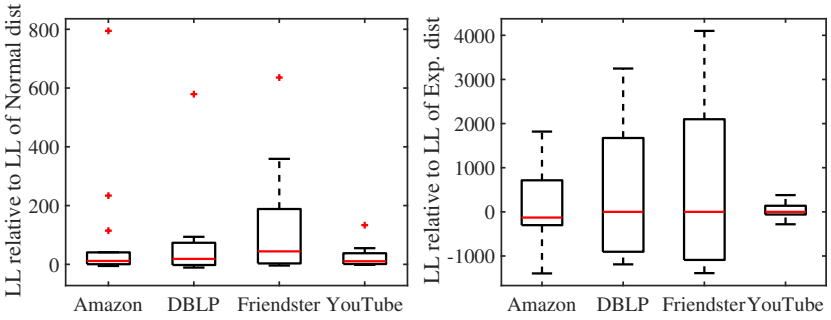
In the theoretic derivation of HyGen in Section 5.1, we suggested distribution functions to use for the distributions of the shape parameters γ and H and the size n_C. Here we detail how these suggestions were obtained.

For these experiments, we use the SNAP collection of real-world data sets. Like in Section 6.2, we fit the hyperbolic model to a sample of 500 communities of size between 100 and 1000 nodes from each network. Thereby, we obtain the empirical distributions for γ, H, and the (truncated) community size n_C.

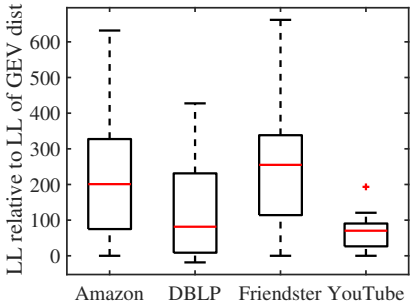
For each of the empirical distributions for γ, H, and n_C, we fit different distributions (namely, generalised extreme value (GEV) distribution, inverse Gaussian distribution, Birnbaum–Saunders distribution, exponential distribution, log-normal distribution, log-logistic distribution, gamma distribution, Rayleigh distribution, Weibull distribution, Nakagami distribution, Rician distribution, normal distribution, logistic distribution, extreme value distribution, and t-location-scale distribution). Not every distribution is applicable for each of the parameters. While the observed γs look normally distributed, H and the community size show an exponential behaviour.

In order to validate our subjective observations, we tested how well the different distributions fit using the negative log-likelihood. Alternative reasonable measures for evaluation would be Akaike or Bayesian information criterion. We observe highly similar results with either of the measures and therefore only present the evaluation with respect to the log-likelihood. To show the results in a concise manner, we present here only comparisons of our chosen distributions against all other options. Namely, for every distribution D that is not the one we chose to model parameter p,[6] we compute our test statistics $T_p(D) = LL_p(D) - LL_p(D^*)$, where $LL_p(D)$ is the negative log-likelihood of modelling parameter p

6: Here, p refers to either γ, H, or the size n_C.



(a) Normal distribution for γ. **(b)** Exponential distribution for H.



(c) GEV for community size n_C.

Figure 7.5: Relative log-likelihoods of the hypothesised distributions compared to the other distributions. For each dataset, the boxplot indicates how well each parameter follows the hypothesised distribution compared to other potential distributions. Positive values indicate that the preferred distribution performs better than the others and zero indicates even performance.

using distribution D and $LL_p(D^*)$ is that for the selected distribution D^*. The larger values $T_p(D)$ obtains, the better the selected distribution performs compared to distribution D, $T_p(D) = 0$ indicates that D and D^* perform equally well, and negative values indicate that D performs better than the chosen distribution D^*.

Based on our experiments, we propose to model γ (relative to the community size) using the normal distribution, H (relative to γ) using the exponential distribution, and the community size n_C using the GEV distribution. Our comparison of these distributions against others in the SNAP datasets is presented in Figure 7.5.

As can be seen in Figure 7.5a, the normal distribution for γ is constantly at least as good as the other distributions,[7] implying that the use of normal distribution is a valid choice. The situation when modelling the distribution of the H parameter (with respect to γ) is more complicated. Our experiments showed the exponential distribution to have the best fit, but as can be seen in Figure 7.5b, in some cases other distributions would be better. More experiments would be needed to give a conclusive answer to the question which distribution explains observed Hs best.

For modelling the distribution of the community size (Figure 7.5c), the GEV distribution is always the best, and shows the strongest performance against the other distributions. Please note, however, that in this test we model the distribution of the sizes of the communities in a single network at one point of time; when studying the sizes of a single community over time, very different distributions might be needed.[8]

7: Negative values would indicate lesser performance.

8: Figure 7.7 gives an impression of how the community size varies.

7.4 Stability of the Graph Generation

We now turn our attention into analysing the HyGen-generated graphs. In this section we will study how well the generated graphs fit to the real-world graphs; in the next section, we will study how random the generated graphs are.

We use the following procedure to test how well the generated graphs fit to the real-world data they were generated from: First, we fit the hyperparameters of the parameter distributions to real-world networks from the SNAP collection. Then we use these distributions in HyGen to sample collections of random graphs. Finally, we compute the best hyperbolic model for each community in the generated graphs and evaluate how accurately the found communities match to the original communities. Our hypothesis is that the found communities have similar distributions of parameters as the original communities, indicating that the generated graphs retain the essence of the community structure of the original graphs.

In order to compare HyGen, we also generated graphs using LFR and DC-SBM. Based on our experiments in the previous section, we know that they cannot model the hyperbolic structure too well, but it is still possible that they can model all of the structure that is observed in real-world graphs.

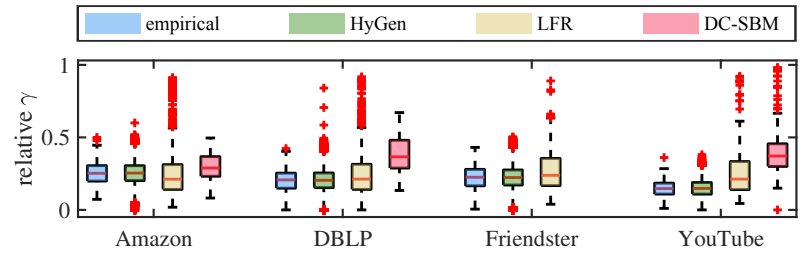
In order to fit the hyperbolic model to the generated communities, we need to know what these communities are. Both HyGen and LFR return this information; the DC-SBM implementation we used [75], on the other hand, only uses this information internally and does not report it. In order to find the communities from graphs generated by DC-SBM, we fit the model again to the generated graphs, and record the community structure from the fitted models. We assume that DC-SBM correctly recovers communities when provided graphs generated by it.

[75]: Nepusz (2015), *Blockmodel: Fitting stochastic blockmodels to empirical networks*

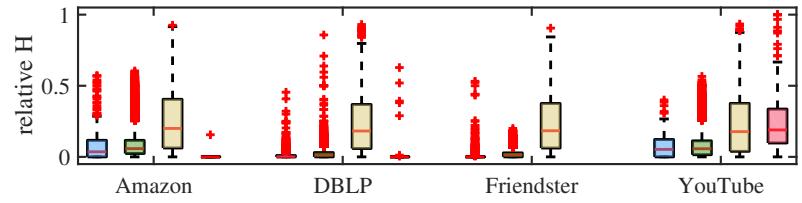
The results of this experiment, with respect to γ, H, and community size n_C, are presented in Figure 7.6. It shows four boxplots for each dataset and parameter combination: the first, called *empirical*, is the distribution observed from the original data,[9] and the other three boxplots show the distributions of the parameters when fitted to the generated graphs.

9: This is the distribution we should match.

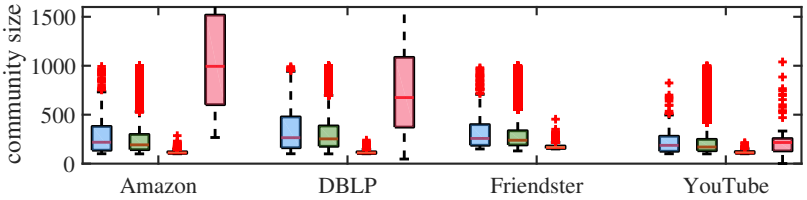
In order to obtain reliable results, with HyGen and LFR, we sampled a hundred times as many communities as was in the original data. That is, for the YouTube data, we sampled 12 900 communities, and for the other datasets, we sampled 50 000 communities. DC-SBM, on the other hand, was so slow to generate the sampled graphs and fit to them that we had to limit it to one graph with 100



(a) Distributions of the relative core sizes γ_R.



(b) Distributions of the relative tail heights H_R.



(c) Distributions of community size n_C (data for DC-SBM > 1600 not displayed).

Figure 7.6: Distributions of parameters in generated graphs compared to the empirical distributions observed in the original data. H and γ are obtained after fitting hyperbolic models.

communities for each dataset. Even with this limitation, it could not finish the Friendster data set within a week, and hence we exclude DC-SBM from the results regarding the Friendster data.

The results in Figure 7.6a show that HyGen generates a very good match for γ_R in all datasets. LFR is somewhat accurate, but has much higher SD and often generates too many communities with too large cores. This behaviour is most pronounced in the YouTube dataset. DC-SBM generates communities with even larger cores, and in case of DBLP and YouTube, the first quartile of γ_R in DC-SBM generated communities is above the third quartile of the original distribution, indicating a very bad fit.

Figure 7.6b shows the results for the relative H. Again, HyGen produces the most accurate results, although the communities have a slightly thicker tails than in the original data at least in the DBLP and Friendster data sets. On the other hand, these data sets have extremely thin tails in their communities. The communities generated by the LFR model have much thicker tails than the real communities or those generated by the HyGen model. In short, it is obvious that LFR cannot model the kind of thin tail real-world data sets often have. DC-SBM, on one hand, under-estimates the tail thickness for the Amazon data set, and on the other hand, over-estimates it for the YouTube data set.

When we look at the sizes of the generated communities, in Figure 7.6c, we can observe that LFR has very small deviation of the community sizes, and they are generally too small, while DC-SBM generates too large communities for Amazon and DBLP data sets, but approximately correctly-sized communities for the YouTube data set. The communities generated by HyGen have again the best fit in the distribution of the sizes, though it seems to generate slightly less of the larger communities than what is seen in the real data.

Overall, we can conclude that HyGen provides a reasonably good fit for the original data, and significantly better than what is provided by either LFR or DC-SBM. This again shows that HyGen is the best method for modelling hyperbolic structure, and that the real-world data actually has communities with structure that cannot be captured by the other models.

7.5 Randomness of the Generated Graphs

We now study whether the generated graphs are random enough. In a way, this experiment tests the opposite of the previous experiment; it would be easy to obtain a very good match to the original graph by simply generating graphs that are identical copies of the original one, but these would be rather useless. Hence, it is important to study also whether the generated graphs have enough randomness.

For this study, we quantify the randomness using the *conditional entropy* $\mathcal{H}(Y \mid X)$. Intuitively, it measures how much information is needed to describe random variable Y given that we know random variable X, that is, how much X 'tells about' Y. The entropy of Y [76, Ch. 2.1] is $\mathcal{H}(Y \mid X) = 0$ if Y is fully determined by X, $\mathcal{H}(Y \mid X) = \mathcal{H}(Y)$ and if Y and X are independent. As $0 \leq \mathcal{H}(Y \mid X) \leq \mathcal{H}(Y)$, we report the *relative conditional entropy*

[76]: Cover et al. (2006), *Elements of Information Theory*

$$\mathcal{H}_R(Y \mid X) = \frac{\mathcal{H}(Y \mid X)}{\mathcal{H}(Y)} \in [0, 1] \,. \tag{7.1}$$

We use the relative conditional entropy to compare the adjacency matrices of the different communities (original and generated). As the graphs are undirected, we will only study the upper triangular part of the adjacency matrix. We sort the rows and columns of the adjacency matrix according to the vertex degree so that they are ordered in similar way for all graphs. We then consider the upper triangular part of the adjacency matrix as a binary vector, and identify that as a discrete random variable.

To generate the data to compute $\mathcal{H}_R(Y \mid X)$, we create 100 random graphs using HyGen. We sample γ and H from fitted distributions, but for the size, we use the original sizes of the communities to ensure that the random variables X and Y are of same length.

The relative conditional entropy $\mathcal{H}_R(Y \mid X)$ is computed per community because communities sampled from the original graph do not maintain their context,[10] and thus we cannot match the generated communities correctly to the original communities unless we generate the communities one-by-one. In addition, using the fixed sizes of communities introduces determinism not present in the full graph model that could bias the analysis.

10: With the sampling, we loose the information which community overlaps with which community.

Our results are presented in Table 7.3. The results indicate that the information we have on the generated graphs given the original data is very small (all relative conditional entropies are larger than 0.96), that is, the generated communities are truly random. Together with the previous experiment, showing that the generation preserves the desired structure, we can conclude that HyGen can generate random graphs that preserve the desired structure.

Table 7.3: Average relative conditional entropy of generated communities given the original data. Standard deviations are within the displayed precision and thus omitted.

	$\mathcal{H}_R(Y \mid X)$
Amazon	0.996
DBLP	0.996
Friendster	0.990
YouTube	0.963

7.6 Modelling Time-Evolving Communities

We now examine how the graphon version of HyGen can be used to model time-evolving communities. Anticipating the main result of the next chapter, communities in social networks, especially in online question–answer sites, have a surprisingly constant relative core size.[11] Thus, the graphon model should be a good model for such communities. The purpose of this experiment is to study whether that is true; in particular, whether the graphon model can generate time-evolving communities that behave similarly to the real ones.

11: In Chapter 8, we will see that the relative γ, γ_R, stays almost constant over time.

In these experiments, we use the four communities from the StackExchange collection. Notice that here each dataset is just one community; this is sufficient for this experiment, as multi-community graphs would not change the behaviour of single communities. We initialise a graphon for each community using the relative parameters γ_R and H_R from Table 7.1 and sample a new community for every month in the data. We do not model the size of these communities, but use the real sizes. Similarly, we keep the same number of nodes from the previous month as was kept in the real data. Also, we sample without adding the noise. This way we can concentrate on the shape of the community; the modelling of the size, or the amount of the overlap, over time is an interesting problem for future work[12] and adding the noise will

12: As Figure 7.7 shows, there is no common trend in the sizes.

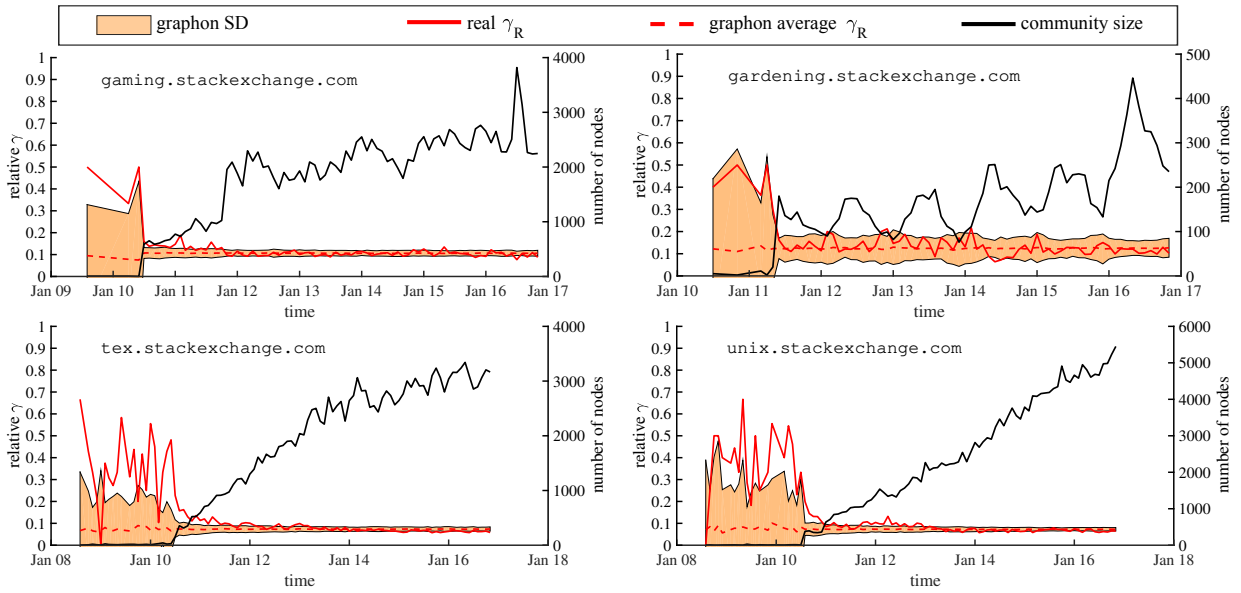


Figure 7.7: Behaviour of γ_R in graphon models of different communities. With the red dashed line, we display the average γ_R over 100 sampled communities; the red shading corresponds to the area two SDs above and below the average γ_R. The scale for the community size is on the right.

not change the analysis of the shape. As the tail height parameter H_R is almost zero for every community,[13] we do not report any results on that one.

Figure 7.7 shows the results for the four StackExchange communities. The red solid line shows the true γ_R for each month. The dashed line shows the average γ_R computed from 100 samples from the graphon, and the shaded area extends two SDs above and below the average (though clipping at 0).[14]

As we can see in the red lines, all communities have much higher values of γ_R at the beginning of their lives but soon the values converge to a lower value and stay rather constant. The biggest reason for the behaviour at the begin seems to be the size of the community (depicted as black line in the figures): initially all communities start small, but as they grow, the relative core size γ_R stabilises. This can be readily seen, for instance, from the gardening community, that has a cyclic pattern in its size: the value of γ_R varies the most when the community has the smallest sizes.

The graphon model can capture this variance based on size very well. The larger the sample (*i.e.* the community), the smaller the SD around the expected core size γ_R. Consequently, the samples follow the behaviour of the real graphs very closely, having much higher deviation at the early stages of the communities and stabilising as the community grows. Notice that the model even follows the individual peaks very well (for example in the unix community), even if the initial values are usually more than two SDs away from the average.

13: We will see in Section 8.4 that this is the case with essentially all communities like this.

14: Notice that in Figure 7.7, compared to the plots in Figure 8.6, we display the entire timeline, while later, for improved clarity, we omit very small communities and only display results with $n_C \geq 20$.

Overall, this experiment shows that the graphon model can be used very effectively to model time-evolving hyperbolic communities, even though there are some important future directions to explore. The current model selects the members that leave the community uniformly at random; potentially more realistic model would depend, for instance, on the length of the node's membership in the community and on its degree. Also, we did not model the community size over time. In order to generate fully synthetic graphs, this is a very important feature, although as can be seen already from Figure 7.7, different communities can have so different behaviour that a single distribution is probably never sufficient to model all of them.

7.7 Discussion

Our experiments demonstrate that HyGen is able to produce realistic intra-community structure. In its graphon formulation, HyGen is particularly well-suited for modelling time-evolving graphs.

While HyGen is already an improvement over the state-of-the-art random graph generators, there are still important topics of further development. The first important topic is to incorporate more realistic noise models to HyGen. At the present, the model assumes uniform noise with different probabilities of eliminating real edges and adding spurious ones. Our experiments, however, indicate that the noise is correlated with the size of the community. Incorporating a size-dependent noise model for removing the true edges is somewhat straight forward, but modelling similar noise for inter-community edges requires future work.

The HyGen-generated communities have currently too thick tails compared to what we see in the real world. This might be because the distribution we use to model the tail thickness parameter (Exponential) is not concentrated strongly enough, or it might have something to do with the noise model.

Finally, HyGen produces non-overlapping communities. In principle, this could be solved relatively easily: the HyGen algorithm (Algorithm 2) could generate partially-overlapping communities assuming it knows the amount of overlap. This could be either provided indirectly by specifying number of nodes for the output graph G, or by specifying the amount of mixture among communities. The real challenge however is not the definition of such a model, but its evaluation. Available test data from real world networks only comes with community information with respect to the nodes. Assuming the hyperbolic model, overlap can either

be within the intra-community area, or outside. In both cases, we would observe overlapping nodes, but only in the first case the communities actually overlap. Due to the lack of data to evaluate the realism of generated graphs with overlapping communities, we leave this extension of the model for future work.

HyGen has its obvious use for testing community detection algorithms. It can generate realistic graphs equipped with reliable labelling of the communities. Besides this, HyGen might serve as an anonymisation tool to study the structure of social networks without revealing the participants identities.

Hyperbolic Communities on Question–Answer Sites

8

In this chapter, we use the hyperbolic community model to study the community structures of online question–answer sites over time. Users of such sites donate their time and effort voluntarily to the community. In return, they gain visibility within the community through votes by other users. We show that the amount of active members is a constant fraction of the entire community throughout its lifetime.

8.1 Online Communication, Volunteer Effort, and Large Networks Under Study

We investigate the particular dynamics of interactions between people in online communities on question–answer sites. Besides the textual content that users provide, they leave traces of their interactions, such as who responds to whom at which time. We study freely available question–answer data from the large popular sites `reddit.com`, `stackexchange.com`, and `healthboards.com`. While `reddit.com` is a social news aggregation site with a very broad spectrum of topics, `stackexchange.com` is known for its free expert advice for the user asking a question, from which the entire community profits as well. Similarly, on `healthboards.com`, experts answer to questions of laymen regarding health topics.

Our primary result is the identification of a unifying pattern present in all examined groups: *the amount of active members is a constant fraction of the entire community throughout its lifetime*. Furthermore, through our analysis we conduct a large-scale assessment of the volunteer effort in online social communities, indicating that the active participation of only few community members within a group is a general organisational principle. Since online communication serves a social function [25, 77–79], this result is relevant for social communities in general.

While our analysis has a quantitative character, prior studies have analysed *who* these people are who volunteer for clubs, charities, or other organisations [36, 80], and who are the contributors of Internet content [81, 82]. Their motivations and backgrounds were the focus of these studies. In particular in the online world, contributions by people on a voluntary basis drive many communities. Wikipedia is one popular example of a community-driven project

[25]: Wellman et al. (1996), 'Computer Networks as Social Networks: Collaborative Work, Telework, and Virtual Community'
[77]: Baym et al. (2004), 'Social Interactions Across Media: Interpersonal Communication on the Internet, Telephone and Face-to-Face'
[78]: Arnaboldi et al. (2013), 'Egocentric online social networks: Analysis of key features and prediction of tie strength in Facebook'
[79]: Dunbar et al. (2015), 'The structure of online social networks mirrors those in the offline world'
[36]: Reed et al. (2001), 'The Civic Core in Canada: Disproportionality in Charitable Giving, Volunteering, and Civic Participation'
[80]: Nesbit et al. (2012), 'Patterns of Volunteer Activity in Professional Associations and Societies'
[81]: Bruns (2008), *Blogs, Wikipedia, Second Life, and Beyond: From Production to Produsage*
[82]: Hargittai et al. (2008), 'The Participation Divide: Content creation in the digital age'

to which everybody can contribute. In contrast to the kind of user interactions we focus on, users of Wikipedia collaboratively edit the same document. They may even revert the changes of each others. It has been shown that a large fraction of the content on Wikipedia is created by only a few highly active users [83, 84], which is in accordance with observations of volunteer efforts in other contexts [36, 80]. While this inequality in participation is an important principle and promotes productivity by forming leadership structures [83, 85], it might also be discriminative and demotivating for new contributors since it can be hard to compete with the well-established leadership core [86]. It is exactly this competition for fame and personal visibility has also been found to be one of the central motivations for volunteering in online communities [87]. With our study, we identify the most active contributors of the analysed sites. We, however, gather no personal information about the users but rather study their interaction patterns on a large scale.

Overall, the dynamics within groups of individuals are a long-running research topic [33–35, 38]. But since only recently, with the increasing popularity of the Internet, large data sets can easily be acquired for analysis: Two decades ago researchers presented hypotheses about social communities after interviewing around 300 people multiple times within one year [35]. Nowadays, such a dataset seems tiny. As millions of people leave traces of their interactions on the Internet, the availability of online network data has enabled numerous studies on the organisation of networks and the communities therein [23, 37, 88–93].

In our work, we study the intra-community structure of various communities from three different online question–answer sites using the hyperbolic community model. We regard this model as especially suitable to analyse the intra-community structure of social communities over time because it provides very intuitive parameters to summarise the connectivity pattern in every time step. We analyse which portion of the members is actively participating in discussions and how stable this core fraction is over an extended period of time.

Especially in the early formation phase of a community, with new members joining month for month, our expectation was to see characteristic formation patterns. For example, Kumar, Novak, and Tomkins [94], when studying the density of the network as a measure for the connectivity of every person in friendship networks, observed rapid growth followed by a decline and then slow but steady growth. In contrast, we find that for question–answer networks, the overall structure of a community remains

[83]: Matei et al. (2015), 'Pareto's 80/20 law and social differentiation: A social entropy perspective'
[84]: Ortega et al. (2008), 'On the Inequality of Contributions to Wikipedia'
[85]: Kittur et al. (2008), 'Harnessing the Wisdom of Crowds in Wikipedia: Quality Through Coordination'
[86]: Haklay (2016), 'Why is participation inequality important?'
[87]: Butler et al. (2007), 'Community Effort in Online Groups: Who Does the Work and Why?'
[33]: Laumann et al. (1976), *Networks of Collective Action: A Perspective on Community Influence Systems*
[34]: Alba et al. (1978), 'Elite Social Circles'
[35]: Morgan et al. (1997), 'The stability of core and peripheral networks over time'
[38]: Corradino (1990), 'Proximity structure in a captive colony of Japanese monkeys (Macaca fuscata fuscata): An application of multidimensional scaling'
[23]: Leskovec et al. (2005), 'Graphs over Time: Densification Laws, Shrinking Diameters and Possible Explanations'
[37]: Panzarasa et al. (2009), 'Patterns and dynamics of users' behavior and interaction: Network analysis of an online community'
[88]: Leskovec et al. (2008), 'Statistical properties of community structure in large social and information networks'
[89]: Cattuto (2006), 'Semiotic dynamics in online social communities'
[90]: Ahn et al. (2007), 'Analysis of topological characteristics of huge online social networking services'
[91]: Newman et al. (2003), 'Why social networks are different from other types of networks'
[92]: Sekara et al. (2016), 'Fundamental structures of dynamic social networks'
[93]: Zhang et al. (2017), 'comeNgo: A Dynamic Model for Social Group Evolution'
[94]: Kumar et al. (2006), 'Structure and evolution of online social networks'

very stable throughout its lifetime. A community might gain or lose active members, but relative to its size, the amount of core contributors remains nearly constant.

This observation raises the question as to *why* there is such a unifying organisational schema. We find that roughly 20 per cent of the members form the active core. This finding might hint towards deeper organisation principles. The 80–20 rule named after Pareto [95] resembles our result. This rule, however, does not allow for a hypothesis towards an underlying generative process to explain human social behaviour in general. Also, the extent to which this result is transferable to human social interaction in general remains an open question.

[95]: Koch (2011), *The 80/20 Principle: The Secret of Achieving More with Less*

We believe that our modelling approach, using concise descriptions of intra-community structures at a high level, leads to a better understanding of network structures within social communities and thereby promotes the comprehension of the underlying processes of social interaction.

8.2 Datasets

We examine meta-information of three large online discussion sites; `reddit.com`, `stackexchange.com`, and `healthboards.com`. We are interested in identifying the active users within the different communities of these sites. Therefore we collect the user interactions, that is, who has replied to whose post at which time. This information reveals the interactions between users and allows for the construction of user interaction graphs for each data set. Every obtained graph denotes the users as nodes and the interactions, labelled with times of occurrence, as edges. A summary of the employed data sets and their characteristics are displayed in Table 8.1. All data is publicly available. Further details on the data preparation are provided in Appendix C.1.

	Reddit	StackExchange		HealthBoards
		forums	meta-forums	
covered time span	2005–2016	2008–2016		1999–2013
nodes	$\sim 230 \times 10^6$	$\sim 8 \times 10^6$		338 079
communities	635 048	160	160	235
used communities	6 056	147	117	222
max. community size	155 511	219 693	30 223	18 924
avg. community size	2 774	15 124	849	3 015
avg. time span (months)	42	58	58	128

Table 8.1: Characteristics of the datasets. The statistics about the community size refer to the number of communities considered in this study and are reported with respect to their number of nodes. We require communities to have more than 100 nodes in total and to cover a time span of more than 12 months.

Reddit. The largest of the analysed sites is reddit.com. Reddit is an American social news aggregation and discussion website. As of the end of 2016, it encompasses 635 048 different topics, called *subreddits*. In every subreddit, users can open new discussion threads where other users can comment. From these threads, we gather the information who answered whom and when and construct a labelled undirected graph. Users constitute the nodes of this graph. There is an edge between two nodes if a user replied to another user (ignoring self-edges). Every edge is labelled with the date of the interaction. Furthermore, we group the nodes into communities by the subreddit under which the users contributed something.

Notice that we do not analyse whether the initiator of a discussion thread actually poses a question. While StackExchange and HealthBoards have a clear question–answer structure, Reddit is a discussion board and may well have threads where no question is posed or answered. This difference in the type of content, in addition to its size, makes the dataset particularly interesting for the present study: The fact that a post encourages other users to reply and engage in a discussion allows us to analyse this dataset in the same way as we do the pure question–answer sites. At the same time, we might expect to observe differences in the results due to the difference in the type of content.

StackExchange (SE). We likewise obtained an undirected labelled graph from the stackexchange.com page [74]. StackExchange is composed of question–answer websites on topics in various fields, the most prominent being related to computer programming and system administration. As of the end of 2016, there are a total of 160 different topics, each of them with a meta-discussion board. The derived graph contains the users as nodes. There is an edge between two nodes if a user replied to a question of another user or if a user commented on a post (can be question or answer) of another user. Every edge is labelled with the timestamp of the interaction. In addition, to define the communities, we group the nodes with respect to the topic under which they made a contribution (for instance 'gardening', depicted Figure 8.1).

[74]: Stack Exchange, Inc. (2016), *Stack Exchange Data Dump*

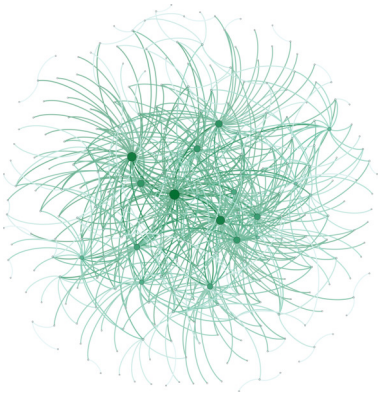


Figure 8.1: The gardening community from StackExchange in June 2016 with 279 users, visualised as Fruchterman–Reingold drawing.[1]

1: We used Gephi [96] for the visualisation.

HealthBoards (HB). healthboards.com is a long-running message board for patient to patient health support. It consists of 235 message boards for different health-related topics. This data is orders of magnitude smaller than the aforementioned resources. However, it is particularly interesting because not only the formation of communities can be observed, but also the dissolving phase where user activity gradually declines. The graph we derive has the users as nodes. Edges are formed through every

answer of one user to a thread opened by another user and are annotated with the timestamp of the interaction. The nodes are grouped in communities according to the message boards where they posted.

8.2.1 Modelling Decisions

We use monthly intervals to discretise the time line for the analysis. This choice trades off between a fine-grained view on the evolution of the communities and keeping the amount of models to compute and evaluate within a feasible range.

We consider the subgraph of every community individually instead of modelling the whole graph at once, as (1) our focus are the intra-community interactions, (2) for many communities there is very little overlap, and (3) the partitioning helps to keep the computational complexity manageable, in particular the memory consumption.

We experimented with how to accumulate the data for the monthly time steps. Options are for every time step (1) to accumulate all interactions from the beginning of the time series to the current time step, (2) to use a sliding window in order to accumulate over the last few months until the current time step, or (3) to only take the interactions of the respective month as edges. The results presented subsequently were obtained employing the latter option, which exhibits the highest variance and, thus, is the hardest case for what we aim to show. The other two options yield denser subgraphs, and because there is overlap between the data of consecutive time steps, the course of the core size parameter γ is much smoother. As we detail in Appendix C.3, we obtain similar results with respect to the relative shapes of the communities with these set-ups.

We report only results for communities with at least 100 nodes in total and a covered time range of more than 12 months.

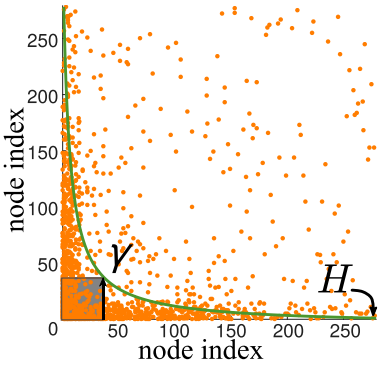


Figure 8.2: The gardening community from StackExchange in June 2016 visualised as degree-ordered adjacency matrix where dots indicate edges between nodes i and j. The green curve indicates the fitted hyperbolic model. The parameter γ denotes where $i = j$, and we have $\gamma = 38$ core members (shaded in grey). The tail H thins down to two members.

8.3 Model Suitability

We use the hyperbolic model (as introduced in Section 4.1) for our study of the development of the connectivity patterns within the communities of question–answer sites. For easy interpretability of the model, we parametrise the model into $\Theta = \{\gamma, H\}$, where the parameters γ and H can, respectively, be interpreted as the *size of the core*, and as the *thickness of the tail* (see Figure 8.2).

We begin our analysis with an assessment of the suitability of the chosen model: in this section, we show that the hyperbolic model is better suited than traditional quasi-clique models to explain the data, and that the hyperbolic models are robust, meaning that moderate changes to the parameters also only have a moderate impact on the model quality.[2]

2: All computations have been carried out using Matlab. The code for the hyperbolic community model is publicly available at https://cs.uef.fi/~pauli/hybobo/.

8.3.1 Comparison to Block Model

To validate that the hyperbolic community model is a good way to describe the analysed data sets, we compare it to the commonly used alternative, a quasi-clique model. A quasi-clique, or in other words, a uniformly dense block, is a common description of a community in a graph. The hyperbolic community model includes this option as a special case (see Section 4.2), the core size γ is then equal to the size of the community. We want to validate that such a model is not capable of describing the data as well as the hyperbolic community model.[3]

3: This test is similar to our analysis in Section 6.3 and confirms our findings on different data.

From every community, from every dataset, we take the time step where the community is the largest. This usually coincides with the last time step. Only in HealthBoards user activity decreases towards the end of the time series of every community. Very small communities are susceptible to noise and are therefore not the focus of this analysis.

We use the log-likelihood to judge the description quality and carry out a likelihood ratio test [72, Ch. 10.6]. Like in Section 6.3, we test the null hypothesis H_0 is that all parameters are fixed versus the alternative hypothesis H_1 is that the parameters are not fixed.

[72]: Wasserman (2004), *All of Statistics: A Concise Course in Statistical Inference*

We find that all StackExchange communities and most communities from Reddit and HealthBoards are statistically significantly better explained by the hyperbolic block model (see Table 8.2). It should be noted that the cases where a hyperbolic model is not statistically significantly better than the block model coincide with extremely small communities, typically below 20 nodes at their maximum.

Table 8.2: Result of the likelihood ratio test between modelling the data as hyperbolic communities compared to modelling them as quasi-cliques. We report the percentage of communities that are better explained using the hyperbolic model, at significance levels $\alpha = 0.01$ and $\alpha = 0.05$.

	percentage	
$\alpha =$	0.01	0.05
Reddit	99.3	99.7
StackExchange	100.0	100.0
HealthBoards	94.1	95.5

8.3.2 Robustness of the Model

We analyse the robustness of the obtained hyperbolic community models. To that end, we analyse how the log-likelihood deviates if we alter the optimal value for the parameter γ by 10 % in each direction. Notice that the absolute log-likelihood of a community model is dependent on the size of the community. To compare among different communities, we normalise the log-likelihood by the number of possible edges within the community.[4]

4: We normalise by $\frac{1}{2}({n_C}^2 - n_C)$ for a community with n_C members.

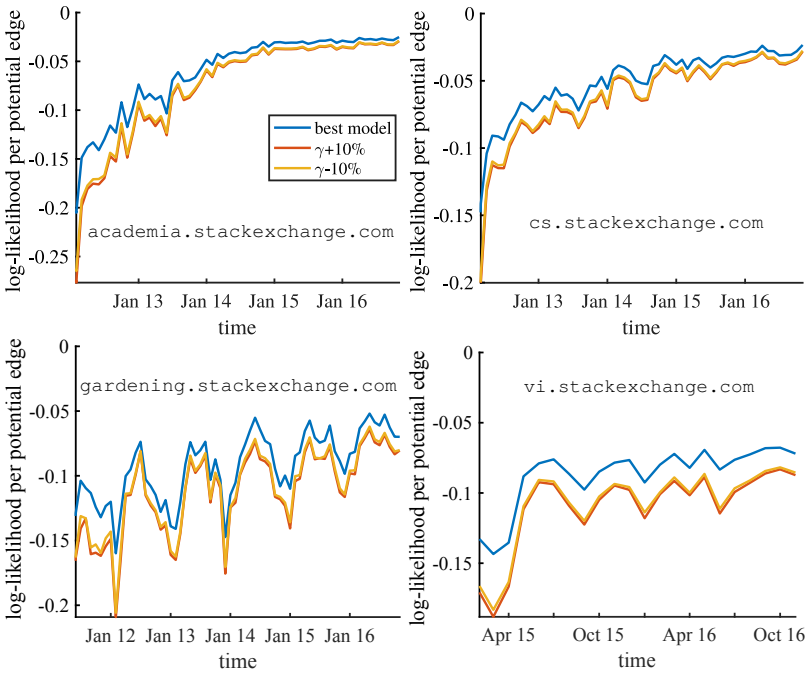
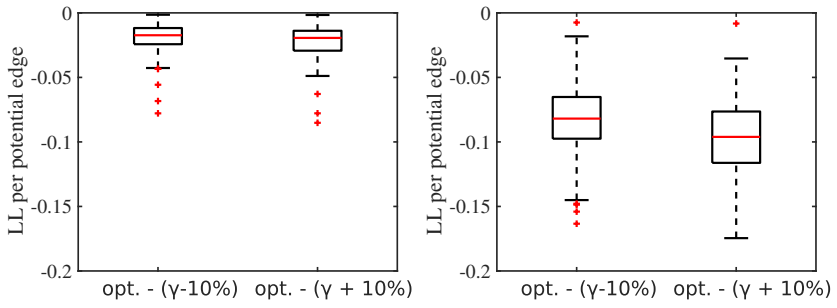


Figure 8.3: Evolution of the log-likelihood of the hyperbolic community model for the StackExchange communities and the log-likelihood when shifting the parameter γ by 10 % away from its optimum. The log-likelihood is reported relative to the number of possible edges within the community.

We observe that, indeed, the log-likelihood never improves when the parameter is shifted away from the optimum. More importantly, the log-likelihood worsens by a similar amount in every time step and in every community and for both directions of shifting γs.

To illustrate the development of the log-likelihood with the altered parameter γ over time, exemplary courses of the evolution are displayed in Figure 8.3. We summarise the differences in the log-likelihood, for StackExchange and its meta-communities separately, in Figure 8.4.[5]

When increasing γ by 10 %, we notice a drop of 0.022 on average for the non-meta StackExchange communities and 0.096 for the meta communities. Likewise, when decreasing γ by 10 %, the average drop is 0.019 and 0.083, respectively. We conclude that the hyperbolic community model is very robust and thus well-suited for our analysis.



(a) Communities of StackExchange. **(b)** Meta-communities of StackExchange.

5: Here again, boxplots display distributions such that in each box, the central mark is the median and the edges of the box are the first and third quartiles. The whiskers extend to the most extreme data points that are not outliers. Points are considered outliers if they are larger (smaller) than the third (first) quantile plus (minus) 1.5 times the difference between the quantiles. Outliers are plotted individually.

Figure 8.4: Differences in the log-likelihood (LL) per possible edge when applying the best hyperbolic community model versus shifting the parameter γ by 10 % away from its optimum. The distributions are the average log-likelihood differences per community.

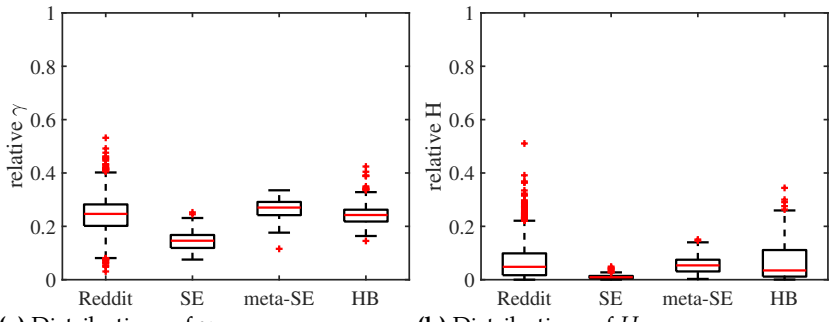
8.4 Fitting the Models

We now examine the results of fitting the hyperbolic model to the time line of every community. Optimising the log-likelihood as described in Section 4.5, we compute a hyperbolic model for every month for all communities from the question–answer sites; at the example of the gardening community, we display the model for a single time step in Figure 8.2. For a number of communities, we show the behaviour of the model parameters over time in Figure 8.6 and refer to Appendix C.8 for an overview of all time line plots.

We observe a predominant shape of the intra-community connectivity pattern: between 10 % and 30 % of nodes form the core; the rest of the nodes are loosely connected to the core area. As can be observed in Figure 8.6, and is examined in more detail subsequently, the value of γ is almost constant in this range over the full lifetime of the community. Surprisingly, we see this shape throughout the whole lifetime of every (sufficiently large) community, suggesting that the core of each community is invariant to its size and life time. Of course some fluctuations occur if there are very few community members in total, such as in the initiation phase of a community.

To quantify the characteristics of the observed shapes, we examine parameter distributions for each dataset. As summarised in Figure 8.5,[6] the distributions for the core size parameter γ predominantly range between 10 % and 30 %, while the tail H is mostly very thin. The cores of the StackExchange communities are even smaller, with their median size being 15 %. We regard separately the actual discussion boards of StackExchange and the meta-communities, where discussions about the respective board take place. Meta-communities are much smaller, their activities often vary heavily, and their members are a subset of the users of the respective basic discussion board. We find the cores of these communities to be 26 % of the community size on average, which is not even within the 90$^{\text{th}}$ percentile of the basic StackExchange communities. This is likely a result of the bias towards otherwise active users as participants in these discussions.

6: Within plots, we use the shorthand notations SE and HB to denote StackExchange and HealthBoards, respectively.



(a) Distributions of γ **(b)** Distributions of H

Figure 8.5: Distribution of the core size γ and the tail height H. Each box displays the distribution of averages over the time lines of each community in the respective data set. Further statistics are displayed in Table C.2.

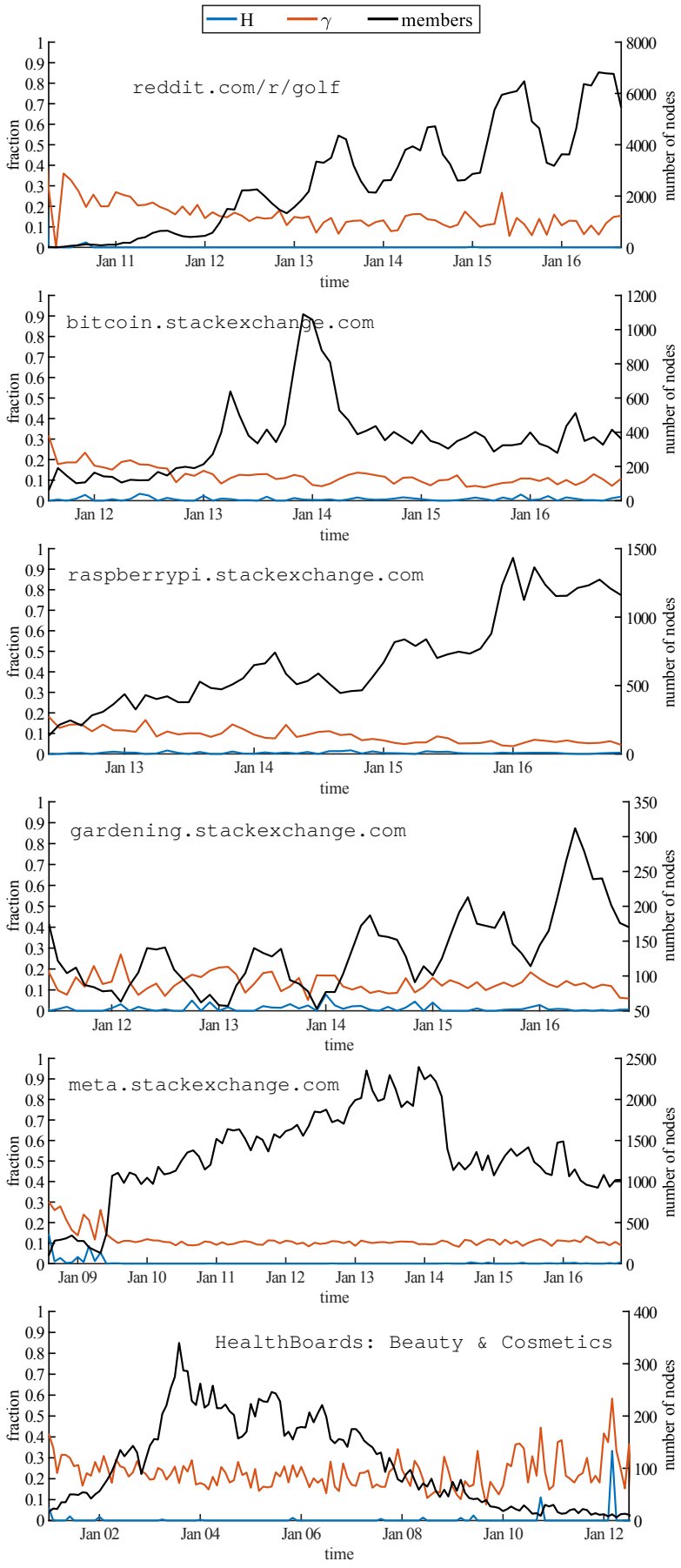


Figure 8.6: Examples of models for communities from the different question–answer sites over time. The model parameters H and γ obtained from fitting the hyperbolic community model are shown relative to the community size. The right axis in every plot denotes the absolute size of the community (black curve). Only data from communities with at least 20 members is displayed. The time lines form all data are displayed in Figures C.8 to C.15.

8.5 Regression on Time

Now, we analyse whether the seemingly constant relative size of the core of the communities is truly constant over time. To that end, we perform two tests.

First, we fit a linear model on each time line of γs. To assert that no quality is gained when using more complex models, we compare the fitting quality to that of higher order polynomials in Appendix C.4. As Figure 8.7a shows, we observe that the slope of the linear models is close to zero everywhere. Mostly large p-values (see Table C.2) provide no evidence to reject the null-hypothesis, which states that the slope is equal to zero.

Since this alone is no proof of the significance of our finding, we secondly measure the Pearson correlation coefficient between each time line of γ and a constant function. We find a strong correlation throughout the data with high significance. This statement may seem incoherent with the definition of Pearson's correlation coefficient: for a sample statistic, the sample correlation r between variables x and y is

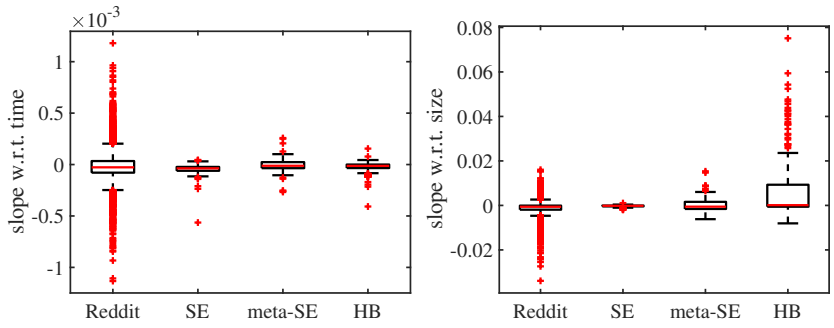
$$r = \frac{\sum_{i=1}^{n}(x_i - x)(y_i - y)}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

where x_i and y_i are the values of x and y for the i^{th} sample [72, Ch. 8.2]. In the case we describe, all y_i would be identical and in particular $y_i = \bar{y}$ which yields 0 in the denominator. Commonly, $r = 0$ is the defined outcome for this ill-defined case. To remedy this deficiency in the definition, we tilt the coordinate system prior to computing the correlation coefficient. To that end, we apply a coordinate transformation to all samples by multiplication with the rotation matrix

$$m = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \quad \text{where} \quad \alpha = \frac{\phi \cdot 2\pi}{360°}.$$

This way, we achieve a turning of the data by an arbitrarily chosen angle of $\phi = 45°$ and may compute the Pearson correlation coefficient in this transformed space. The argument for this procedure is that if there is no correlation and the data is just normally distributed along both axes, then no rotation will reveal any correlation. Vice versa, however, if there is any correlation with the flat line, then we can see it by rotation.

[72]: Wasserman (2004), *All of Statistics: A Concise Course in Statistical Inference*



(a) Time as x-axis. **(b)** Community size as x-axis.

Figure 8.7: Distribution of the slopes obtained from linear regression on every community.

The p-values of the correlation are computed for testing the hypothesis of no correlation against the alternative that there is a non-zero correlation. For every data set, we observe a significant amount of correlation, with no p-value larger than 0.000 001 (see Table C.2).

In particular, we notice that all examined datasets of the different question–answer boards show a similarly constant core size. While Reddit and StackExchange data range over less than five years on average and include mostly growing communities, the average HealthBoards communities have user interactions recorded over more than a decade. Even within such extended time periods, which include the formation and dissolving of communities, we have identified the constant core size as a unifying pattern.

8.6 Regression on Size

In this section, we study whether there is also no trend in the core size with respect to the size of a community. With this experiment, we aim to confirm what is observable in Figure 8.6: over time, the community sizes vary, but the sizes of community cores are invariant to community size.

To that end, we compute regression models for every community, like we did in the previous section, this time, however, with the community size instead of time on the x-axis. As Figure 8.7b shows, the relative core size is close to constant, also under variation of community size. We can therefore assert that the size of the cores is invariant to changes in the community size. Like for the regression with respect to time, we measure the Pearson correlation coefficient towards a constant function also in this setup. We again observe a strong correlation throughout the data with high significance (all p-values <0.000 001, compare with Table C.2).

As Figure 8.7b indicates, outliers are an order of magnitude more extreme in this set up and appear to be more prominent, especially in the HealthBoards data. A closer look at the data (see Appendix C.7 and, in particular, Figure C.7) reveals that the smallest of the communities constitute the outliers in these boxplots and are most likely explained by insufficient data to compute reliable hyperbolic models.

8.7 Stability of the Core

Our results above indicate that the relative *size* of the cores stays constant; however, we do not see whether the involved users inside the core vary strongly over time. To analyse how constant the fraction of core members stays from one time step to the next, we measured the stability of the core using the containment index.

Given two sets of nodes, the containment index indicates which percentage of nodes from the smaller set is contained in both sets. Formally, given two sets of nodes, $\mathcal{G}$ and $\mathcal{H}$, the containment index $C(\mathcal{G}, \mathcal{H})$ is defined as

$$C(\mathcal{G}, \mathcal{H}) = \frac{|\mathcal{G} \cup \mathcal{H}|}{\min(|\mathcal{G}|, |\mathcal{H}|)} .$$

We observe that from one time step to the next, between 40 % and 60 % of the nodes are overlapping. Figure 8.8 displays the result per dataset and includes the comparison of cores that are two, four, and eight time steps apart. As expected, we observe a gradual decrease of the overlap. However, even between time intervals that are more than half a year apart, the overlap is more than 30 % for all datasets. This suggests that a substantial portion of the active members stay active for a longer period of time and might even be a conservative estimate, since the employed measure ignores users who take a pause and then return.

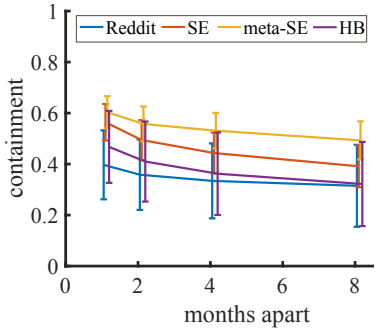


Figure 8.8: Core overlap between time-wise subsequent cores, and cores two, four, and eight months apart. The exact numbers can be found in Table C.2.

We further observe that the cores of the StackExchange communities and, in particular, of the meta-discussion boards are more stable than those of Reddit or HealthBoards. This suggests that active StackExchange users generally stay more committed to their community, and, in particular, those users who discuss their community on a meta-level adhere to their communities. Reddit users, on the other hand, are less devoted to their communities, called *subreddits*. This is logical, as new subreddits can be opened easily, and many of them are about arbitrary topics relating to current events.

For both, HealthBoards and StackExchange, we observe a stronger decline in the fraction of steady members when increasing the time interval than we do for Reddit or the meta-forums of StackExchange. We hypothesise that this is due to the expert-layman interaction nature in these two boards: a layman comes, discusses his matters with the experts, and leaves again.

An interesting direction of future research is to characterise the active users who constitute the core, and especially those who stabilise the community over time. For StackExchange, where each user has a *reputation* score that reflects expertise, we analyse to what extent high reputation is correlated to users in the cores of communities. As we detail in Appendix C.5, high reputation alone is, however, not a reliable indicator of the activity of a user.

8.8 Discussion

Modelling the interaction graphs of various communities of different online question–answer sites, we observe very similar patterns that remain stable over the lifetime of every community. We find that a constantly small fraction of the members constitutes the active core of each community, donating their time and effort voluntarily to the entire group.

Placement with respect to earlier work. One might think that particular communities have a substantially higher fraction of active members than others. Or, communities in the early phase, during their formation, might initially start as a block-like structure, where everybody interacts with everybody else. This phase might then be followed by a star pattern, as new members join and connect first to the old members. Such a formation pattern has been reported by Kumar, Novak, and Tomkins [94] for the social networks Flickr and Yahoo! 360, but cannot be confirmed for the analysed data sets.

[94]: Kumar et al. (2006), 'Structure and evolution of online social networks'

Arguably, it is a matter of conjecture whether differences in the data analysis disguise such patterns or whether they are truly absent. Kumar, Novak, and Tomkins, as well as Leskovec, Kleinberg, and Faloutsos [23] base their findings on the density observed in whole networks, whereas we study the hyperbolic community model. The major difference is that we consider only those nodes that have interactions with others while Kumar, Novak, and Tomkins consider every node irrespective of the connectivity. It is likely that the pattern they reported is caused by singleton users signing up and not immediately building up connections. Furthermore, Kumar, Novak, and Tomkins use discretisation based on weeks'

[23]: Leskovec et al. (2005), 'Graphs over Time: Densification Laws, Shrinking Diameters and Possible Explanations'

granularity; we use months. However, since their detected patterns occur within a period of 40 weeks, they should also be visible when studying monthly patterns.

Another important distinction between our study and earlier work is that in previous representations of social networks [*e.g.* 23, 94], an edge corresponds to a *passive* friendship relation. That means, adding more friends to someone's network does not, per se, increase the workload of this person. This is in contrast to our setting, where each edge represents *active* participation in the question–answer site, and especially in case of complex answers, can involve significant time investment. Such active commitment, done at least partially for altruistic reasons [80, 87], is likely to follow different dynamics than the mostly passive inclusion of friends.

One possible organisational principle underlying the networks studied here is the widely applied Pareto principle, or the *law of the vital few* [95], which states that around 80 per cent of the effects come from around 20 per cent of the causes. This principle matches observations in various fields, such as business and biology [97–99] and also explains communication in online discussion boards surprisingly well: about 20 per cent of the people are responsible for 80 per cent of the content in every community. The actual people who contribute actively might change over time, but the structural stability appears to be a property of large groups. Palla, Barabási, and Vicsek [100] emphasise that small groups, on the contrary, are susceptible to change and are likely to dissolve if their members change. This observation agrees with the increased amounts of variance we observe within small communities.

From the viewpoint of an individual, preceding studies [78, 79, 101–103] yield supporting evidence for more general organisation principles in human communication. Saramäki et al. [104] coin the term *social signature*, suggesting that most communication between individuals is limited to a small portion thereof. The frequent contacts of a person might change over time, but the overall distribution persists. The authors hypothesise that this is a consequence of finite resources, such as time available for communication and emotional capital. Following this thought, most active users in the examined question–answer sites would either have extremely frequent contact with their closest friends, or they have no further social circle. As either scenario seems unlikely, a subsequent qualitative analysis might be necessary to unify these results. Our study covers voluntary effort in the form of question answering. We assume that this serves as a proxy to communication behaviour in general. However, the conducted study is limited to the communication that people 'donate' to the examined sites. The large amount of studied data allows for

[80]: Nesbit et al. (2012), 'Patterns of Volunteer Activity in Professional Associations and Societies'
[87]: Butler et al. (2007), 'Community Effort in Online Groups: Who Does the Work and Why?'

[95]: Koch (2011), *The 80/20 Principle: The Secret of Achieving More with Less*

[97]: Fernández-Sánchez et al. (2010), 'A methodology to identify sustainability indicators in construction project management—Application to infrastructure projects in Spain'
[98]: Daly et al. (2011), *Ecological economics: principles and applications*
[99]: Jankowski et al. (2013), 'Birds Shed RNA-Viruses According to the Pareto Principle'

[100]: Palla et al. (2007), 'Quantifying social group evolution'

[78]: Arnaboldi et al. (2013), 'Egocentric online social networks: Analysis of key features and prediction of tie strength in Facebook'
[79]: Dunbar et al. (2015), 'The structure of online social networks mirrors those in the offline world'
[101]: Roberts et al. (2009), 'Exploring variation in active network size: Constraints and ego characteristics'
[102]: Dunbar (1992), 'Neocortex size as a constraint on group size in primates'
[103]: De Salve et al. (2016), 'The impact of user's availability on On-line Ego Networks: A Facebook analysis'

[104]: Saramäki et al. (2014), 'Persistence of social signatures in human communication'

statistically well-grounded hypotheses, but we cannot make holistic claims about the communication patterns from the perspective of an individual.

Data representation. To construct the studied interaction graphs, we use the response of one user to the posting of another as an indicator for a social interaction. For simplicity, these interactions are modelled dichotomously and we do not account for the direction of the interactions. Incorporating the latter would require a more complex community model in which donators and receivers are distinguishable. What further insights could be gained from such a modelling remains an open question. In the current approach, we seek for a very simple summary of the communities under study, describing their shape by just two different parameters. This allows us to summarise especially large data sets into a graspable format.

To model additional information about the strength of the edges, every interaction could be weighted, either by the amount of interactions in a given time interval, or by the elapsed time between a post and its response. Empirical examination of StackExchange communities indicates that the majority of connections, in the former setting, would show very low weights, mostly equal to 1 and, therefore, would not provide much additional information (see Appendix C.2). The reason is potentially the particular expert-layman interaction on this site. Whether more general discussion boards like on Reddit could benefit more from weighted interaction graph models, and how the time between interactions as an indicator of strength could be beneficial, are interesting directions of future research. Such an analysis would also require the use of a modified approach to model the communities. Our model is restricted to undirected, unweighted networks.

Use for finding discontinuities. We have seen that the relative amount of active members in a community remains constant over time. In particular, this holds while the actual size of the community might vary – either through growth in general or due to seasonal trends, such as in the gardening community of StackExchange, or the golf community of Reddit (first and fourth plot in Figure 8.6). Occasionally, however, seemingly independent of the community size, we observe short-term enlargements of the core size by about ten per cent . The Raspberry Pi community of StackExchange, for instance, shows such discontinuities around March 2013 and April 2014 (third plot in Figure 8.6). Coincidentally, these dates refer to release dates of new versions of the product. Unlike for the Bitcoin community (second plot in Figure 8.6), where the size of the community can easily be aligned with major events in the value

development of the currency, the Raspberry Pi release events do not show in the number of members, rather in their connectivity patterns. Our employed modelling framework might thus facilitate new methods for event detection.

In summary, our work provides new insight on the evolution of community structures in large networks. The examined online question–answer sites show a common scheme of roughly 20 per cent of highly connected active members and 80 per cent loosely associated members who mainly communicate with the active core. This scheme remains throughout the lifetime of a community.

9 Conclusion

This chapter concludes the first part of the thesis. We summarise our contribution towards modelling hyperbolic communities and discuss further directions for this line of research.

9.1 Summary

We have introduced the hyperbolic community model and its graph generator, HyGen. Our model captures non-uniform edge distributions within communities in networks. It enables a realistic, yet simple description of the building blocks of networks. We present the model in terms of three different but equivalent parametrisations, each with its own merits:

- `hyperbolic` employs a hyperbola equation and emphasises the geometric shape of the community.
- `fixed` uses the core size and tail height as parameters and thus gives an immediate intuition about the shape of the community.
- `mixture` offers a probabilistic perspective by expressing the community shape as a mixture of a line and a hyperbola.

In addition, we express the hyperbolic model in terms of a graphon, thereby offering the perspective of an exchangeable random graph model. Furthermore, we present a random graph generator on the basis of the hyperbolic model. HyGen generates modular networks with realistic intra-community structures, using parameter distribution derived from observations in real graphs.

In extensive experiments, we confirm that the hyperbolic community model explains real graphs much better than previous approaches, and that HyGen produces satisfactory modular random graphs with realistic intra-community structure.

In a large-scale study of volunteer effort on online question–answer sites, we successfully demonstrate the use of the hyperbolic community model. Our primary result is the identification of a unifying pattern, present in all examined data: *the amount of active members is a constant fraction of the entire community throughout its lifetime.*

9.2 Challenges

Notwithstanding the achievements towards modelling communities in networks more realistically, there are still challenges to meet along this line of work. The major point to name here is *how to find the communities* within a network.

[12]: Javed et al. (2018), 'Community detection in networks'

There is a wide range of approaches for community detection [12]. Overlapping or not, many of them regard communities simply as denser blocks in the data, and therefore aim to identify (quasi-) cliques. Modifying those methods to account for hyperbolic community structure is always possible as a post-processing step. As we demonstrate in Section 6.4, existing community detection algorithms can be improved by expressing the result using hyperbolic communities.[1] A favourable alternative would be to incorporate hyperbolic community structure already in the detection procedure. As opposed to searching for evenly dense blocks and expressing them with the hyperbolic model, the community detection would be tuned towards finding the hyperbolic structures right from the start.

1: We measure the improvement in terms of the log-likelihood of the solution.

Hyperbolic community detection faces the challenge that three different types of overlap between communities can occur (see Section 4.3): our model specifies not only which nodes of a graph constitute a community, but also which edges between these nodes belong to the community. Therefore, in addition to the usual distinction of having common nodes, or not, hyperbolic communities might be edge-disjoint despite having common nodes.

The latter case is particularly difficult to identify, since node-overlapping but edge-disjoint communities seemingly appear like a single community with one big core. If more detailed examination reveals bipartite intra-community structure, that would be a hint that actually two communities overlap each other. A Bayesian inference approach, where the best assignment of community memberships is determined in an MDL fashion, might serve as a basis for hyperbolic community detection. The starting point for such an approach could be the method of Peixoto [45], targeted at a hierarchical partitioning of DC-SBMs.

[40]: Karaev et al. (2018), 'Logistic-Tropical Decompositions and Nested Subgraphs'

2: An undirected graph $G = (V, E)$ is *nested* if we can order the vertices $v \in V$ in a sequence $(v_1, v_2, \ldots, v_n)$ such that $N(v_{i+1}) \subseteq N(v_i)$ for all $i = 1, \ldots, n-1$.

[41]: Neumann et al. (2016), 'What You Will Gain By Rounding: Theory and Algorithms for Rounding Rank'

Another promising direction for the development of a detection approach is offered by the recent works on nested matrices and rounding rank. As discussed by Karaev, Metzler, and Miettinen [40], hyperbolic communities are a special case of nested matrices.[2] Nested matrices have non-negative rounding rank 1 [41], which suggests that rounding rank decompositions could provide an approach for finding hyperbolic communities. Yet, the combination of nested graphs is problematic, since the characterisation from rounding rank falls apart in higher-rank decompositions. To

overcome this difficulty, Karaev, Metzler, and Miettinen propose a decomposition approach for overlapping nested matrices using *tropical algebra*.[3] Under tropical algebra, the rounding rank extends naturally to higher-rank decompositions. This approach, amended with appropriate constraints[4] could be suitable method for the detection of hyperbolic communities.[5]

No matter how promising an detection approach appears, the next challenge is its validation. Testing community detection approaches requires large amounts of reliable test data – in our case, with labels for the nodes *and* the edges. Indeed, we propose HyGen not least exactly to close this gap. However, independent real network data would still be an asset.

Yet another challenging direction is to explore the *detectability* of hyperbolic communities from a theoretic point of view. For (clique-like) community detection, there exists a resolution limit: as networks becomes too sparse, communities are undetectable. Although they still exist, it becomes information-theoretically impossible to identify them better than by chance, because the densities within and outside of communities are too similar. Nadakuditi and Newman [105] explore the conditions under which the detectability of communities vanishes with a focus on matrix-based methods for community detection, while Decelle et al. [106] especially analyse methods based on Bayesian inference. Both works regard networks generated by SBMs. The internal structure of communities in our approach differs that of SBM, but since we rely on differences in density to decide upon the community membership of nodes, it is likely that any approach to hyperbolic community detection also faces a similar resolution limit.

3: *Tropical algebra* is defined over $\bar{\mathbb{R}} = \mathbb{R} \cup \{-\infty\}$ with the operations

$a \boxplus b = \max\{a, b\}$ (addition) and
$a \boxdot b = a + b$ (multiplication).

The identity elements are $-\infty$ and 0, respectively.

4: Not every nested matrix is hyperbolic, but all hyperbolic matrices are nested.

5: We will return to this idea in Chapter 15.

[105]: Nadakuditi et al. (2012), 'Graph Spectra and the Detectability of Community Structure in Networks'

[106]: Decelle et al. (2011), 'Inference and Phase Transitions in the Detection of Modules in Sparse Networks'

Boolean Tensor Clustering

10 Introduction

We now switch topics. The second part of this thesis is about Boolean tensor clustering, and we start here with an introductory chapter, where we motivate the idea, outline our results, and discuss related approaches.

10.1 Tensors, Clustering, and Binary Data

Tensors are multi-way generalisations of matrices. They are common in data mining: three-way tensors can be used to record ternary relations – for example source IP–target IP–target port in network traffic analysis; a set of different relations between the same entities – for example subject–relation–object data for information extraction; or different instances of the same relation between the same entities – for example the adjacency matrices of a dynamic graph over time, such as who sent email to whom in which month.

Given such data, a typical goal in data mining is to find structure and regularities from it in order to create an informative summary that describes the data well. To that end, tensor decomposition methods are often employed. Their output are the building blocks, called factors, that together explain the data. For three-way tensors, a decomposition yields three factors; one for each direction in the data, referred to as mode. The factors are represented by matrices whose 'multiplication' yields the tensor again.[1] If the data are binary – as in all the examples above – it is natural to restrict also the factor matrices to the binary domain. It is also natural to consider the data to denote presence or absence of features, making Boolean algebra a natural choice. Hence we will study *Boolean tensor decompositions*.

1: We mean the multiplication in terms of the tensor product here. A brief introduction to terminology and tensor operations is in Section 11.1.

Finding good Boolean tensor decompositions is computationally hard. The hardness comes from the complexity caused by overlapping factors. In applications on real data sets, however, it has been observed that the returned factors often are (almost) non-overlapping in at least one mode [107, 108]. This means, the respective factor matrix has exactly one non-zero per row.

[107]: Erdős et al. (2013), 'Discovering Facts with Boolean Tensor Tucker Decomposition'
[108]: Erdős et al. (2013), 'Walk'n'Merge: A Scalable Algorithm for Boolean Tensor Factorization'

Typically the mode where these non-overlapping factors appear is unsurprising given the data: in the above examples, it would be target ports in the network traffic analysis data, relations in the

information extraction data, and time (month) in the email data – in all of these cases, this mode has a different behaviour compared to the other two.

Exactly these observations are the motivation for our approach: we restrict one mode to non-overlapping factors, thereby removing one of the main sources of computational complexity. We will see subsequently that the resulting problem admits better approximability results, and yields simpler algorithms with better results, compared to other Boolean tensor decomposition approaches. We refer to the problem as *Boolean tensor clustering (BTC)*, since the factor matrix with exactly one non-zero per row can be interpreted as a cluster assignment matrix.

If clustering is what we propose here, then why do we not simply cluster the slices of the tensor using any off-the-shelf clustering algorithm? The answer is two-fold: On one hand, having binary centroids to express binary data often improves the interpretability (and sparsity) of the results, and hence we want to develop specific algorithms for the Boolean case. On the other hand, just clustering the slices will not reveal the underlying structure *within* the slices; extracting that structure facilitates the interpretation of the clustering, and will also alleviate the curse of dimensionality, as we shall see in the experiments. With BTC, we seek for a concise representation of the major patterns in the data. Given the example of subject–relation–object data, this would amount to finding clusters of relations and describing them by the most important subjects and objects for each cluster.

10.1.1 Our Contribution

We present the problem of Boolean tensor clustering (BTC) in Section 11.2. The general form of BTC is in fact very similar to standard binary clustering problems, and therefore we concentrate on the more restricted problem of Boolean CP clustering (BCPC). While we approach these problems from the point of view of tensor decompositions, they can be equivalently seen from the clustering point of view: the task then is to find a clustering while the cluster centroids are constrained to comply with the tensor structure.

While we concentrate on three-way tensors, our approach can be extended to higher-order Boolean tensors. We discuss about application-specific design decisions this extension involves.

Normally, the quality of a clustering or tensor decomposition is measured using the distance of the original data to its representation. In Section 11.3 we argue, however, that for many data mining

problems (especially with binary data), measuring the quality using the similarity instead of distance leads to a more meaningful analysis.

Motivated by this, the goal of our algorithms is to maximise the similarity instead of minimising the dissimilarity, as we detail in Section 11.4. As a by-product of the analysis of our algorithms, we also develop and analyse algorithms for maximum-similarity binary rank-1 decompositions. We show that the problem admits a polynomial-time approximation scheme (PTAS) and present a scalable algorithm that achieves a $2(\sqrt{2}-1) \approx 0.828$ approximation ratio.

The experimental evaluation, in Chapter 12, shows that our algorithm achieves comparable representations of the input tensor when compared to methods that do (Boolean or normal) tensor CP decompositions – even when our algorithm is solving a more restricted problem – while being generally faster and admitting good generalisation to unseen data.

10.2 Related Work

Tensor clusterings have received some research interest in recent years. The existing work can be divided roughly into two separate approaches: on one hand Zhao and Zaki [109], Jegelka, Sra, and Banerjee [110], and Papalexakis, Sidiropoulos, and Bro [111] study the problem of clustering simultaneously all modes of a tensor (tensor co-clustering); and on the other hand, Huang et al. [112] and Liu et al. [113] (among others) study the problem where only one mode is clustered and the remaining modes are represented using a low-rank approximation. The latter form is closer to what we study, but the techniques used for the continuous methods do not apply to the binary case.

Normal tensor factorisations are well-studied, dating back to the late Twenties. The Tucker and CP decompositions were proposed in the Sixties [114] and Seventies [115, 116], respectively. The topic has nevertheless attained growing interest in recent years, both in numerical linear algebra and computer science communities. For a comprehensive study of recent work, see Kolda and Bader [117], and the recent work on scalable factorisations by Papalexakis, Faloutsos, and Sidiropoulos [118].

The first algorithm for Boolean CP factorisation was presented by Leenen et al. [119], although without much analysis. Working in the framework of formal tri-concepts, Bělohlávek, Glodeanu, and Vychodil [120] studied the exact Boolean tensor decompositions, while Ignatov et al. [121] studied the problem of finding dense rank-1

[109]: Zhao et al. (2005), 'triCluster: An Effective Algorithm for Mining Coherent Clusters in 3D Microarray Data'

[110]: Jegelka et al. (2009), 'Approximation Algorithms for Tensor Clustering'

[111]: Papalexakis et al. (2013), 'From *K*-Means to Higher-Way Co-Clustering: Multilinear Decomposition With Sparse Latent Factors'

[112]: Huang et al. (2008), 'Simultaneous Tensor Subspace Selection and Clustering: The Equivalence of High Order SVD and *K*-Means Clustering'

[113]: Liu et al. (2010), 'Hybrid Clustering of Multiple Information Sources via HOSVD'

[114]: Tucker (1966), 'Some mathematical notes on three-mode factor analysis'

[115]: Carroll et al. (1970), 'Analysis of individual differences in multidimensional scaling via an N-way generalization of "Eckart–Young" decomposition'

[116]: Harshman (1970), 'Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multimodal factor analysis'

[117]: Kolda et al. (2009), 'Tensor decompositions and applications'

[118]: Papalexakis et al. (2012), 'ParCube: Sparse Parallelizable Tensor Decompositions'

[119]: Leenen et al. (1999), 'Indclas: A three-way hierarchical classes model'

[120]: Bělohlávek et al. (2012), 'Optimal Factorization of Three-Way Binary Data Using Triadic Concepts'

[121]: Ignatov et al. (2011), 'From Triconcepts to Triclusters'

subtensors from binary tensors. In data mining, Miettinen [122] studied computational complexity and sparsity of the Boolean tensor decompositions. Recently, Erdős and Miettinen [108] proposed a scalable algorithm for Boolean CP and Tucker decompositions, and applied that algorithm for information extraction [107]. The tri-concepts are also related to closed n-ary relations, that is, n-ary extensions of closed itemsets [see, *e.g.* 123, 124]. For more on these methods and their relation to Boolean CP factorisation, see Miettinen [122].

Miettinen et al. [32] presented the discrete basis partitioning problem (DBPP) for clustering binary data and using binary centroids. They gave a $(10 + \varepsilon)$ approximation algorithm based on the idea that any algorithm that can solve the so-called binary graph clustering (where centroids must be rows of the original data) within factor f, can solve the arbitrary binary centroid version within factor $2f$. Recently, Jiang [125] gave a 2-approximation algorithm for the slightly more general case with $k + 1$ clusters with one centroid restricted to be an all-zero vector.

The maximisation dual of binary clustering, the hypercube segmentation problem, was studied by Kleinberg, Papadimitriou, and Raghavan [126, 127] and they also gave three algorithms for the problem, obtaining an approximation ratio of $2(\sqrt{2} - 1)$. Later, Seppänen [128] proved that this ratio is tight for this type of algorithms, and Alon and Sudakov [129] presented a PTAS for the problem.

Recently, Kim and Candan [130] proposed the tensor/relational model, where relational information is represented as tensors. They defined the relational algebra over the (decompositions of) the tensors, noting that in the tensor-relational model, the tensor decomposition becomes the costliest operation. Hence, their subsequent work [131, 132] has concentrated on faster decompositions in the tensor-relational framework.

[122]: Miettinen (2011), 'Boolean Tensor Factorizations'

[108]: Erdős et al. (2013), 'Walk'n'Merge: A Scalable Algorithm for Boolean Tensor Factorization'

[107]: Erdős et al. (2013), 'Discovering Facts with Boolean Tensor Tucker Decomposition'

[123]: Cerf et al. (2013), 'Closed and noise-tolerant patterns in n-ary relations'

[124]: Cerf et al. (2009), 'Closed patterns meet n-ary relations'

[32]: Miettinen et al. (2008), 'The Discrete Basis Problem'

[125]: Jiang (2014), 'Pattern extraction and clustering for high-dimensional discrete data'

[126]: Kleinberg et al. (1998), 'A Microeconomic View of Data Mining'

[127]: Kleinberg et al. (2004), 'Segmentation problems'

[128]: Seppänen (2005), 'Upper bound for the approximation ratio of a class of hypercube segmentation algorithms'

[129]: Alon et al. (1999), 'On two segmentation problems'

[130]: Kim et al. (2011), 'Approximate tensor decomposition within a tensor-relational algebraic framework'

[131]: Kim et al. (2012), 'Decomposition-by-Normalization (DBN): Leveraging Approximate Functional Dependencies for Efficient Tensor Decomposition'

[132]: Kim et al. (2014), 'Pushing-Down Tensor Decompositions over Unions to Promote Reuse of Materialized Decompositions'

11 Theory

In this chapter, we devise the theory. We give a brief introduction to tensors, reviewing properties and basic operations. We formalise the task of Boolean tensor clustering (BTC), discuss variations and related problems, and present an algorithm towards it.

[117]: Kolda et al. (2009), 'Tensor decompositions and applications'

[122]: Miettinen (2011), 'Boolean Tensor Factorizations'

11.1 Preliminaries

We start with a brief introduction to the basic notation and concepts relevant for the BTC problem. For detailed presentations of general tensor operations and decompositions, we refer the reader to Kolda and Bader [117], and to Miettinen [122] for the specifics of Boolean tensors.

11.1.1 Notation and Terminology

We indicate vectors as bold lowercase letters ($\mathbf{v}$), matrices as bold uppercase letters ($\mathbf{M}$), and tensors as bold uppercase calligraphic letters ($\mathcal{T}$). Element (i, j, k) of a three-way tensor $\mathcal{X}$ is denoted as x_{ijk}. A colon in a subscript denotes taking that mode entirely; for example, $\mathbf{X}_{::k}$ is the k^{th} *frontal slice* of $\mathcal{X}$ ($\mathbf{X}_k$ in short).

Fibers of a tensor are the higher order analogues of matrix rows and columns. They are obtained by fixing every index but one; for example, $\mathbf{x}_{:jk}$ is the j^{th} column of the k^{th} frontal slice of $\mathcal{X}$.

An N-way tensor can be *unfolded*, or *matricised*, into a matrix by arranging its fibers as columns of a matrix. For a *mode-n matricisation*, the mode-n fibers are used as the columns and the result is denoted by $\mathbf{X}_{(n)}$. In case of a three-way tensor, we refer to the fibers $\mathbf{X}_{(1)}$, $\mathbf{X}_{(2)}$, $\mathbf{X}_{(3)}$, as columns, rows, and tubes, respectively.

11.1.2 Right up to Decompositions

The *outer product* of vectors is denoted by $\boxtimes$. It generalises to tensors in the following way: for vectors $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$ of length n, m, and l, $\mathcal{X} = \mathbf{a} \boxtimes \mathbf{b} \boxtimes \mathbf{c}$ is an n-by-m-by-l tensor with $x_{ijk} = a_i b_j c_k$.

For a tensor $\mathcal{X}$, $|\mathcal{X}|$ denotes its number of non-zero elements. The *Frobenius norm* of a three-way tensor $\mathcal{X}$ is

$$\|\mathcal{X}\| = \sqrt{\sum_{i,j,k} x_{ijk}^2} \,. \tag{11.1}$$

If $\mathcal{X}$ is binary, we have $|\mathcal{X}| = \|\mathcal{X}\|^2$.

The *similarity* between two n-by-m-by-l binary tensors $\mathcal{X}$ and $\mathcal{Y}$ is defined as

$$\mathsf{sim}(\mathcal{X}, \mathcal{Y}) = nml - |\mathcal{X} - \mathcal{Y}| \,. \tag{11.2}$$

The *Boolean tensor sum* of binary tensors $\mathcal{X}$ and $\mathcal{Y}$ is defined as

$$(\mathcal{X} \vee \mathcal{Y})_{ijk} = x_{ijk} \vee y_{ijk} \,. \tag{11.3}$$

For binary matrices $\mathbf{X}$ and $\mathbf{Y}$ where $\mathbf{X}$ has r columns and $\mathbf{Y}$ has r rows their *Boolean matrix product*, $\mathbf{X} \circ \mathbf{Y}$, is defined as

$$(\mathbf{X} \circ \mathbf{Y})_{ij} = \bigvee_{k=1}^{r} x_{ik} y_{kj} \,. \tag{11.4}$$

The *Boolean matrix rank* of a binary matrix $\mathbf{A}$ is the least r such that there exists a pair of binary matrices $(\mathbf{X}, \mathbf{Y})$ of inner dimension r with

$$\mathbf{A} = \mathbf{X} \circ \mathbf{Y} \,. \tag{11.5}$$

A binary matrix $\mathbf{X}$ is a *cluster assignment matrix* if each row of $\mathbf{X}$ has exactly one non-zero element. In that case the Boolean matrix product corresponds to the regular matrix product,

$$\mathbf{X} \circ \mathbf{Y} = \mathbf{X}\mathbf{Y} \,. \tag{11.6}$$

We can now define the Boolean tensor rank and two decompositions: Boolean CP and Tucker3.

Definition 11.1.1 *The* Boolean tensor rank *of a three-way binary tensor* $\mathcal{X}$, $\mathrm{rank}_B(\mathcal{X})$, *is the least integer* r *such that there exist* r *triplets of binary vectors* $(\mathbf{a}_i, \mathbf{b}_i, \mathbf{c}_i)$ *with*

$$\mathcal{X} = \bigvee_{i=1}^{r} \mathbf{a}_i \boxtimes \mathbf{b}_i \boxtimes \mathbf{c}_i \,.$$

The low-rank Boolean tensor CP decomposition is defined as follows:

Problem 11.1.1 (Boolean CP) *Given an n-by-m-by-l binary tensor $\mathcal{X}$ and an integer r, find binary matrices $\mathbf{A}$ (n-by-r), $\mathbf{B}$ (m-by-r), and $\mathbf{C}$ (l-by-r) such that they minimise*

$$\left| \mathcal{X} - \bigvee_{i=1}^{r} \mathbf{a}_i \boxtimes \mathbf{b}_i \boxtimes \mathbf{c}_i \right| .$$

The standard (non-Boolean) tensor rank and CP decomposition are defined analogously [117]. Both finding the least error Boolean CP decomposition and deciding the Boolean tensor rank are NP-hard [122]. Following Kolda and Bader [117], we use $[[\mathbf{A}, \mathbf{B}, \mathbf{C}]]$ to denote the normal three-way CP decomposition and $[[\mathbf{A}, \mathbf{B}, \mathbf{C}]]_B$ for the Boolean CP decomposition.

[117]: Kolda et al. (2009), 'Tensor decompositions and applications'

[122]: Miettinen (2011), 'Boolean Tensor Factorizations'

Let $\mathbf{X}$ be an n_1-by-m_1 matrix and $\mathbf{Y}$ be an n_2-by-m_2 matrix. Their *Kronecker (matrix) product*, $\mathbf{X} \otimes \mathbf{Y}$, is the $n_1 n_2$-by-$m_1 m_2$ matrix defined by

$$\mathbf{X} \otimes \mathbf{Y} = \begin{pmatrix} x_{11}\mathbf{Y} & x_{12}\mathbf{Y} & \cdots & x_{1m_1}\mathbf{Y} \\ x_{21}\mathbf{Y} & x_{22}\mathbf{Y} & \cdots & x_{2m_1}\mathbf{Y} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n_1 1}\mathbf{Y} & x_{n_1 2}\mathbf{Y} & \cdots & x_{n_1 m_1}\mathbf{Y} \end{pmatrix}. \tag{11.7}$$

The *Khatri–Rao (matrix) product* of $\mathbf{X}$ and $\mathbf{Y}$ is defined as 'column-wise Kronecker'. That is, $\mathbf{X}$ and $\mathbf{Y}$ must have the same number of columns ($m_1 = m_2 = m$), and their Khatri–Rao product $\mathbf{X} \odot \mathbf{Y}$ is the $n_1 n_2$-by-m matrix defined as

$$\mathbf{X} \odot \mathbf{Y} = \begin{pmatrix} \mathbf{x}_1 \otimes \mathbf{y}_1, \mathbf{x}_2 \otimes \mathbf{y}_2, \ldots, \mathbf{x}_m \otimes \mathbf{y}_m \end{pmatrix} . \tag{11.8}$$

Notice that if the matrices $\mathbf{X}$ and $\mathbf{Y}$ are binary, both their Kronecker product, $\mathbf{X} \otimes \mathbf{Y}$, and their Khatri–Rao product, $\mathbf{X} \odot \mathbf{Y}$, are binary as well.

We can write the Boolean CP decomposition equivalently as matrices in three different representations using unfolding and matrix products:

$$\begin{aligned} \mathbf{X}_{(1)} &\approx \mathbf{A} \circ (\mathbf{C} \odot \mathbf{B})^T , \\ \mathbf{X}_{(2)} &\approx \mathbf{B} \circ (\mathbf{C} \odot \mathbf{A})^T , \\ \mathbf{X}_{(3)} &\approx \mathbf{C} \circ (\mathbf{B} \odot \mathbf{A})^T . \end{aligned} \tag{11.9}$$

The Boolean Tucker3 decomposition can be seen as a generalisation of the Boolean CP decomposition:

Problem 11.1.2 (Boolean Tucker3) *Given an n-by-m-by-l binary tensor* $\mathcal{X}$ *and three integers* r_1, r_2, *and* r_3, *find a binary* r_1*-by-*r_2*-by-*r_3 *core tensor* $\mathcal{G}$ *and binary factor matrices* $\mathbf{A}$ *(n-by-*r_1*),* $\mathbf{B}$ *(m-by-*r_2*), and* $\mathbf{C}$ *(l-by-*r_3*) such that they minimise*

$$\left| \mathcal{X} - \bigvee_{\alpha=1}^{r_1} \bigvee_{\beta=1}^{r_2} \bigvee_{\gamma=1}^{r_3} g_{\alpha\beta\gamma}\, \mathbf{a}_\alpha \boxtimes \mathbf{b}_\beta \boxtimes \mathbf{c}_\gamma \right|.$$

We use $[[\mathcal{G}; \mathbf{A}, \mathbf{B}, \mathbf{C}]]_B$ as the shorthand notation for the Boolean Tucker3 decomposition.

11.2 Problem Definitions

We consider the variation of *tensor clustering* where the idea is to cluster one mode of a tensor and potentially reduce the dimensionality of the other modes. Another common approach is to do the co-clustering equivalent, that is, to cluster each mode of the tensor simultaneously. The former is the approach taken, for example, by Huang et al. [112], while the latter appears for instance in Jegelka, Sra, and Banerjee [110]. Unlike either of these methods, however, we concentrate on binary data endowed with the Boolean algebra.

[112]: Huang et al. (2008), 'Simultaneous Tensor Subspace Selection and Clustering: The Equivalence of High Order SVD and *K*-Means Clustering'

[110]: Jegelka et al. (2009), 'Approximation Algorithms for Tensor Clustering'

For the presentation of our problems, we leave the exact measure of goodness of the approximation open. Only later, in Section 11.3, we will argue that the similarity – instead of the more common dissimilarity – should be used as the measure of goodness. After defining BTC for three-way tensors, we change the perspective and provide alternative definitions as clustering tasks. The end of this section is marked by a discussion of extensions to higher-order tensors in Section 11.2.4.

11.2.1 General Problem

Assuming a three-way tensor and that we do the clustering in the last mode, we can express the general *Boolean tensor clustering (BTC)* problem as follows:

Problem 11.2.1 (BTC) *Given a binary n-by-m-by-l tensor* $\mathcal{X}$ *and integers* r_1, r_2, *and* r_3, *find a* r_1*-by-*r_2*-by-*r_3 *binary tensor* $\mathcal{G}$ *and matrices* $\mathbf{A} \in \{0,1\}^{n \times r_1}$, $\mathbf{B} \in \{0,1\}^{m \times r_2}$, *and* $\mathbf{C} \in \{0,1\}^{l \times r_3}$ *such that* $\mathbf{C}$ *is a cluster assignment matrix and that* $[[\mathcal{G}; \mathbf{A}, \mathbf{B}, \mathbf{C}]]_B$ *is a good Boolean Tucker decomposition of* $\mathcal{X}$.

If we let $r_1 = n$ and $r_2 = m$, we obtain essentially a traditional (binary) clustering problem: given a set of l binary matrices $\{\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_l\}$ (the frontal slices of $\mathfrak{X}$), cluster these matrices into r_3 clusters, each represented by an n-by-m binary matrix $\mathbf{G}_i$ (the frontal slices of the core tensor $\mathfrak{G}$); the factor matrices $\mathbf{A}$ and $\mathbf{B}$ can be left as identity matrices. We call this problem *unconstrained BTC (U-BTC)*. Writing each of the involved matrices as an nm-dimensional vector, we obtain the following problem, called the *discrete basis partitioning problem (DBPP)* in [32]:

[32]: Miettinen et al. (2008), 'The Discrete Basis Problem'

Problem 11.2.2 (U-BTC [DBPP, 32]) *Given an l-by-nm binary matrix $\mathbf{X}$ and a positive integer r, find matrices $\mathbf{C} \in \{0,1\}^{l \times r}$ and $\mathbf{G} \in \{0,1\}^{r \times nm}$ such that $\mathbf{C}$ is a cluster assignment matrix and $\mathbf{C}$ and $\mathbf{G}$ minimise $|\mathbf{X} - \mathbf{CG}|$.*

Miettinen et al. [32] show that DBPP is NP-hard and give a $(10 + \varepsilon)$-approximation algorithm that runs in polynomial time with respect to n, m, l, and r, while Jiang [125] gives a 2-approximation algorithm that runs in time $O(nml^r)$.

[125]: Jiang (2014), 'Pattern extraction and clustering for high-dimensional discrete data'

11.2.2 Boolean CP Clustering

In what can be seen as the other extreme, we can restrict $r_1 = r_2 = r_3 = r$ and $\mathfrak{G}$ (which now is r-by-r-by-r) to having 1s exactly on the hyperdiagonal to obtain the *Boolean CP clustering (BCPC)* problem:

Problem 11.2.3 (BCPC) *Given a binary n-by-m-by-l tensor $\mathfrak{X}$ and an integer r, find matrices $\mathbf{A} \in \{0,1\}^{n \times r}$, $\mathbf{B} \in \{0,1\}^{m \times r}$, and $\mathbf{C} \in \{0,1\}^{l \times r}$ such that $\mathbf{C}$ is a cluster assignment matrix and that $[[\mathbf{A}, \mathbf{B}, \mathbf{C}]]_B$ is a good Boolean CP decomposition of $\mathfrak{X}$.*

What BCPC does is perhaps easiest to understand if we use the unfolding rules (Equation 11.9) and write

$$\mathbf{X}_{(3)} \approx \mathbf{C} \circ (\mathbf{B} \odot \mathbf{A})^T \,, \tag{11.10}$$

where we can see that compared to the general BTC, we have restricted the type of centroids: each centroid must be a row of type $(\mathbf{b} \otimes \mathbf{a})^T$. This restriction plays a crucial role in the decomposition, as we shall see shortly. Notice also that using Equation 11.6 we can rewrite Equation 11.10 without the Boolean product,

$$\mathbf{X}_{(3)} \approx \mathbf{C}(\mathbf{B} \odot \mathbf{A})^T \,. \tag{11.11}$$

11.2.3 Clustering Perspective

We defined U-BTC and BCPC as tensor decomposition problems with additional constraints to enforce the clustering. Equivalently, we can define the problems as clustering problems with additional constraints to enforce the tensor structure. We provide these alternative definitions here, as they will make the connections to other clustering problems easier to see.

Clustering version of U-BTC. The U-BTC problem is straight forward to express as a clustering task:

Problem 11.2.4 (U-BTC, clustering version) *Given a set X of binary vectors, $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_l\} \subset \{0,1\}^{nm}$ and an integer r, partition X in r disjoint sets $C_1, C_2, \ldots, C_r$, with each C_i associated with a centroid $\mathbf{g}_i \in \{0,1\}^{nm}$, such that the centroids $\mathbf{g}_i$ are good representations of vectors $\mathbf{x}_j$ that are in their clusters.*

The input vectors $\mathbf{x}_j \in X$ are simply the rows of matrix $\mathbf{X}_{(3)}$, and the assignment of $\mathbf{x}_j$s to clusters C_i defines the matrix $\mathbf{C} \in \{0,1\}^{l \times r}$: $c_{ij} = 1$ if and only if vector $\mathbf{x}_j$ is assigned to cluster C_i. Finally, treating vectors $\mathbf{g}_1, \ldots, \mathbf{g}_r$ as columns of an unfolded tensor $\mathbf{G}_{(3)} \in \{0,1\}^{nm \times r}$ gives us the m-by-n-by-r core tensor $\mathcal{G}$ that we used earlier to describe U-BTC.

This formulation of U-BTC makes its connection to standard partition clustering problems obvious; in particular, if the quality of the result is measured as the sum of Hamming distances from input vectors to their centroids, it is clear that U-BTC is equivalent to binary k-median clustering [133]. It is also clear from the formulation that U-BTC can easily suffer from the curse of dimensionality: a modestly-sized 100-by-100-by-100 tensor yields 100 vectors of dimensionality 10 000 to be clustered.

[133]: Miettinen (2009), 'Matrix Decomposition Methods for Data Mining: Computational Complexity and Algorithms'

Clustering version of BCPC. The BCPC problem imposes an additional constraint on the centroids: they must be of form $\mathbf{g}_i = (\mathbf{b}_i \otimes \mathbf{a}_i)^T$ for some $\mathbf{a} \in \{0,1\}^n$ and $\mathbf{b} \in \{0,1\}^m$.

Problem 11.2.5 (BCPC, clustering version) *Given a set X of binary vectors, $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_l\} \subset \{0,1\}^{nm}$ and an integer r, partition X in r disjoint sets $C_1, C_2, \ldots, C_r$, with each C_i associated with a centroid $\mathbf{g}_i = (\mathbf{b}_i \otimes \mathbf{a}_i)^T \in \{0,1\}^{nm}$, such that the centroids $\mathbf{g}_i$ are good representations of vectors $\mathbf{x}_j$ that are in their clusters.*

The vectors $\mathbf{a}_1, \ldots, \mathbf{a}_r$ yield the columns of the n-by-r factor matrix $\mathbf{A}$, and the vectors $\mathbf{b}_1, \ldots, \mathbf{b}_r$ yield the columns of the m-by-r factor matrix $\mathbf{B}$ used in the previous section to define BCPC.

While the additional constraints might not seem very intuitive in the clustering framework, they do increase the interpretability of the clusters. Also, they can help with the curse of dimensionality: compared to the unconstrained version, the degree of freedom of the centroids[1] is reduced to only those that admit a rank-1 decomposition. Conveniently, the centroids that admit the constraint can also be described in a more compact way than arbitrary centroids.

1: Degree of freedom refers to the number of possible centroids.

11.2.4 Extensions to Multi-Way Tensors

All the above definitions are restricted to three-way tensors. Three-way tensors are the most fundamental case, and they also allow relatively simple notation. Extending the definitions to three-way tensors, with $N > 3$, involves design decisions that are highly application-specific: for example, how many of the modes should be (co-)clustered and how high order should the centroids have. With four-way tensors, for instance, we could either co-cluster two modes so that the centroids would still be matrices, or we could cluster just one mode and have the centroids as three-way tensors.

If we cluster only the last mode of an N-way tensor $\mathfrak{X}$ and use $\mathbf{C}$ to denote the cluster assignment matrix and $\mathbf{A}^{(i)}$ to denote the factor for mode i with $i = 1, \ldots, N-1$, Equation 11.11 is generalised into

$$\mathbf{X}_{(N)} \approx \mathbf{C}(\mathbf{A}^{(N-1)} \odot \mathbf{A}^{(N-2)} \odot \cdots \odot \mathbf{A}^{(1)})^T \,. \tag{11.12}$$

If we keep the centroids as matrices and co-cluster the other modes, finding the centroids is similar to the three-way setting, but the clustering must be changed to some type of tensor co-clustering, such as [109].

[109]: Zhao et al. (2005), 'triCluster: An Effective Algorithm for Mining Coherent Clusters in 3D Microarray Data'

11.3 Similarity versus Dissimilarity

So far we have avoided defining what we mean by 'good' clustering. Arguably the most common approach is to say that any clustering is good if it minimises the sum of distances (or dissimilarities) between the original elements and their cluster representative (or centroid). An alternative is to maximise the similarity between the elements and their representatives. If the data and the centroids are binary, this problem is known as *hypercube segmentation* [127].[2]

[127]: Kleinberg et al. (2004), 'Segmentation problems'

2: Hypercube segmentation immediately solves U-BTC. Hence, it plays an important role in our algorithm for BCPC, as we shall see in Section 11.4

Maximising similarity is obviously a dual of minimising the dissimilarity – in the sense that an optimal solution to one is also an optimal solution to the other. The two, however, behave differently

when we aim at approximating the optimal solution. For example, for a fixed r, the best known approximation algorithm to DBPP is the aforementioned 2-approximation algorithm [125], while Alon and Sudakov [129] gave a PTAS for the hypercube segmentation problem.

[125]: Jiang (2014), 'Pattern extraction and clustering for high-dimensional discrete data'
[129]: Alon et al. (1999), 'On two segmentation problems'

The differences between the approximability, and in particular the reasons behind those differences, are important for data miners. It is not uncommon that our ability to minimise the dissimilarities is bounded by a family of problems where the optimal solution obtains extremely small dissimilarities, while the best polynomial-time algorithm has to do with slightly larger errors. These larger errors, however, might still be small enough relative to the data size that we can ignore them. This is what essentially happens when we maximise the similarities.

For example, consider an instance of DBPP where the optimal solution causes ten errors. The best-known algorithm can guarantee to not cause more than twenty errors; assuming the data size is 1000-by-500, making mistakes on twenty elements can still be considered excellent. On the other hand, should the optimal solution make an error on every fourth element, the best approximation could only guarantee to make errors in at most every second element. When considering the similarities instead of dissimilarities, these two setups look very different: in the first, we get within 0.99998 of the best solution, while in the other, we are only within $2/3$ of the optimal.[3] Conversely, if the algorithm wants to be within $9/10$ of the optimal similarity in the latter setup, it must approximate the dissimilarity within a factor of 1.3.[4] Hence similarity is less sensitive to small errors (relative to the data size) and more sensitive to large errors, than dissimilarity is. In many applications, this is exactly what we want.

None of this is to say that we should stop considering the error and concentrate only on the similarities. On the contrary: knowing the data size and the error, we can easily compute the similarity for the binary data, and especially when the error is relatively small, it is much easier to compare than the similarity scores. What we argue, however, is that when analysing the approximation ratio a data mining algorithm can obtain, similarity can give us a more interesting picture of the actual behaviour of the algorithm. Hence, subsequently, we shall concentrate on the maximum-similarity variants of the problems; most notably, we concentrate on the maximum-similarity BCPC, denoted $\text{BCPC}_{\max}$.

3: In the first case,

$$\frac{500\,000 - 20}{500\,000 - 1/2 \cdot 20} = 0.99998\,,$$

and in the second scenario

$$\frac{1/2 \cdot 500\,000}{500\,000 - 1/4 \cdot 500\,000} = \frac{2}{3}\,.$$

4: Derivation for the factor of 1.3:

$$\frac{(1-x) \cdot 500\,000}{500\,000 - 1/4 \cdot 500\,000} = \frac{9}{10}$$
$$\Leftrightarrow x = 0.325 = 1/4 \cdot 1.3\,.$$

11.4 Solving BCPC$_{\max}$

In this section, we present our main algorithm, `SaBoTeur`,[5] for solving the BCPC problem. Our algorithm works in two parts: it first finds a solution to U-BTC, and then turns that into a valid solution to BCPC. The maximum-similarity U-BTC is equivalent to hypercube segmentation, explained in Section 11.4.1. The overall approach of `SaBoTeur` is explained in Section 11.4.2, and the problem of turning a solution of U-BTC to a valid solution of BCPC is further studied in Section 11.4.3. We present an intuitive iterative updates scheme in Section 11.4.4, and discuss about the implementation details in Section 11.4.5. Finally, we consider the problem of selecting the correct number of clusters in Section 11.4.6.

5: `SaBoTeur` stands for Sampling for Boolean Tensor clustering.

11.4.1 The Hypercube Segmentation Problem

Given a tensor $\mathcal{X}$, for the optimal solution to BCPC$_{\max}$, we need matrices $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ that maximise $\mathsf{sim}(\mathbf{X}_{(3)}, \mathbf{C}(\mathbf{B} \odot \mathbf{A})^T)$. In the maximum-similarity U-BTC we replace $\mathbf{B} \odot \mathbf{A}$ with an arbitrary binary matrix $\mathbf{G}$. This latter problem is equal to the hypercube segmentation problem defined by Kleinberg, Papadimitriou, and Raghavan [127]:

[127]: Kleinberg et al. (2004), 'Segmentation problems'

Problem 11.4.1 (Hypercube Segmentation) *Given a set S of l vertices of the d-dimensional cube $\{0,1\}^d$, find r vertices $P_1, \ldots, P_r \in \{0,1\}^d$ and a partition of S into r segments such that*

$$\sum_{i=1}^{r} \sum_{c \in S} \mathsf{sim}(P_i, c)$$

is maximised.

The partition S defines the cluster assignment matrix $\mathbf{C}$, while the vertices P_i correspond to the rows of $\mathbf{G}$.[6]

6: This amounts to the clustering version of U-BTC, compare Problem 11.2.4.

Alon and Sudakov [129] gave a PTAS for the hypercube segmentation problem. The PTAS obtains a similarity within $(1 - \varepsilon)$ of the optimum with running time of $\mathrm{e}^{O\left((r/\varepsilon)^2 \ln r\right)} nml$ for n-by-m-by-l data and r clusters. While technically linear in data size, the first term turns the running time unfeasible even for moderate values of r and ε. Kleinberg, Papadimitriou, and Raghavan proposed a sampling-based algorithm that obtains an approximation ratio of $0.828 - \varepsilon$ with constant probability and running time $O(nmlrs)$,

[129]: Alon et al. (1999), 'On two segmentation problems'

where $s = (9/\varepsilon)^r \ln 1/\varepsilon$ is the number of times the sampling is done. While the number of samples this algorithm requires is still exponential in r, in practice we can take $s = \Theta(1)$ for a linear-time algorithm.[7] We will therefore follow a sampling approach similar to that of Kleinberg, Papadimitriou, and Raghavan in `SaBoTeur`.

7: The linear-time algorithm comes with the cost of losing approximation guarantees, though.

We note that both the PTAS [129] and the sampling-based algorithm [127] can be used as-is to solve the maximum-similarity U-BTC problem using the above connection between it and the hypercube segmentation problem.

11.4.2 The `SaBoTeur` Algorithm

We outline our algorithm for BCPC in Algorithm 3. `SaBoTeur` considers only the unfolded tensor $\mathbf{X}_{(3)}$. In each iteration, it samples r rows of $\mathbf{X}_{(3)}$ as the initial, unrestricted centroids. It then turns these unrestricted centroids into the restricted type in Line 3, and then assigns each row of $\mathbf{X}_{(3)}$ to its closest restricted centroid. The sampling is repeated multiple times, and in the end, the factors that gave the highest similarity are returned.

The algorithm is extremely simple and fast. The repeated sampling of rows of $\mathbf{X}_{(3)}$ for (initial) centroids is similar to the aforementioned hypercube segmentation algorithm, and we can use the analysis of Kleinberg, Papadimitriou, and Raghavan to justify why sampling the *rows* of the input matrix is justifiable (instead of finding arbitrary binary vectors). In particular, Kleinberg, Papadimitriou, and Raghavan proved that among the rows of $\mathbf{X}_{(3)}$ that are in the same optimal cluster, one is a good approximation of the (unrestricted) centroid of the cluster:

[127]: Kleinberg et al. (2004), 'Segmentation problems'

Lemma 11.4.1 ([127, Lemma 3.1]) *Let $\mathbf{X}$ be an n-by-m binary matrix and let*

$$\mathbf{y}^* = \arg\max_{\mathbf{y} \in \{0,1\}^m} \sum_{i=1}^{n} \mathsf{sim}(\mathbf{x}_i, \mathbf{y}) \ .$$

Then there exists a row $\mathbf{x}_j$ of $\mathbf{X}$ such that

$$\sum_{i=1}^{n} \mathsf{sim}(\mathbf{x}_i, \mathbf{x}_j) \geq (2\sqrt{2} - 2) \sum_{i=1}^{n} \mathsf{sim}(\mathbf{x}_i, \mathbf{y}^*) \ .$$

That is, if we sample one row from each cluster, we have a high probability of inducing a close-to-optimal clustering, and with enough re-samples, we have a high probability to have at least one such sample. The crux of BCPC however is the restriction of the centroids, and we will now turn our attention to the problem of enforcing this restriction in `SaBoTeur`.

Algorithm 3: `SaBoTeur` algorithm for the BCPC

Data: Three-way binary tensor $\mathcal{X}$, number of clusters r, number of samples s
Result: Binary factor matrices $\mathbf{A}$ and $\mathbf{B}$, cluster assignment matrix $\mathbf{C}$

1 **repeat**
2 Sample r rows of $\mathbf{X}_{(3)}$ into matrix $\mathbf{Y}$
3 Find binary matrices $\mathbf{A}$ and $\mathbf{B}$ that maximise $\mathsf{sim}(\mathbf{Y}, (\mathbf{B} \odot \mathbf{A})^T)$
4 Cluster $\mathbf{C}$: assign each row of $\mathbf{X}_{(3)}$ to closest row of $(\mathbf{B} \odot \mathbf{A})^T$
5 **until** s resamples are done;
6 **return** Best $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$

11.4.3 Binary Rank-1 Matrix Decompositions

After sampling, the next part of the `SaBoTeur` algorithm is to turn the unrestricted centroids into the restricted form (Line 3). We start by showing that this problem is equivalent to finding the maximum-similarity binary rank-1 decomposition of a binary matrix:

Problem 11.4.2 (Binary rank-1 decomposition) *Given an n-by-m binary matrix $\mathbf{X}$, find an n-dimensional binary vector $\mathbf{a}$ and an m-dimensional binary vector $\mathbf{b}$ that maximise* $\mathsf{sim}(\mathbf{X}, \mathbf{a} \boxtimes \mathbf{b})$.

Lemma 11.4.2 *Given an r-by-nm binary matrix $\mathbf{X}$, finding n-by-r and m-by-r binary matrices $\mathbf{A}$, $\mathbf{B}$ that maximise* $\mathsf{sim}(\mathbf{X}, (\mathbf{B} \odot \mathbf{A})^T)$ *is equivalent to finding the most similar binary rank-1 approximation of each row $\mathbf{x}$ of $\mathbf{X}$, where the rows are re-shaped as n-by-m binary matrices.*

Proof. If $\mathbf{x}_i$ is row i of $\mathbf{X}$ and $\mathbf{z}_i$ is the corresponding row of $(\mathbf{B} \odot \mathbf{A})^T$, then $\mathsf{sim}(\mathbf{X}, (\mathbf{B} \odot \mathbf{A})^T) = \sum_{i=1}^{k} \mathsf{sim}(\mathbf{x}_i, \mathbf{z}_i)$, and hence we can solve the problem row-by-row.

Let $\mathbf{x} = (x_{1,1}, x_{2,1}, \ldots, x_{n,1}, x_{1,2}, \ldots, x_{n,m})$ be a row of $\mathbf{X}$. Rewrite $\mathbf{x}$ as an n-by-m matrix $\mathbf{Y}$,

$$\mathbf{Y} = \begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,m} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,m} \end{pmatrix}.$$

Consider the row of $(\mathbf{B} \odot \mathbf{A})^T$ that corresponds to $\mathbf{x}$, and notice that it can be written as $(\mathbf{b} \otimes \mathbf{a})^T$, where $\mathbf{a}$ and $\mathbf{b}$ are the columns of $\mathbf{A}$ and $\mathbf{B}$ that correspond to $\mathbf{x}$.

As $(\mathbf{b} \otimes \mathbf{a})^T = (b_1\mathbf{a}^T, b_2\mathbf{a}^T, \cdots, b_m\mathbf{a}^T)$, rewriting it similarly as $\mathbf{x}$ we obtain

$$\begin{pmatrix} a_1b_1 & a_1b_2 & \cdots & a_1b_m \\ a_2b_1 & a_2b_2 & \cdots & a_2b_m \\ \vdots & \vdots & \ddots & \vdots \\ a_nb_1 & a_nb_2 & \cdots & a_nb_m \end{pmatrix} = \mathbf{a}\mathbf{b}^T = \mathbf{a} \boxtimes \mathbf{b} \ .$$

Therefore, $\mathsf{sim}(\mathbf{x}, (\mathbf{b} \otimes \mathbf{a})^T) = \mathsf{sim}(\mathbf{Y}, \mathbf{a} \boxtimes \mathbf{b})$. □

Now, we show that the maximum-similarity binary rank-1 decomposition admits a PTAS. To that end, we present a family of randomised algorithms that on expectation attain a similarity within $(1 - \varepsilon)$ of the optimum. The family is similar to that of Alon and Sudakov and can be analysed and de-randomised following the techniques presented by them [129]. The family of algorithms (one for each ε) is presented as Algorithm 4.

[129]: Alon et al. (1999), 'On two segmentation problems'

Lines 6 and 7 require us to solve one of the factor vectors when the other is given. To build $\mathbf{b}$ in Line 6, we simply take the column-wise majority element of those rows where $\mathbf{a}$ is 1, and we extend $\mathbf{a}$ similarly in Line 7. The analysis of Algorithm 4 follows closely the proofs by Alon and Sudakov.

The running time of the PTAS algorithm becomes prohibitive even with moderate values of ε, and therefore we will not use it in `SaBoTeur`.

Instead, we present a simple, deterministic algorithm that approximates the maximum similarity within 0.828, shown as Algorithm 5. It is similar to the algorithm for hypercube segmentation based on random sampling [127]. The algorithm considers every row of $\mathbf{X}$ as a potential vector $\mathbf{b}$ and finds the best $\mathbf{a}$ given $\mathbf{b}$.

[127]: Kleinberg et al. (2004), 'Segmentation problems'

Algorithm 4: Randomised PTAS for maximum-similarity binary rank-1 decompositions

Data: n-by-m binary matrix $\mathbf{X}$
Result: Binary vectors $\mathbf{a}$ and $\mathbf{b}$
1 Sample $l = \Theta(\varepsilon^{-2})$ rows of $\mathbf{X}$ u.a.r. with replacements
2 **forall** Partitions of the sample into two sets **do**
3 Let $\mathbf{X}'$ contain the rows in the sample
4 **for** Both sets in the partition **do**
5 Let $\mathbf{a}$ be the incidence vector of the set
6 Find $\mathbf{b}^T$ maximising $\mathsf{sim}(\mathbf{X}', \mathbf{a}\mathbf{b}^T)$
7 Extend $\mathbf{a}$ to rows not in sample maximising $\mathsf{sim}(\mathbf{X}, \mathbf{a}\mathbf{b}^T)$
8 **return** Best vectors $\mathbf{a}$ and $\mathbf{b}$

Algorithm 5: Approximation of maximum-similarity binary rank-1 decompositions

Data: n-by-m binary matrix $\mathbf{X}$
Result: Binary vectors $\mathbf{a}$ and $\mathbf{b}$
1 **forall** Rows $\mathbf{x}_i$ of $\mathbf{X}$ **do**
2 Let $\mathbf{b} = \mathbf{x}_i$
3 Find $\mathbf{a}$ maximising $\mathsf{sim}(\mathbf{X}, \mathbf{a}\mathbf{b}^T)$
4 **return** Best vectors $\mathbf{a}$ and $\mathbf{b}$

Using Lemma 11.4.1 it is straight forward to show that the algorithm achieves the claimed approximation ratio:

Lemma 11.4.3 *Algorithm 5 approximates the optimum similarity within* 0.828 *in time* $O(nm \min\{n, m\})$.

Proof. To prove the approximation ratio, let $\mathbf{a}^*(\mathbf{b}^*)^T$ be the optimum decomposition. Consider the rows in which $\mathbf{a}^*$ has 1. Per Lemma 11.4.1, selecting one of these rows, call it $\mathbf{b}$, gives us[8]

$$\mathsf{sim}(\mathbf{X}, \mathbf{a}^*\mathbf{b}^T) \geq (2\sqrt{2} - 2)\, \mathsf{sim}(\mathbf{X}, \mathbf{a}^*\mathbf{b}^T) \,.$$

Selecting $\mathbf{a}$ that maximises the similarity given $\mathbf{b}$ can only improve the result, and the claim follows as we try every row of $\mathbf{X}$.

If $n < m$, the time complexity follows as for every candidate $\mathbf{b}$ we have to make one sweep over the matrix. If $m < n$, we can operate on the transpose. □

8: Notice that $\mathbf{a}^*\mathbf{b}^T$ agrees with the optimal solution in rows where $\mathbf{a}^*$ is zero.

11.4.4 Iterative Updates

The `SaBoTeur` algorithm bears resemblance to the initialisation phase of k-means style clustering algorithms. We propose a variation of `SaBoTeur`, `SaBoTeur+itUp` shown in Algorithm 6, where k-means style iterative updates are performed on the result of `SaBoTeur` until it does not improve further. In each iteration of the iterative updates phase, the new centroids are first determined by placing 1s in those positions where the majority of cluster members have a 1. Next, these new centroids are constrained to rank-1, and finally, the cluster assignment is recomputed.

This procedure clearly slows down the algorithm. Also, due to the constraint on the centroids the iterative updates might actually impair the result, unlike in the normal k-means algorithm that converges to a local optimum. However, as we shall see in Chapter 12, the results in practice improve slightly.

Algorithm 6: `SaBoTeur+itUp` algorithm solves BCPC with k-means-like updates

Data: Three-way binary tensor $\mathfrak{X}$, number of clusters r, number of samples s
Result: Binary factor matrices $\mathbf{A}$ and $\mathbf{B}$, cluster assignment matrix $\mathbf{C}$

```
repeat
    Sample r rows of X_(3) into matrix Y
    repeat
        Find binary matrices A, B that maximise sim(Y, (B ⊙ A)^T)
        Cluster C: assign each row of X_(3) to closest row of (B ⊙ A)^T
        Update centroids Y with cluster members' majority vote
    until Similarity of [[A, B, C]]_B to X no longer improves;
until s resamples are done;
return Best A, B, and C
```

11.4.5 Implementation

We implemented the `SaBoTeur` algorithm in C.[9] In the implementation, the tensor $\mathfrak{X}$ is represented as a bitmap. This allows for using fast vectorised instructions such as `xor` and `popcnt`. In addition this representation takes exactly one bit per entry (excluding necessary padding), being very memory efficient even compared to sparse tensor representations.

9: The code is available from `https://cs.uef.fi/~pauli/btc/`

The second ingredient to the fast algorithm we present is parallelisation. As every row of $\mathbf{X}_{(3)}$ is handled independently, the algorithm is embarrassingly parallel. We use the OpenMP [134] and parallelise along the sampled cluster centroids. This means that the rank-1 decompositions of each centroid as well as the computation of the similarity in each of the columns of the Khatri–Rao product (Line 3 of Algorithm 3) are computed in parallel.

[134]: Dagum et al. (1998), 'OpenMP: an industry standard API for shared-memory programming'

11.4.6 Number of Clusters

A common problem in data analysis is that many methods require the selection of the *model order* a priori; for example, most low-rank matrix factorisation methods require the user to provide the target rank before the algorithm starts. Many clustering methods – ours included – assume the number of clusters as a parameter, although there are other methods that do not have this requirement.

When the user does not know what would be a proper number of clusters for the data, we propose the use of the *MDL principle* [135] for automatically inferring it. In particular, we adopt the recent work of Miettinen and Vreeken [136] on using MDL for BMF to our setting.

[135]: Rissanen (1978), 'Modeling by shortest data description'

The intuition behind the MDL principle is that the best model for the data[10] is the one that lets us compress the data most. We use the *crude MDL* where the *description length* contains two parts: the description length of the model M, $L(M)$, and that of the data D given the model M, $L(D \mid M)$. The overall description length is simply $L(M) + L(D \mid M)$. Intuitively, when the model complexity $L(M)$ increases,[11] the model fits better to the data and $L(D \mid M)$ decreases; however, at some point, the model starts over-fitting, and the increase in $L(M)$ overtakes the decrease in $L(D \mid M)$. In the sense of MDL, the optimal model is the one where $L(M)+L(D \mid M)$ is minimised.

10: In our case, that would be the most suitable number of clusters.

11: The model complexity increases for instance when more clusters are added.

In BCPC, the model consists of the two factor matrices $\mathbf{A}$ and $\mathbf{B}$ and the cluster assignment matrix $\mathbf{C}$. Given these, we can reconstruct the original tensor $\mathcal{X}$ if we know the locations where $[[\mathbf{A}, \mathbf{B}, \mathbf{C}]]_B$ differs from $\mathcal{X}$. Therefore, $L(M)$ is the total length of encoding $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$, while $L(D \mid M)$ is the length of encoding the locations of the differences, $\mathcal{X} \oplus [[\mathbf{A}, \mathbf{B}, \mathbf{C}]]_B$, where $\oplus$ is the element-wise *exclusive-or*.

Encoding $\mathbf{A}$ and $\mathbf{B}$ is similar to BMF, so we can use the data-to-model (DtM) encoding of Miettinen and Vreeken for encoding them (and the related dimensions of the data).[12] To encode the matrix $\mathbf{C}$, we only need to store to which cluster each frontal slice is associated to, taking $l \cdot \log_2(r)$ bits. This finalises the computation of $L(M)$.

12: The idea of DtM encoding is to encode all entries in one code [137]. Thus, in an abstract way, we can enumerate all possible strings that satisfy the given information.

[137]: Vereshchagin et al. (2004), 'Kolmogorov's structure functions and model selection'

To compute $L(D \mid M)$, we note that we can rewrite

$$\mathcal{X} \oplus [[\mathbf{A}, \mathbf{B}, \mathbf{C}]]_B = \mathbf{X}_{(3)} \oplus \mathbf{C} \circ (\mathbf{B} \circ \mathbf{A})^T \ . \tag{11.13}$$

The computation of $L(D \mid M)$ now becomes equivalent of computing it for BMF with factor matrices $\mathbf{C}$ and $\mathbf{D} = (\mathbf{B} \circ \mathbf{A})^T$. Hence, we can follow the T*yped XOR DtM* approach of Miettinen and Vreeken [136] directly.

To sum up, in order to use MDL to select the number of clusters, we have to run `SaBoTeur` with different numbers of clusters, compute the description length for each result, and take the one that yields the smallest description length.

12 Experimental Evaluation

Our experimental evaluation of SaBoTeur focuses on two central aspects: First, we test whether SaBoTeur can compete with other solvers for the Boolean and continuous CP decomposition in terms of reconstruction error and speed. Second, we examine the clusterings obtained by SaBoTeur in comparison to a standard k-means-type approach. In particular, we assess the ability of SaBoTeur to classify unseen data after learning the centroids.

12.1 Experimental Setup

To the best of our knowledge, this work presents the first algorithms for Boolean tensor clustering and hence we cannot compare directly to other methods. To assess the reconstruction error, we decided to compare SaBoTeur to continuous and Boolean tensor CP decompositions. We did not use other tensor clustering methods as they aim at optimising significantly different targets.[1] To evaluate the clustering quality, we compare SaBoTeur to a standard k-means-type approach.[2]

12.1.1 Compared Methods

We used the BCP_ALS [122] and Walk'n'Merge [108] algorithms for computing Boolean CP decompositions. BCP_ALS is based on iteratively updating the factor matrices one at a time, similarly to the classical alternating least squares optimisations, while Walk'n'Merge is an algorithm for scalable Boolean tensor factorisation in sparse binary tensors.

We did not use Walk'n'Merge on synthetic data as BCP_ALS is expected to perform better on smaller and denser tensors [108], but we used it on some larger real-world tensors; BCP_ALS, on the other hand, does not scale well to larger tensors and hence we had to omit it from most real-world experiments.

Of the continuous methods we compared to ParCube[3] and CP_APR.[4] The CP_APR algorithm [138] uses alternating Poisson regression and is specifically developed for sparse counting data, with the goal of returning sparse factors. It is debatable whether CP_APR is suitable for comparison in the first place, since it aims to minimise the generalised K–L divergence, not the squared error. And

1: We have highlighted this aspect in the discussion of related methods in Section 10.2.

2: We will explain the details subsequently in Section 12.2.5.

[122]: Miettinen (2011), 'Boolean Tensor Factorizations'

[108]: Erdős et al. (2013), 'Walk'n'Merge: A Scalable Algorithm for Boolean Tensor Factorization'

3: ParCube code is from http://www.cs.cmu.edu/~epapalex/.

4: The CP_APR implementation is from the Matlab Tensor Toolbox v2.5, found at http://www.sandia.gov/~tgkolda/TensorToolbox/.

[138]: Chi et al. (2012), 'On Tensors, Sparsity, and Nonnegative Factorizations'

furthermore, while counting data can be expected to follow the Poisson distribution, this is not true for binary data. Due to its aim for sparsity, we still consider it for comparison.

[118]: Papalexakis et al. (2012), 'ParCube: Sparse Parallelizable Tensor Decompositions'

The other method, `ParCube` [118], aims to minimise the squared error. It uses sampling to find smaller subtensors, solves the CP decomposition in this subtensor, and merges the solutions back into one. We used a non-negative variant of `ParCube` that expects non-negative data, and returns non-negative factor matrices.

12.1.2 Evaluation Criteria

For the synthetic data, we report the relative similarity. That is, the fraction of the elements where the data and the clustering agree. For the real-world data, we report the error in terms of squared Frobenius norm. That is the number of disagreements between the data and the clustering when both are binary (see Equation 11.1).

We have to take special care when comparing binary methods against continuous ones, though. Using the squared Frobenius norm can help the real-valued methods, as it scales all errors less than 1 down, but at the same time, small errors cumulate unlike with fully binary data. To alleviate this problem, we also rounded the reconstructed tensors from `CP_APR` and `ParCube` to binary tensors. We tried different rounding thresholds between 0 and 1 and selected the one that gave the lowest Boolean reconstruction error. The rounded versions are denoted by $\texttt{CP\_APR}_{0/1}$ and $\texttt{ParCube}_{0/1}$.

It is worth emphasising that all of the methods we are comparing against are solving a relaxed version of the BCPC problem: unlike BCPC, the Boolean CP factorisation is not restricted to clustering the third mode; the normal CP factorisation lifts even the requirement for binary factor matrices. Hence, a valid solution to BCPC is always also a valid solution to the (Boolean and normal) CP factorisation.[5] The methods we compare against should therefore always perform better than (or at least as good as) `SaBoTeur`.

5: The solutions for Boolean and normal CP factorisation are not interchangeable.

12.2 Synthetic Experiments

To test the `SaBoTeur` algorithm in a controlled environment, we created synthetic data sets that measured the response of the algorithm to

- different numbers of clusters;
- different density of data;
- different levels of additive noise;
- different levels of destructive noise.

All tensors were of size 700-by-500-by-50. All data sets were created by first creating random binary factor matrices $\mathbf{A}$, and $\mathbf{B}$, and a random cluster assignment matrix $\mathbf{C}$. The outer product of the factor matrices yields the ground-truth tensor $\mathcal{X} = [[\mathbf{A}, \mathbf{B}, \mathbf{C}]]_B$. The default number of clusters was $r = 7$ and the default density of the tensor was $d = 0.05$. To obtain the desired density for the tensor $\mathcal{X}$, we uniformly at random set a fraction of $\sqrt{d}$ elements of $\mathbf{A}$ and of $\mathbf{B}$ to 1. The outer product of these matrices and matrix $\mathbf{C}$ gives a tensor with the desired density.

Additive and destructive noise were applied to the tensor $\mathcal{X}$. Additive noise turns zeros into ones while destructive noise turns ones into zeros. The default noise level for both types was 10 %, yielding to a noised input tensor $\widetilde{\mathcal{X}}$. We report the noise levels with respect to the number of non-zeros.

We varied each of the four features one at a time keeping the others in their default values, and created five random copies on each parameter combination. We report the mean values over these five random copies. In all experiments, the number of clusters was set to the true number of clusters used to create the data. The number of re-samples in `SaBoTeur` was set to $r = 20$ in all experiments.

For these experiments, we compared the obtained reconstructions $[[\tilde{\mathbf{A}}, \tilde{\mathbf{B}}, \tilde{\mathbf{C}}]]_B$ to both the original tensor $\mathcal{X}$ and the noised tensor $\widetilde{\mathcal{X}}$, respectively using the relative similarity towards the original tensor,

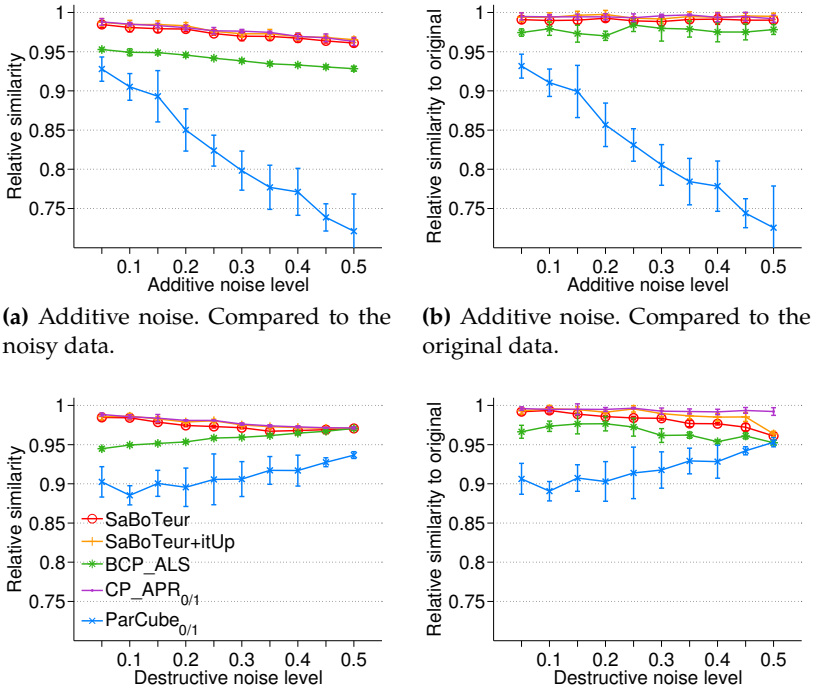
$$1 - \frac{|\mathcal{X} - [[\tilde{\mathbf{A}}, \tilde{\mathbf{B}}, \tilde{\mathbf{C}}]]_B|}{nml} ,$$

and the relative similarity towards the noised tensor,

$$1 - \frac{|\widetilde{\mathcal{X}} - [[\tilde{\mathbf{A}}, \tilde{\mathbf{B}}, \tilde{\mathbf{C}}]]_B|}{nml} .$$

12.2.1 Varying the Noise

In the first experiment, we studied the effects different noise levels have to the reconstruction accuracy. First, we varied the level of additive noise from 5 % to 50 % in steps of 5 %. The results are shown in Figure 12.1a, where we can see that `SaBoTeur` (with and without iterative updates) and $\texttt{CP\_APR}_{0/1}$ all achieve very high similarity with a gentle downwards slope as the level of additive noise increases. `BCP_ALS` is slightly worse, but consistent, while the performance of $\texttt{ParCube}_{0/1}$ suffers significantly from increased noise levels. In order to test if the good performance of `SaBoTeur` means it is just modelling the noise, we also compared the similarity of the reconstruction to the original, noise-free tensor $\mathcal{X}$ (Figure 12.1b). This did not change the results in meaningful ways, except that `BCP_ALS` improved by a bit.



(a) Additive noise. Compared to the noisy data.

(b) Additive noise. Compared to the original data.

(c) Destructive noise. Compared to the noisy data.

(d) Destructive noise. Compared to the original data.

Figure 12.1: Results from additive and destructive noise variation on synthetic data. All y-axes show the relative similarity. All markers are mean values over five iterations and the width of the error bars is twice the SD.

When moving from additive to destructive noise (Figure 12.1c), `SaBoTeur`, `SaBoTeur+itUp`, and `CP_APR`$_{0/1}$ stay the best three methods. As the destructive noise level increases, `BCP_ALS` approaches the three. That behaviour is mostly driven by 'modelling the noise', as can be seen in Figure 12.1d, which shows the similarity to the noise-free data. There, with highest levels of destructive noise, the performance of `SaBoTeur` suffers more than that of `CP_APR`$_{0/1}$.

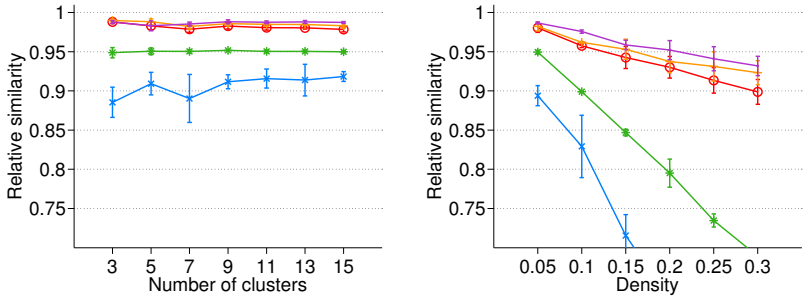
12.2.2 Varying the Number of Clusters

The number of clusters varied from 3 to 15 with steps of 2. The results are shown in Figure 12.2a. None of the tested methods show any significant effect to the number of clusters, and the best three methods are still `SaBoTeur`, `SaBoTeur+itUp`, and `CP_APR`$_{0/1}$.

12.2.3 Varying the Density

The density of the tensor varied from 5 % to 30 % with steps of 5 %. The results can be seen in Figure 12.2b. All methods perform worse with denser data, with `CP_APR`$_{0/1}$, `SaBoTeur+itUp`, and `SaBoTeur` being the best three methods in that order. The results of `BCP_ALS` and `ParCube` quickly went below 75 % similarity, `ParCube`$_{0/1}$ going as low as 55 % with 30 % density; we omit the worst results from the plots for clarity.[6]

6: Note that an increased density implies higher amounts of noise as the level of noise is defined with respect to the number of ones.



(a) Varying the number of clusters.

(b) Varying the density.

Figure 12.2: Results from varying the number of clusters and the density on synthetic data. Legend and data representation as in Figure 12.1. The relative similarity is with respect to the noisy data.

12.2.4 Scalability

We run the scalability tests on a dedicated machine with two Intel Xeon X5650 6-Core CPUs at 2.66 GHz and 64 GB of main memory. All reported times are wall-clock times. For these experiments, we created a new set of tensors with a default size of 800-by-800-by-500, a density of 5 %, 10 % of additive and destructive noise, and 20 clusters by default. During the experiments, we varied the size, the number of clusters, and the number of threads used for the computation.

We also tried `ParCube`, `CP_APR`, and `BCP_ALS` with these data sets, but only `ParCube` was able to handle that big data sets without running out of memory. However, already with 200-by-200-by-500 data, the smallest we tried, it took 155.4 seconds. With 10 clusters, the least number we used, `ParCube` finished only after 1 319.4 seconds. Therefore, we omitted these results from the figures.

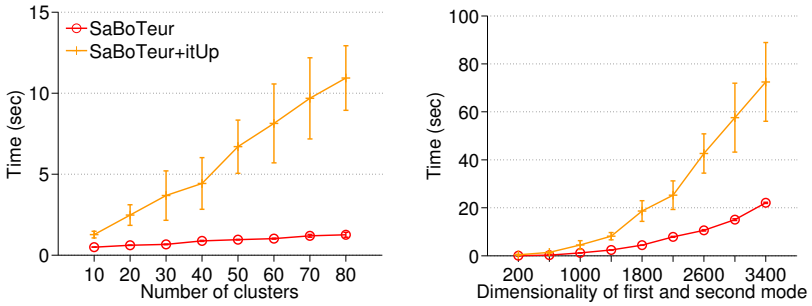
As can be seen in Figure 12.3a, the `SaBoTeur` algorithm scales very well with the number of clusters. This is mostly due to efficient parallelisation of the computing of the clusters.[7] `SaBoTeur+itUp`, however, slows down with higher number of clusters. This is to be expected, given that more clusters require more update computations.

For the second experiment, we varied the dimensionality of the first and second mode between 200 and 3 400 with steps of 400. The results can be seen in Figure 12.3b. As we grew both modes simultaneously, the size of the tensor grows as a square.[8] Given this, the running time of `SaBoTeur` grows as expected, or even slower, while `SaBoTeur+itUp` is again clearly slower.

In the last scalability experiment (Figure 12.3c), we tested how well `SaBoTeur` parallelises. As the computer we used had 12 cores, we set that as the maximum number of threads. Yet, after 9 threads, there were no more obvious benefits. We expect that the memory bus is the limiting factor here.

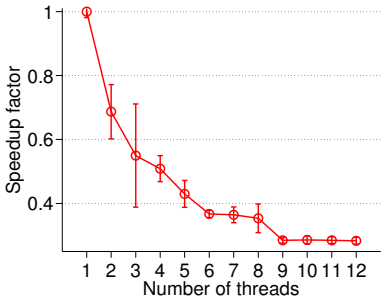
7: Anticipating the result from the last experiment on scalability.

8: The number of non-zeros likewise grows as a square, as the density was kept constant.



(a) Speed versus number of clusters.

(b) Speed versus size of the first and second mode.



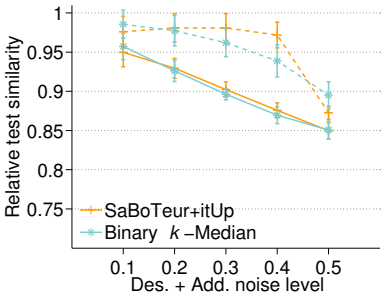
(c) Speedup factor for increasing number of threads.

Figure 12.3: Results from scalability tests. All times are wall-clock times. The speedup factor in (c) is provided with respect to the time a run with a single thread took. Markers are mean values over five iterations and the width of the error bars is twice the SD.

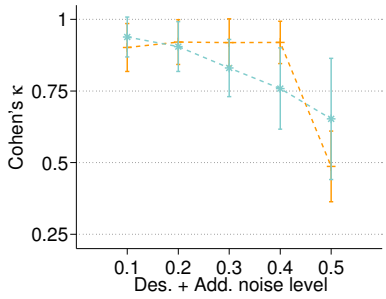
12.2.5 Generalisation Tests

Our final test with synthetic data is how well the clusterings of `SaBoTeur+itUp` generalise to yet-unseen data. Our hypothesis is that restricting the centroids to rank-1 matrices helps with the overfitting, and hence we compared `SaBoTeur+itUp` to an algorithm solving U-BTC, where the centroids are arbitrary binary matrices. As this problem is equivalent to clustering under binary constraints in the centroids (see Section 11.2.3), the algorithm we used follows the standard k-means-type approach: first it samples k random points as its centroids, and then it alternates between assigning the frontal slices to the closest centroid and recomputing the centroid as the element-wise majority matrix. We call this algorithm `Binary` k`-Median`. Notice that `Binary` k`-Median` will always obtain at least as good reconstruction error on the training data as `SaBoTeur+itUp`. Notice further that an in-between approach would be to run `Binary` k`-Median` with a subsequent rank-1 decomposition on the centroids. We did not include this variation in the set of experiments but instead compared `SaBoTeur` to the more extreme counterpart with no additional constraint on the centroids.

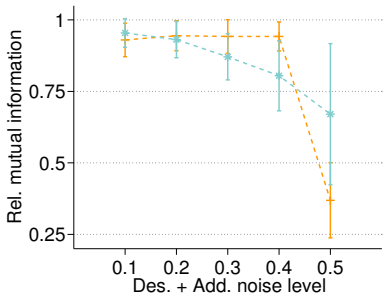
We generated tensors of size 700-by-500-by-300 with 7 clusters, 15 % of density and different levels of both additive and destructive noise, from 10 % till 50 %. We randomly selected 25 % of the frontal slices



(a) Similarity with respect to noised (solid) and original data (dashed lines).



(b) Cohen's κ.



(c) Normalised mutual information.

Figure 12.4: Evaluation of the generalisation quality using (a) reconstruction error on test data, and (b) Cohen's κ statistic and (c) normalised mutual information of cluster labels between the obtained and true labels in the test data. Markers are mean values over 50 different data sets and the width of the error bars is twice the standard deviation.

as the test data and computed the clusterings on the remaining data. We then connected each frontal slice in the test set to its closest centroid.

The quality of the results was measured in two different ways. First, we used the reconstruction error in the test data (Figure 12.4a) and computed the overall similarity of expressing the testing data using the centroids. We also used the original, noise-free frontal slices to assess whether we model the noise. The results show that with the noise-free test data (dashed lines), `SaBoTeur+itUp` is better than the `Binary` k`-Median` with 30 % to 40 % to noise, and the results are reversed with 10 % and 50 % of noise. With noisy test data, the differences are less.

We also measured the similarity of the cluster labels with respect to the ground truth. We used two metrics for that, Cohen's κ statistic (Figure 12.4b) and normalised mutual information (Figure 12.4c). Cohen's κ takes values from $[-1, 1]$, the former meaning complete disagreement and the latter meaning complete agreement. As Cohen's κ requires the labels of the clusters to match, we paired the labels using bipartite maximum matching.[9] The mutual information was normalised by dividing it with the joint entropy; the resulting statistic takes values from $[0, 1]$, with 1 meaning perfect agreement. To assess the significance of the results we used one-sided Wilcoxon rank sum tests for either side with a significance level of 5 %.

9: To compute the matching, we used the Hungarian method [139].

[139]: Papadimitriou et al. (1982), *Combinatorial Optimization: Algorithms and Complexity*

By significance tests on all results from Cohen's κ and on all results from the normalised mutual information, we find that `SaBoTeur+itUp` can recover clusterings overall significantly better than `Binary` k`-Median`. Looking at each level of noise separately, the two measures agree that `SaBoTeur+itUp` is significantly better at 30 % and 40 % of noise and worse with 10 % and 50 % of noise. With 20 % of noise, the difference is not significant with either metric.

Perhaps the clearest sign of the benefits of the rank-1 centroids that distinguish `SaBoTeur+itUp` from `Binary` k`-Median` is the consistently smaller standard deviation obtained by `SaBoTeur+itUp`. This suggests that `SaBoTeur+itUp` is less susceptible to random variations in the data, which should improve the generalisation quality. Indeed, our generalisation experiments with the real-world data in Section 12.3.5 support this.

12.3 Real Data Experiments

We test `SaBoTeur` with multiple real-world data sets of varying characteristics. The purpose of these experiments is to verify our findings with synthetic data. To that end, we study how well `SaBoTeur` (and the other methods) can reconstruct the data, how the methods scale with real-world data, and how `SaBoTeur` generalises to unknown data. In addition, we also study the sparsity of the factors, using MDL to select the number of clusters. And we assess how interpretable the results of `SaBoTeur` are.

12.3.1 Data Sets

We used nine data sets from different online resources. The size and density for each data set are listed in Table 12.1. Their contents, in row–column–tube order, comprise:

Table 12.1: Size and density of the real-world data sets.

	rows	columns	tubes	density (10^{-7})
Delicious	1 640	23 584	7 968	8.23
Enron	146	146	38	22 752.86
Facebook	42 390	39 986	224	16.51
Last.FM	1 892	12 523	9 749	8.07
MovieLens	2 113	5 908	9 079	4.23
MAG	10 197	10 673	20	1 036.37
Resolver	343	360	200	626.01
TracePort	10 266	8 622	501	2.51
YAGO	190 962	62 428	25	67.35

Delicious — *user–bookmark–tag* tuples from the `del.icio.us` social bookmarking system [140];

Enron — *sender–recipient–month* tuples from email communication of Enron employees [141];

Facebook — *user–user–week* tuples telling who posted a message on whose wall on `facebook.com` per week [142];

Last.FM — *user–artist–tag* tuples from the `Last.fm` online music system [140];

MovieLens — *user–film–tag* tuples from the `movielens.org` film recommendation site [140];

MAG — *film–actor–genre* tuples also from `movielens.org`;[10]

Resolver — *entity–entity–relation* tuples from the TextRunner open information extraction algorithm [143];[11]

TracePort — *source–target–port* tuples encoding anonymised passive IP traffic traces from 2009 [144];

YAGO — *subject–object–relation* tuples of the semantic knowledge base `yago-knowledge.org` [145].

[140]: Cantador et al. (2011), 'Second Workshop on Information Heterogeneity and Fusion in Recommender Systems'

[141]: Klimt et al. (2004), 'The Enron Corpus: A New Dataset for Email Classification Research'

[142]: Viswanath et al. (2009), 'On the Evolution of User Interaction in Facebook'

10: The MAG data has a special structure, as the actors are connected to the genres only through the films.

[143]: Yates et al. (2009), 'Unsupervised methods for determining object and relation synonyms on the web'

11: We used the sample called ResolverL from Miettinen [122].

[122]: Miettinen (2011), 'Boolean Tensor Factorizations'

[144]: Center for Applied Internet Data Analysis (2009), *The CAIDA UCSD Anonymized Internet Traces*

[145]: Suchanek et al. (2007), 'Yago: A Core of Semantic Knowledge'

Throughout the experiments, the clustering constraint is imposed on the last-mentioned dimension.

As a preprocessing step we removed all the all-zero slices in every mode. In addition, for the Delicious data set, also all slices that had less than six entries were purged. For the MAG data, all actors that appeared in less than five films were discarded.

12.3.2 Reconstruction Accuracy

As all employed data sets are very sparse, we report the errors, not the similarity, for these experiments. The error is measured as the number of disagreements, when the reconstructed tensor is also binary, or as squared tensor Frobenius distance, when the reconstructed tensor is real-valued.[12] The results can be seen in Table 12.2. The results for $\texttt{CP\_APR}_{0/1}$ and $\texttt{ParCube}_{0/1}$ for all data sets except Enron and Resolver are obtained by sampling: If $|\mathcal{X}|$ is the number of non-zeros in data $\mathcal{X}$, we sampled $200 \cdot |\mathcal{X}|$ locations of zeros, and computed the error the methods make for every non-zero and for the sampled zero elements. The sampled results were then extrapolated to the size of the full tensor. We had to use sampling, as the factor matrices were too dense to allow reconstructing the full tensor.[13]

12: We use the not-rounded versions of CP_APR and ParCube, unlike above.

13: We will discuss more on density in Section 12.3.3.

The smallest reconstruction error is obtained by $\texttt{ParCube}_{0/1}$ in almost all experiments, YAGO being a notable exception. Remember, however, that $\texttt{ParCube}_{0/1}$ returns a non-negative tensor CP decomposition that is afterwards rounded to binary tensor; $\texttt{ParCube}_{0/1}$ is

Table 12.2: Reconstruction errors on real data sets, measured as squared tensor Frobenius distance. '—' means that the algorithm was not able to finish.

	Delicious				Enron				MAG	
$r =$	7	10	15	30	5	10	12	15	5	7
SaBoTeur	253 608	253 546	253 447	253 004	1 811	1 779	1 769	1 753	224 322	223 580
SaBoTeur+itUp	253 153	253 293	252 738	252 859	1 793	1 756	1 750	1 735	223 869	223 489
BCP_ALS	—	—	—	—	1 850	1 850	1 850	1 850	—	—
CP_APR	253 558	253 516	253 433	253 196	1 718	1 631	1 598	1 560	224 996	224 696
CP_APR$_{0/1}$	253 653	253 653	253 652	253 639	1 838	1 817	1 811	1 781	225 580	225 155
ParCube	367 521	373 710	363 581	418 252	2 137	2 273	2 352	2 270	375 290	322 121
ParCube$_{0/1}$	247 129	246 566	246 016	241 850	1 802	1 744	1 751	1 690	222 133	221 933
Walk'n'Merge	—	251 034	—	—	—	—	1 753	—	—	—

	MovieLens				Resolver				Facebook	
$r =$	5	15	20	30	5	10	15	30	15	20
SaBoTeur	56 761	56 712	56 678	56 571	1 530	1 509	1 485	1 455	626 999	626 776
SaBoTeur+itUp	47 933	47 479	47 280	47 316	1 522	1 503	1 457	1 443	—	—
BCP_ALS	—	—	—	—	1 624	1 624	1 626	1 632	—	—
CP_APR	47 584	46 927	46 555	46 287	1 519	1 509	1 489	1 457	626 343	626 202
CP_APR$_{0/1}$	47 178	44 655	43 614	42 609	1 545	1 545	1 538	1 539	626 945	626 945
ParCube	252 719	463 025	454 826	449 534	1 784	1 762	1 798	1 939	750 891	797 534
ParCube$_{0/1}$	41 223	38 098	37 915	37 393	1 507	1 471	1 450	1 370	619 939	620 894
Walk'n'Merge	—	46 807	—	—	—	—	1 534	—	—	—

	Last.FM				TracePort				YAGO
$r =$	8	15	20	30	5	10	15	30	7
SaBoTeur	195 905	195 735	195570	195 278	11 085	11 004	10 923	10 776	1 957 737
SaBoTeur+itUp	186 072	185 713	185651	185 399	10 990	10 961	10 913	10 673	—
BCP_ALS	—	—	—	—	—	—	—	—	—
CP_APR	184 513	184 038	183 489	182 659	11 140	11 114	11 082	11 027	—
CP_APR$_{0/1}$	180 766	177 527	174 491	170 939	11 110	11 059	10 893	10 617	—
ParCube	2 745 259	2 879 381	2 856 941	2 770 361	19 803	30 793	31 107	29 796	619 476 763
ParCube$_{0/1}$	133 057	139 207	119 461	131 322	10 708	9 913	10 097	9 647	4 793 390
Walk'n'Merge	—	—	—	—	—	—	10 679	—	—

Table 12.3: Total density of the factor matrices **A** and **B** for different r on the Delicious data.

$r =$	7	10	15	30
SaBoTeur	0.000 16	0.000 31	0.000 29	0.000 29
CP_APR	0.202 19	0.147 86	0.103 12	0.053 96
ParCube	0.281 86	0.210 69	0.166 58	0.099 00
Walk'n'Merge	—	0.000 22	—	—

neither clustering nor Boolean, and hence together with CP_APR$_{0/1}$ is expected to be better than SaBoTeur. Also, without the rounding, ParCube is often the worst method by a large margin. CP_APR benefits much less from the rounding, CP_APR$_{0/1}$ being comparable to SaBoTeur.

In those data sets where we were able to run BCP_ALS, its results were consistently worse than SaBoTeur's results, despite it solving more relaxed problem. We were unable to get Walk'n'Merge to finish within a reasonable time with most data sets: it found rank-1 tensors sufficiently quickly, but took too much time to select the top ones from there.[14] When it did finish, however, it was slightly better than SaBoTeur+itUp or SaBoTeur.

14: See Erdős and Miettinen [108] for explanations on how Walk'n'Merge finds a Boolean CP decomposition.

[108]: Erdős et al. (2013), 'Walk'n'Merge: A Scalable Algorithm for Boolean Tensor Factorization'

Finally, the iterative updates of SaBoTeur+itUp consistently improved the results compared to SaBoTeur, but did so with significant increase to the running time. It seems that with most data sets, the iterative updates are not necessarily worth the extra wait.

12.3.3 Sparsity of the Factors

An important question on the practical feasibility of the tensor decomposition and clustering algorithms is the density of the factor matrices. Too dense factor matrices increase the computational complexity and also the storage requirements. Furthermore, multiplying the dense factors together becomes prohibitively expensive, making it impossible to fully reconstruct the tensor. This is the reason why, for example, we had to use sampling to compute the rounded representations of CP_APR and ParCube.

We studied the sparsity of the factors using the Delicious data. In Table 12.3 we report the total densities of the matrices $\mathbf{A}$ and $\mathbf{B}$, that is,

$$\frac{|\mathbf{A}| + |\mathbf{B}|}{rn + rm}$$

for n-by-r and m-by-r factors.

It is obvious that the Boolean methods, SaBoTeur and Walk'n'Merge, are orders of magnitude sparser than the continuous methods, CP_APR and ParCube. This was to be expected as similar behaviour has already been observed with Boolean matrix factorisations [146].

[146]: Miettinen (2010), 'Sparse Boolean Matrix Factorizations'

We did not compare the third-mode factor matrices for densities. For SaBoTeur, the clustering assignment matrix $\mathbf{C}$ has density $1/l$ that depends only on the number of frontal slices l; obtaining density less than that requires having rows of $\mathbf{C}$ that have no non-zeros.

Table 12.4: Average wall-clock running times in seconds for real-world data sets. Averages are over five restarts for MovieLens and ten for Resolver. Both data sets used $r = 15$ clusters.

	MovieLens	Resolver
SaBoTeur	45.6	0.02
SaBoTeur+itUp	303.6	0.11
CP_APR	196.6	22.00
ParCube	24.8	7.68
Walk'n'Merge	207.5	2.15

12.3.4 Scalability

Table 12.4 shows the wall-clock times for MovieLens and Resolver with $r = 15$. MovieLens is one of the sparsest data sets, while Resolver is one of the densest ones, and hence these two show us how the relative speed of the methods changes as the density changes. The other data sets generally followed the behaviour we report here, adjusting to their density.

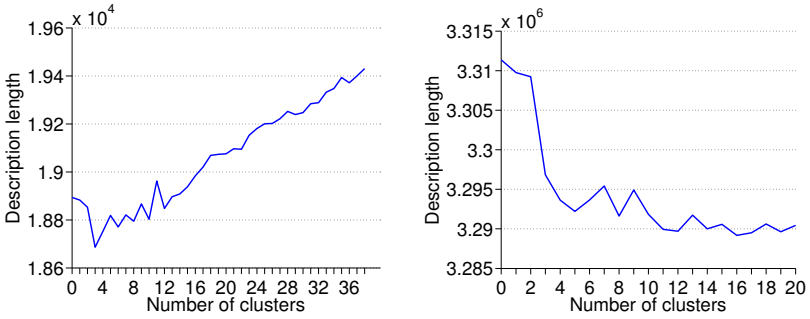
With the MovieLens data, ParCube is the fastest of the methods, followed by SaBoTeur. CP_APR, Walk'n'Merge, and SaBoTeur+itUp are significantly slower than the two. With the denser Resolver data, however, the order is changed, with SaBoTeur and SaBoTeur+itUp being an order of magnitude faster than Walk'n'Merge, which still is faster than ParCube or CP_APR. The density of the data does not affect the speed of SaBoTeur, while it does have a significant effect to CP_APR and ParCube. SaBoTeur+itUp is less affected by density, and more affected by the size of the data, than the other methods.

12.3.5 Generalisation Capability

We repeated the generalisation test we did with synthetic data with three real-world data sets – MovieLens, Resolver, and TracePort – keeping 85 % of the frontal slices as training data while the remaining 15 % we used for testing. The results can be seen in Table 12.5. We used $r = 15$ clusters for MovieLens and TracePort, and ten clusters for the Resolver data. On each data set we conducted ten repetitions of the training phase, and selected the one with the smallest training error.

Table 12.5: Training and testing errors on real-world data. The number of clusters was $r = 15$ for MovieLens and TracePort and $r = 10$ for Resolver.

		MovieLens	Resolver	TracePort
train	SaBoTeur+itUp	40 075	1 282	9 730
	Binary k-Median	39 351	1 122	8 843
test	SaBoTeur+itUp	7 258	225	1 144
	Binary k-Median	8 256	274	1 554



(a) Enron **(b)** MAG

Figure 12.5: Description length vs. number of clusters.

As expected, `Binary` k`-Median` achieves lower training error, but larger testing error. This agrees with our synthetic experiments, and strongly indicates that using the rank-1 centroids helps to avoid overfitting.

12.3.6 Selecting the Number of Clusters

We tested the use of MDL to select the number of clusters with Enron and MAG. To that end, we ran `SaBoTeur` with every possible number of clusters: from having all slices in one cluster to having one cluster for each slice. Further, we computed the description length of representing the data with no clusters.[15] The description lengths for the different numbers of clusters are presented in Figure 12.5. For Enron, the number of clusters that gave the smallest description length was 3, while for MAG it was 16.

15: This amounts to having the full data explained in the $L(D \mid M)$ part.

Both Enron and MAG show non-smooth behaviour in the description length. This is likely partially due to the randomised behaviour of `SaBoTeur` and partially due to the fact that `SaBoTeur` does not consider the description length when it builds the clustering. Nevertheless, especially with Enron (Figure 12.5a), the selection of the best number of clusters – in the sense of MDL – seems clear.

12.3.7 Interpretability

For the final experiment, we studied some of the results we obtained from `SaBoTeur` in order to see if they can provide insights to the data. We report results from four data sets: Enron, MAG, Delicious, and Last.FM. For Enron and MAG, we can interpret all three modes (email sender and recipient per month in Enron and films, actors, genres in MAG); for the other two, we can only interpret two of the three modes (tags and URLs, and tags and artists, respectively), as the third mode corresponds to user IDs.

For Enron, the densest and smallest of the examined data sets, we mined five clusters. This data set is created from a set of email messages that was made public during the legal operation concerning the Enron corporation. The clusters `SaBoTeur` obtained can be associated to certain events in the timeline of this scandal or to the roles of certain people. One cluster we find is characterised by communication among the directors of Enron: in many months between September 2000 and December 2001, John Arnold (vice president), sends messages to Michael Maggi (director), to Andy Zipper (vice president), to Dutch Quigley (unknown position), and to John Griffith (unknown position). Another cluster has only November 2001 as member and describes communication of John Lavorato (CEO) to Kevin Presto (vice president), John Arnold (vice president), Richard Shapiro (vice president), Barry Tycholiz (vice president), Rick Buy (chief risk management officer), Stanley Horton (president), Bradley Mckay (employee), Kenneth Lay (CEO), Greg Whalley (president), and Mike Swerzbin (trader). This month is when the formal investigation started and Enron negotiated to be bought by Dynegy to avoid bankruptcy. Dynegy first agrees on a deal and withdraws its offer in the end of November 2001.

For MAG, we used 16 clusters, as suggested by MDL, and clustered the genres. As there are only 20 genres in the data, this means many clusters contained only one genre; this, however, is not surprising, as we would expect the sets of films that fall in different genres be generally rather different. The results picked famous films of the genre as the centroids: for example, the cluster of *animation* and *comedy* genres had the 1995 animation *Toy Story* appearing in its centroid. Another cluster, containing genres *adventure* and *fantasy*, had as its centroid Peter Jackson's *Lord of the Rings* trilogy, and its main cast. Similarly to *Toy Story*, *Lord of the Rings* films are arguably stereotypical adventure–fantasy films.

The centroids for the MAG data contained very few films (for example the three *Lord of the Rings* films). This is due to the fact that the data sets are very sparse, and the algorithm can only pick up the most prominent elements in the centroids.

With even sparser data sets, such as Delicious and Last.FM, the centroids get even sparser, perhaps containing just a singleton element from some mode. This is a common problem to many Boolean methods [see, *e.g.* 32]. To alleviate it, we used the weighted the reconstruction error, a method proposed by Miettinen et al. [32]. Normally, representing a 0 as a 1 and representing a 1 as a 0 both increase the error by 1, but for the next experiments, we set the weight so that representing a 1 with a 0 causes ten times more error as representing 0 as a 1.

[32]: Miettinen et al. (2008), 'The Discrete Basis Problem'

For Delicious and Last.FM, we mined 30 clusters. Even with the 10-fold weight, results for Delicious were rather sparse.[16] Some clusters were very informative, however, such as one with the tags *software*, *apple*, *mac*, and *osx* that contained bookmarked pages on *HFS for Windows*, *iVPN*, *Object-Oriented Programming with Objective-C*, and *iBooks and ePub*, among others. For Last.FM, the clusters were generally more dense, as users tend to tag artists more homogeneously. For example, the cluster containing the tags *pop* and *rnb* contained the superstar artists such as *Beyoncé*, *Britney Spears*, *Katy Perry*, and *Miley Cyrus* (and many more). But importantly, one could also find less-known artists: in a cluster containing tags *dance*, *trance*, *house*, *progressive trance*, *techno*, and *vocal trance*, we found artists such as *Infected Mushroom*, *T.M.Revolution*, *Dance Nation*, and *RuPaul*.

16: This sparsity indicates that the behaviour of the users is very heterogeneous.

12.4 Discussion

All the experiments show that `SaBoTeur` can hold its own even against methods that are theoretically superior: For reconstruction error (or similarity) with both synthetic (Sections 12.2.1 to 12.2.3) and real-world data (Section 12.3.2), `SaBoTeur` (and `SaBoTeur+itUp`) achieve results that are the best or close to the best results, notwithstanding that other methods, especially $\texttt{CP\_APR}_{0/1}$ and $\texttt{ParCube}_{0/1}$, benefit from significantly less constrained forms of decomposition, coupled with the post-hoc rounding. These relaxations also come with a price, as both methods create significantly denser factor matrices, and – as can be seen in Sections 12.2.4 and 12.3.4 – are unable to scale to denser data. Compared to the Boolean CP factorisation methods, `BCP_ALS` and `Walk'n'Merge`, `SaBoTeur` is comparable (or sometimes clearly better) in reconstruction error and density of the factor matrices, while being significantly more scalable.

Our hypothesis that the rank-1 centroids help with overfitting was also confirmed (Sections 12.2.5 and 12.3.5): while `Binary` k`-Median` obtained lower training errors (as expected), `SaBoTeur` obtained better results with testing data in all real-world data sets and in most synthetic experiments (with TracePort, `SaBoTeur`'s testing error was more than 25 per cent better than that of `Binary` k`-Median`).

Our experiments also show that MDL can be used to select the number of clusters automatically, even though we do think that more studies are needed to better understand the behaviour of the description length. Also, when studying the results, it seems obvious that `SaBoTeur` can return results that are easy to interpret and insightful; in part, this is due to the Boolean algebra, and in part, due to the clustering.

The last open question is whether one should use `SaBoTeur` or `SaBoTeur+itUp`. Judging by the results, `SaBoTeur+itUp` can provide tangible benefits over `SaBoTeur`, however, with a significant price in the running time. Thence, we think that it is best to start with `SaBoTeur`, and only if the user thinks she would benefit from more accurate results, move to `SaBoTeur+itUp`.

Conclusion 13

To conclude the second part of the thesis, in this chapter, we summarise our contributions for Boolean tensor clustering and discuss potential future directions.

13.1 Summary

We have studied the problem of clustering one mode of a three-way binary tensor while simultaneously reducing the dimensionality in the two other modes. This problem bears close resemblance to the Boolean CP tensor decomposition, but the additional clustering constraint makes the problem significantly different. The main source of computational complexity – the consideration of overlapping factors in the tensor decomposition – does not play a role in Boolean CP clustering. From another point of view, the problem can be seen as a clustering task where the centroids are constrained to rank-1 matrices in order to comply with the tensor structure.

For our assessment of the theoretical quality of the algorithm, we analyse the difference of maximising the similarity and minimising the dissimilarity. While typically, the quality of a clustering or tensor decomposition is measured using the distance of the original data to its representation, find that using the similarity instead of distance leads to a more meaningful analysis – also for other data mining problems, especially with binary data.

As a by-product of the analysis of our algorithm in terms of maximising the similarity instead of minimising the dissimilarity, we show that the problem of finding maximum-similarity binary rank-1 decompositions admits a PTAS and present a scalable algorithm that achieves a 0.828 approximation ratio.

Our experiments show that the `SaBoTeur` algorithm obtains similar reconstruction errors compared to the much less restricted tensor factorisation methods while obtaining sparse factor matrices and overall best scalability. We also showed that restricting the cluster centroids to rank-1 matrices significantly helped the generalisation to unseen data.

13.2 Challenges

While studying the task of Boolean tensor clustering, we encountered three different subjects which could be interesting for future considerations.

First, throughout our work, we concentrated on three-way tensors as they are the most common type of tensor data. The techniques we presented can be extended to higher-order tensors. Though, when realising such an extension, one has to first decide whether to cluster or decompose the further modes. It is also an option to cluster some modes and decompose some others. Just like the selection of the mode along which to cluster a three-way tensor, such a decision will of course highly depend on the data. Therefore it could be a helpful addition to develop strategies to assist the decision of decomposing or clustering for every mode. Furthermore, of course the resulting algorithms for higher-order tensors will depend on these decisions. Development and in-depth study of those algorithms could yield interesting future research.

Second, the idea of restricting cluster centroids to a specific structure opens opportunities for future research: Other clustering approaches, also on other types of data, could be amended by restrictions of cluster centroids. While the exact benefits of such a modification need to be explored, we envision some potential of such constraints to help with the curse of dimensionality. Another idea along these lines is to consider higher-rank centroids for our algorithm. That would mean to explore the behaviour of the algorithm when moving from BCPC towards unconstrained BTC. Then, the rank of the centroid would act as a regularisation parameter. To explore whether such a regularisation is viable, both for the Boolean and the continuous domain, could be an intriguing topic of further research.

Third, we argue that the similarity – as opposed to distance or error – is often a more meaningful metric for studying the (theoretical) performance of data mining algorithms due to its robustness towards small errors and more pronounced effects of large errors. While we have seen the benefits of assessing similarity instead of error in our analysis, such a shift of view might be an asset for more methods, both existing and novel. Our hypothesis is that good theoretical performance in terms of similarity correlates better with good real-world performance than good (or bad) performance correlates with the error.

14 Interconnecting the Pieces

To round off, in this chapter, we outline how hyperbolic communities and Boolean tensor clustering could benefit from each other.

14.1 Hyperbolic Cluster Centroids

We have presented two major lines of work: hyperbolic communities and Boolean tensor clustering. Both lines, we have introduced as beneficial for analysing and understanding graph data in particular. And in both lines, we rely on the representation in terms of binary data. Yet, the goals of both works, as well as their methodology, differ to a great extent: in one case we seek for a fast, yet performant, tensor decomposition; in the other case our focus is the realistic modelling of network structures.

What benefits could their combination have? In our implementation of Boolean tensor clustering, we suggest rank-1 matrices as cluster representatives. While our experiments indicate that this is a reasonable idea, we may ask whether, for certain data, not just a higher rank but another form of representative would be a better fit given the data. For instance, if we would like to analyse the time course of a social network, its modules would – as we have argued extensively – be well explained by means of hyperbolic communities. To replace the rank-1 centroids that `SaBoTeur` discovers with hyperbolic communities, could help the interpretability of the results. In our Boolean tensor clustering framework we discover the rank-1 centroids through decomposition of the tensor. Doing the same, but discovering hyperbolic communities as the factors in the decomposition brings us back exactly to the problem of detecting the hyperbolic communities (compare Section 9.2): an approach towards solving BCPC such that the hyperbolic communities serve as cluster centroids would likewise be an approach towards hyperbolic community detection.

As touched upon, hyperbolic communities are a special case of nested subgraphs [40]. In the following, we outline how this observation offers a community detection approach by means of a factorisation.

[40]: Karaev et al. (2018), 'Logistic-Tropical Decompositions and Nested Subgraphs'

Nested subgraphs can be defined based on the neighbourhood N of every node:

Definition 14.1.1 *An undirected graph $G = (V, E)$ is* nested *if we can order the vertices $v \in V$ in a sequence $(v_1, v_2, \ldots, v_n)$ such that $N(v_{i+1}) \subseteq N(v_i)$ for all $i = 1, \ldots, n-1$.*

Karaev, Metzler, and Miettinen propose an algorithm for covering by nested subgraphs. To achieve a formulation of the problem as a matrix factorisation problem, they use an alternate characterisation of nested subgraphs by means of the rounding rank; a binary matrix $\mathbf{A} \in \{0,1\}^{n \times m}$ has *nonnegative rounding rank* of 1 if and only if there exist nonnegative vectors $\mathbf{x} \in \mathbb{R}^n_{\geq 0}$ and $\mathbf{y} \in \mathbb{R}^m_{\geq 0}$ such that

$$A = \tau_\alpha(\mathbf{x}\mathbf{y}^T) \,, \tag{14.1}$$

where $\tau_\alpha \colon \mathbb{R}_{\geq 0} \to \{0,1\}$ is a threshold function with $\tau_\alpha(x) = 1$ if $x \geq \alpha$ and 0 otherwise.[1] As the following proposition shows, the nonnegative rounding rank is connected to nestedness:

1: Notice that we apply the threshold function τ_α independently to each element of the matrix $\mathbf{x}\mathbf{y}^T$.

[41]: Neumann et al. (2016), 'What You Will Gain By Rounding: Theory and Algorithms for Rounding Rank'

Proposition 14.1.1 ([41, Theorem 11]) *Let $\mathbf{A}$ be a binary matrix. $\mathbf{A}$ is nested if and only if it has nonnegative rounding rank of 1.*

The extension of this characterisation to a union of nested subgraphs is unfortunately not immediate: a rank-k decomposition is unequal to the union of k nested submatrices. As a remedy, Karaev, Metzler, and Miettinen replace the standard algebra with the subtropical algebra and show that the characterisation of nested matrices under rounding rank extends to higher-rank tropical decompositions. They propose a stochastic gradient descent approach, called `SLTF`, for finding nested subgraphs in practice.

Since hyperbolic communities are nested subgraphs, we suggest to use `SLTF` with additional constraints. The constraints need to be such that only those nested subgraphs are found that also are hyperbolic communities. Also, to be compatible with the decomposition approach of `SLTF`, we present a formulation of hyperbolic communities in terms of an outer product.

In fact, the `hyperbolic` parametrisation can already be interpreted as an outer product. The inequality $(i + p)(j + p) \leq \theta$ (see Section 4.1.1), can equivalently be stated in terms of an arithmetic progression:[2] We define

$$\mathbf{s}_n = \frac{n + p}{\sqrt{\theta}} \tag{14.2}$$

for $n = [1, \ldots, n_C]$. The positions (i, j) of the outer product $\mathbf{s}_n \mathbf{s}_n^T$ where we have

$$\mathbf{s}_n \mathbf{s}_n^T \leq 1 \tag{14.3}$$

2: An arithmetic progression is a sequence of numbers in which the difference of any two successive numbers is constant.

are those which constitute the area of the community. But, to be compatible with the `SLTF` algorithm, we need the values of the vector such that the higher values indicate where the information is. With the hyperbolic function, the behaviour is exactly the opposite: values *below* a certain threshold are those that contain the information, *i.e.* are within the community. Therefore, we propose (yet) an alternate parametrisation of our hyperbolic model:

Definition 14.1.2 *The decision function for* `converse`(q, Φ) *is*

$$f_{converse}(i, j, q, \Phi) = \left[(q - \frac{1}{i})(q - \frac{1}{j}) \geq \Phi \right]$$

with the parameter set $\Theta = \{q, \Phi\}$.

Equivalence of `converse`. As an interlude, we demonstrate that this model is related to our previously-defined characterisations.

Like for the derivation of `fixed` (see Section 4.1.2), we express the point at which the decision boundary, $(q - \frac{1}{i})(q - \frac{1}{j}) = \Phi$, crosses the diagonal (where $i = j$), and the point where the hyperbola exits the community (that is j for which $i = n_C$ or vice versa). Analogously to Equations 4.7 and 4.8, we have

$$(q - \frac{1}{\gamma})(q - \frac{1}{\gamma}) = \Phi \tag{14.4}$$

$$(q - \frac{1}{H})(q - \frac{1}{n_C}) = \Phi \tag{14.5}$$

Equating those expressions determines q by means of γ and H and thus allows to translate to our previously derived models:

$$\begin{aligned} & (q - \frac{1}{\gamma})(q - \frac{1}{\gamma}) = (q - \frac{1}{n_C})(q - \frac{1}{H}) \\ \Leftrightarrow \quad & q^2 + \frac{1}{\gamma^2} - \frac{2q}{\gamma} = q^2 - \frac{q}{H} - \frac{q}{n_C} + \frac{1}{Hn_C} \\ \Leftrightarrow \quad & q = \frac{\frac{1}{\gamma^2} - \frac{1}{Hn_C}}{\frac{2}{\gamma} - \frac{1}{H} - \frac{1}{n_C}} \end{aligned} \tag{14.6}$$

Notice that we assume 1-based indexing here, since this expression is not defined for $i = 0$ or $j = 0$. With the assumption of 1-based indexing, we remain with strictly positive values for i and j, and the discontinuity occurs outside the admitted value range.

Using the `converse` model, we can state the arithmetic progression respectively as

$$\mathbf{t}_n = \frac{q - \frac{1}{n}}{\sqrt{\phi}} \,. \tag{14.7}$$

We achieve the desired behaviour that intra-community edges are associated with values that *exceed* a threshold. To constrain SLTF to only search for nested communities that are also hyperbolic, a straight-forward approach would be to extend the algorithm with an alternating updates procedure where we map the solution of SLTF always to the closest allowed hyperbolic model $\mathbf{t}_n$.

While using SLTF seems to be a promising direction, having the hyperbolic communities as cluster centroids would require to give up on the pure binary character of Boolean tensor clustering: the centroids that admit the arithmetic progression $\mathbf{t}_n$ would be represented by means of positive real numbers. Our algorithms, however, are currently optimised for the binary domain and more investigation is needed to achieve a scalable alternative when this is no longer the case.

15 Final Conclusion

This chapter marks the end of the thesis. We conclude with a review of our contributions.

15.1 Summary of Contributions

In this thesis, we have studied two aspects of structural patterns in graph data: the description of communities by means of the hyperbolic model, and the discovery of factors by means of Boolean tensor clustering. Although being dissimilar in their nature, both strains of work contribute to understandably summarise large networks; they reveal patterns in graph data, make use of the dualism between graphs and matrices, and rely on binary data.

Part One. We have introduced the *hyperbolic community model*. The model offers very intuitive parameters to describe groups of more densely connected nodes within networks. It accounts for connectivity patterns that are frequently observed in real data, especially in social networks: a few community members are highly interconnected, while the remaining majority mainly has ties to this core.

To express the hyperbolic community model, we have presented three different but equivalent parametrisations: `hyperbolic` underlines the geometric shape of the communities; `fixed` intuitively characterises every community by means of the core size and the tail height; `mixture` highlights the probabilistic nature of community shapes. The additional graphon representation enables the perspective of an exchangeable random graph model, thereby offering options to efficiently model time-evolving communities.

Alongside with the hyperbolic community model, we have introduced the random graph generator HyGen. On the basis of the hyperbolic model, HyGen generates modular networks with realistic intra-community structures, using parameter distributions derived from observations in real graphs.

In extensive experiments on various data, we have demonstrated that our model fits real data much better than previously-proposed models, and that our random graph generator creates graphs with realistic intra-community structure. Our large scale study of the

temporal evolution of online question-answering communities has revealed that the user activity within a community is constant with respect to its size throughout its lifetime. Furthermore, we have seen that only a small group of users is responsible for the majority of the social interactions.

Part Two. We have proposed an algorithm for uncovering the structural building blocks in multiply-related binary data. As a special tensor factorisation, *Boolean tensor clustering* assumes that one of the tensor directions has only non-overlapping factors. This assumption is valid for many real-world data, in particular for time-evolving networks, and lifts much of the computational complexity from the task.

For our analysis of the theoretical quality of the algorithm, we have studied the differences between maximising the similarity and minimising the dissimilarity. While practically similar to each other, we find that using the similarity instead of distance yields a more meaningful theoretical analysis.

In our experiments, we have demonstrated that our algorithm for Boolean tensor clustering obtains results of similar quality like much less restricted tensor factorisation methods. At the same time our algorithm showed overall best scalability, obtained sparse factor matrices, and, because of the restriction to rank-1 centroids, showed good generalisation to unseen data.

Notwithstanding the contributions, the research in neither line of work is exhaustive. The most noticeable open ends that may inspire future investigations are:

- How to detect hyperbolic communities in networks?
- Can the idea of restricting the cluster centroids in Boolean tensor clustering be transferred to other methods, and what are the benefits if the rank of the centroids acts as a regularisation parameter?
- Is the consideration of similarity opposed to distance also beneficial for the analysis of other methods?

In addition, as we have discussed in Chapter 14, there is also a potential benefit of bringing together the lines of work by constraining the cluster centroids in Boolean tensor clustering to be hyperbolic communities. Exactly this combination offers a promising direction for addressing remaining challenge to detect hyperbolic communities.

Appendix

A

Proof of Proposition 4.5.1

Recall that the proposition was:

Proposition 4.5.1 *Let $C = \texttt{fixed}(\gamma, H)$ be a community of n_C nodes defined by $\gamma \in \mathbb{R}$ and $H \in \mathbb{N}$, and let A_C be its area. Then there exists integer γ' such that if $D = \texttt{fixed}(\gamma', H)$ is the community defined by γ' and H, and A_D is the respective area, then $|A_C - A_D| \in \Theta\left(\gamma \ln(n_C)\right)$.*

We will prove the claim by bounding the difference of the area between γ and $\gamma + 1$, which is clearly sufficient.

Notice first that we can assume $H = 0$: the tail part will always contribute the same area of $Hn_C - {}^{H^2}/_2$ irrespective of γ.

Consider for now the `hyperbolic` model and recall that the hyperbolic equation is

$$(x + p)(y + p) = \theta \ , \tag{A.1}$$

where we used x and y instead of i and j to emphasize their continuous nature. From (A.1) we get that

$$y = \frac{\theta}{x + p} - p \ . \tag{A.2}$$

The area of the function is the integral of (A.2) from 0 to n_C, that is,

$$A_C = \int_0^{n_C} \frac{\theta}{x + p} - p \ \mathrm{d}x \ . \tag{A.3}$$

Here we dropped the -1 from n_C for the sake of clarity; it will not effect the asymptotic analysis in any case. To get back to the `fixed` model, we can substitute

$$p = \frac{\gamma^2}{n_C - 2\gamma} \tag{A.4}$$

$$\theta = \frac{\gamma^2(\gamma^2 - n_C)^2}{(n_C - 2\gamma)^2} \ , \tag{A.5}$$

following Equations 4.10 and 4.14, respectively.

Given the constraints that $\gamma \in [0, {}^{n_C}/_2)$, we can solve the integral (A.3) with the substitutions (A.4) and (A.5) to be

$$\begin{aligned}\big(\gamma^2(4\gamma\ln(\gamma)n_C - 2\gamma^2\ln(\gamma) - 2\ln(\gamma){n_C}^2 + 2\gamma^2\ln(-\gamma+n_C)\\ - {n_C}^2 + 2\gamma n_C + 2\ln(-\gamma+n_C){n_C}^2 - 4\gamma\ln(-\gamma+n_C)n_C\big)\\ /\big({n_C}^2 - 4\gamma n_C + 4\gamma^2\big)\,. \end{aligned} \tag{A.6}$$

Hence, the difference between the areas with γ and $\gamma+1$ is

$$\begin{aligned}-\big(&-2\,n_C + \ln(\gamma^2+2\,\gamma+1)\gamma^2 n_C^2\\ &- \ln(\gamma^2+2\,\gamma-2\,\gamma n_C+1-2\,n_C+n_C^2)\gamma^2 n_C^2\\ &- 6\,\gamma n_C + \ln(\gamma^2+2\,\gamma+1)n_C^2\\ &+ \gamma^4\ln(\gamma^2+2\,\gamma+1) - \ln(\gamma^2+2\,\gamma-2\,\gamma n_C+1-2\,n_C+n_C^2)n_C^2\\ &- \gamma^4\ln(\gamma^2+2\,\gamma-2\,\gamma n_C+1-2\,n_C+n_C^2)\\ &+ \gamma^2 n_C^2 - 4\,\gamma^3\ln(\gamma^2+2\,\gamma-2\,\gamma n_C+1-2\,n_C+n_C^2)\\ &- 6\,\ln(\gamma^2+2\,\gamma-2\,\gamma n_C+1-2\,n_C+n_C^2)\gamma^2\\ &- 4\,\ln(\gamma^2+2\,\gamma-2\,\gamma n_C+1-2\,n_C+n_C^2)\gamma\\ &+ 4\,\gamma^3\ln(\gamma^2+2\,\gamma+1) + 6\,\ln(\gamma^2+2\,\gamma+1)\gamma^2\\ &+ 4\,\ln(\gamma^2+2\,\gamma+1)\gamma - 2\,\ln(\gamma^2+2\,\gamma+1)n_C\\ &+ 2\,\ln(\gamma^2+2\,\gamma-2\,\gamma n_C+1-2\,n_C+n_C^2)n_C\\ &- 2\,\gamma^3 n_C - 6\,\gamma^2 n_C + 2\,\gamma n_C^2 + n_C^2\\ &- \ln(\gamma^2+2\,\gamma-2\,\gamma n_C+1-2\,n_C+n_C^2)\\ &+ \ln(\gamma^2+2\,\gamma+1) - 2\,\gamma^3\ln(\gamma^2+2\,\gamma+1)n_C\\ &+ 6\,\ln(\gamma^2+2\,\gamma-2\,\gamma n_C+1-2\,n_C+n_C^2)\gamma^2 n_C\\ &+ 6\,\ln(\gamma^2+2\,\gamma-2\,\gamma n_C+1-2\,n_C+n_C^2)\gamma n_C\\ &- 6\,\ln(\gamma^2+2\,\gamma+1)\gamma^2 n_C + 2\,\ln(\gamma^2+2\,\gamma+1)\gamma n_C^2\\ &- 6\,\ln(\gamma^2+2\,\gamma+1)\gamma n_C\\ &- 2\,\ln(\gamma^2+2\,\gamma-2\,\gamma n_C+1-2\,n_C+n_C^2)\gamma n_C^2\\ &+ 2\,\gamma^3\ln(\gamma^2+2\,\gamma-2\,\gamma n_C+1-2\,n_C+n_C^2)n_C\big)\\ &/\big(n_C^2 - 4\,\gamma n_C - 4\,n_C + 4\,\gamma^2 + 8\,\gamma + 4\big)\\ &+\big(\gamma^2(-4\,\gamma\ln(\gamma)n_C + 2\,\gamma^2\ln(\gamma)\\ &+ 2\,\ln(\gamma)n_C^2 - \gamma^2\ln(\gamma^2 - 2\,\gamma n_C + n_C^2)\\ &+ n_C^2 - 2\,\gamma n_C - \ln(\gamma^2 - 2\,\gamma n_C + n_C^2)n_C^2\\ &+ 2\,\gamma\ln(\gamma^2 - 2\,\gamma n_C + n_C^2)n_C\big)\big/\big(n_C^2 - 4\,\gamma n_C + 4\,\gamma^2\big)\end{aligned} \tag{A.7}$$

If we denote (A.7) with $d(\gamma, n_C)$, we can notice that

$$\lim_{n_C\to\infty}\frac{\gamma\ln(n_C)}{d(\gamma,n_C)} \to \frac{1}{{}^2/_\gamma + 4}\,, \tag{A.8}$$

which is in $(0, {}^1/_4)$ for any γ, showing that $d(\gamma, n_C) \in \Theta\big(\gamma\ln(n_C)\big)$ and concluding the proof. □

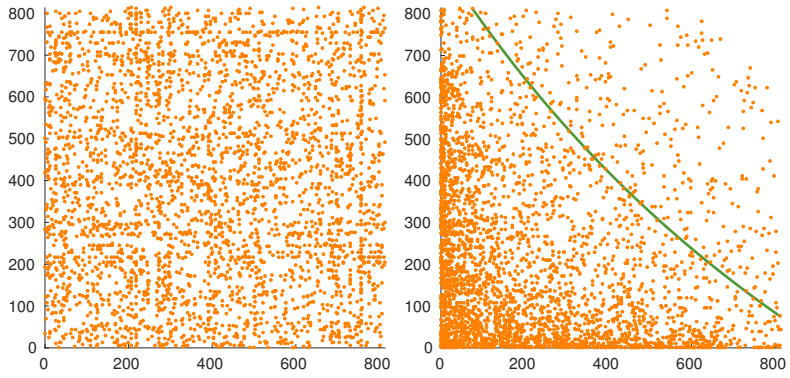
B

Examples of Modelled Communities

Models from ground-truth communities. Figures B.1 to B.5 show further examples of communities from different data sets and the models we found.

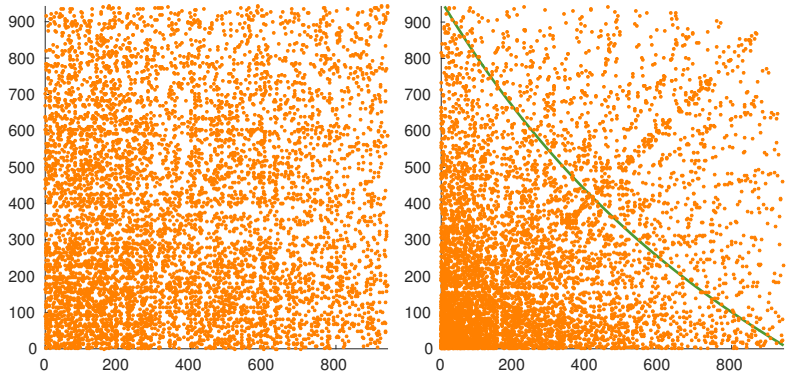
Models after spectral clustering. Figures B.6 to B.8 give additional examples of the modelled communities obtained by spectral clustering.

Models after BMF. Figures B.9 to B.12, show the models obtained for communities found by BMF.



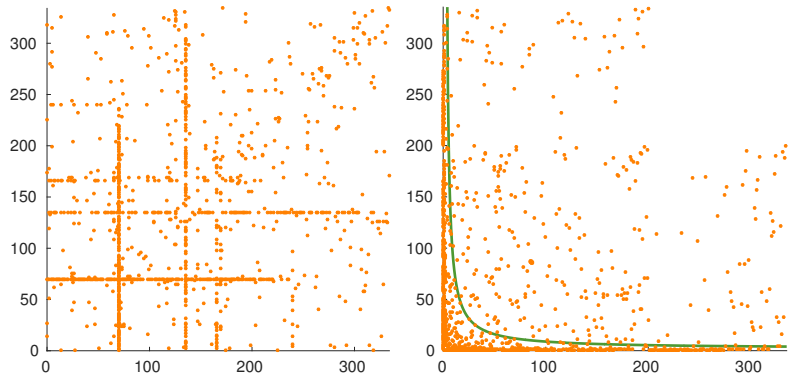
(a) Before. **(b)** After fitting the model.

Figure B.1: Example of a community obtained from the Amazon data.



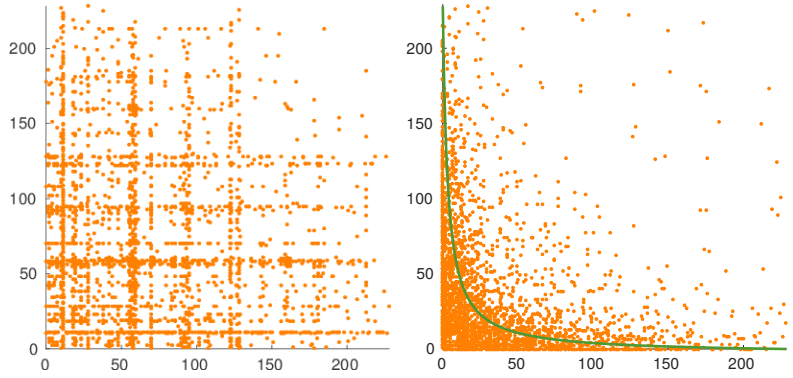
(a) Before. **(b)** After fitting the model.

Figure B.2: Example of a community obtained from the DBLP data.



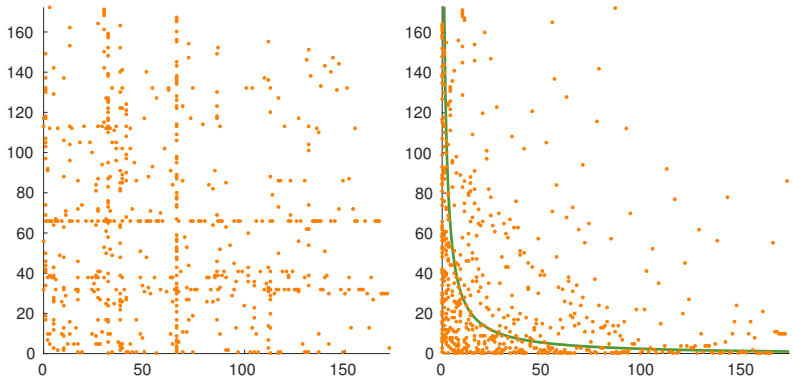
(a) Before. **(b)** After fitting the model.

Figure B.3: Example of a community obtained from the Friendster data.



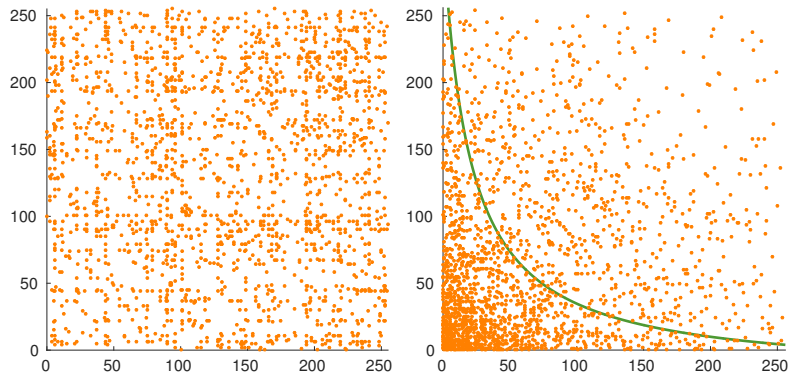
(a) Before. **(b)** After fitting the model.

Figure B.4: Example of a community obtained from the YouTube data.



(a) Before. **(b)** After fitting the model.

Figure B.5: Example of a community obtained from the YouTube data.



(a) Before. **(b)** After fitting the model.

Figure B.6: Examples of a community obtained from spectral clustering of the Erdős data.

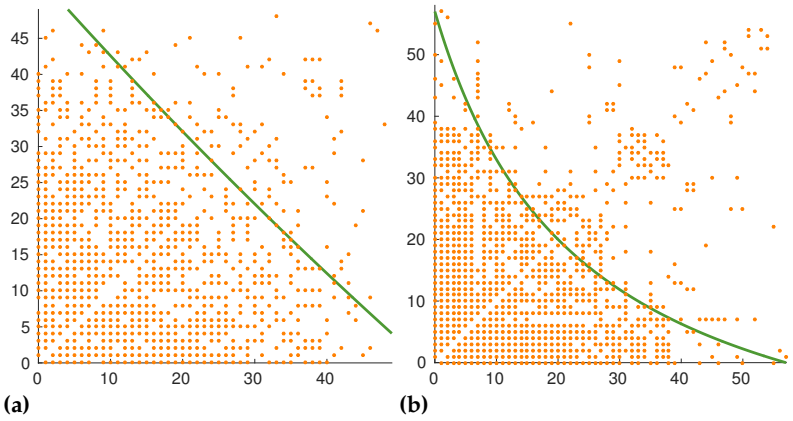


Figure B.7: Examples of communities obtained from spectral clustering of the Jazz data fitted by our model.

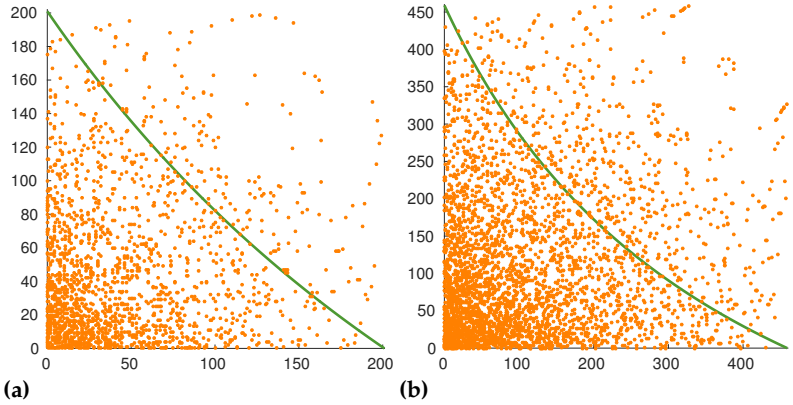


Figure B.8: Examples of communities obtained from spectral clustering of the Email data fitted by our model.

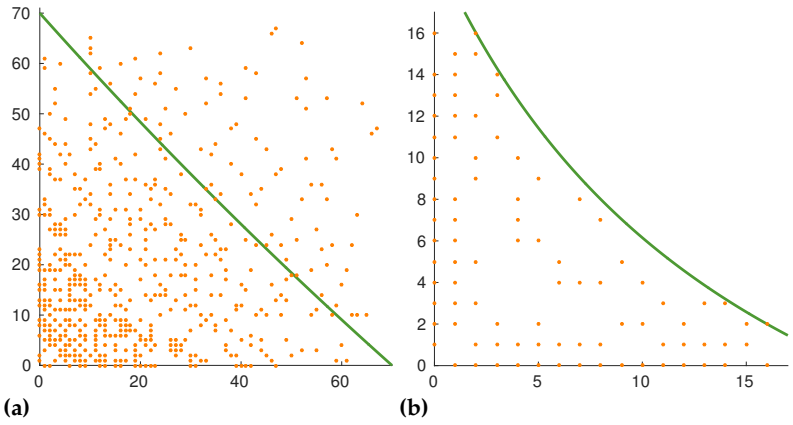


Figure B.9: Examples of communities obtained from BMF of the Email data.

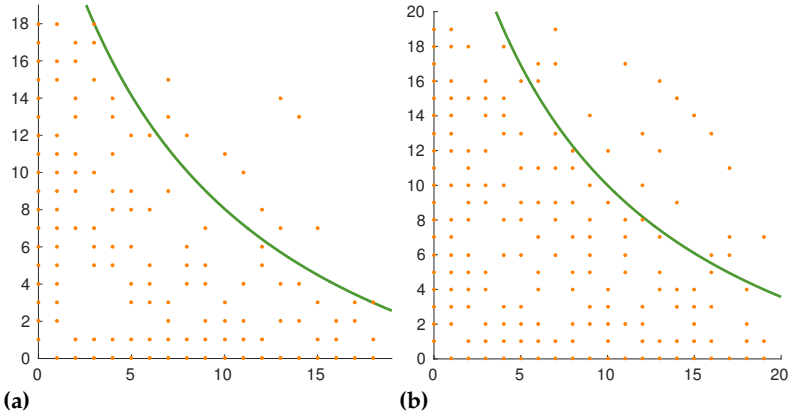


Figure B.10: Examples of communities obtained from BMF of the Erdős data.

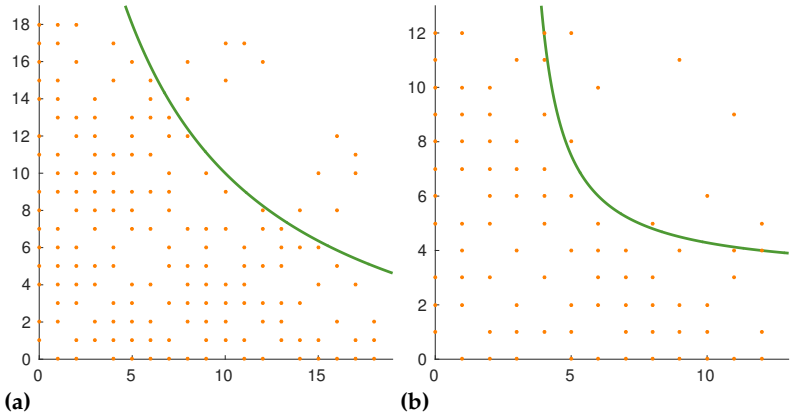


Figure B.11: Examples of communities obtained from BMF of the PolBooks data.

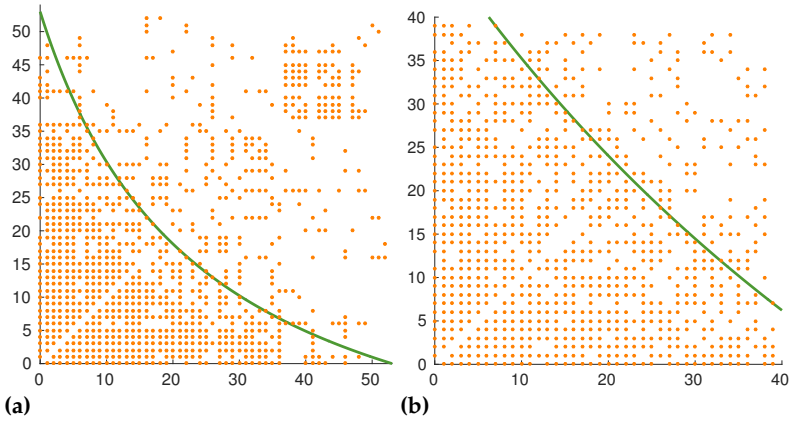


Figure B.12: Examples of communities obtained from BMF of the Jazz data.

C

Hyperbolic Communities on Question–Answer Sites

In order to further validate the presented results, we performed additional experiments which we detail in this appendix. The experiments comprise

- a comparison of different options to discretise the time axis;
- a study of higher order polynomials fitted to the time lines;
- an analysis of the correlation between highly reputed users and their position within a community,

in Appendices C.3 to C.5. In addition, we present the time lines of all communities (Appendix C.8). Furthermore, we provide additional information, tables, or figures to accompany the results presented in the thesis (Appendices C.1, C.2, C.6 and C.7):

- details about the data gathering;
- a discussion on the use of weighted data;
- a tabular summary of the numeric results from the linear regression and core stability experiments;
- a visualization of where outliers occur in the linear regression experiments with respect to community size.

C.1 Data Preparation

We use data from three large online message boards for this study. This section gives further details about the data preparation for each data set.

C.1.1 Reddit

We downloaded the Reddit data dump from `https://files.pushshift.io/reddit/` and gathered meta-data from all posts and comments up to November 2016. There are 635 048 topics, called subreddits, in total. Out of these we further consider 33 732. We ignore singleton subreddits where no replies occurred at all and also very small subreddits with less than 10 user interactions. For each subreddit of sufficient size, we list every user–user interaction

together with its time stamp, ignoring whether the contributions were made as posts or comments. To that end, we group the entries by who created a post (or comment), at what time, and in reply to which parent post by the parent ID. In a subsequent step, we construct a graph where users are connected to other users if they have created a post in reply to the post of another user. We label the connections with the time stamps of the interaction, but we do not consider the direction of the link. For users A, B, and C, where A creates a post and B and C reply, that would mean that (A, B) and (A, C) are edges of the constructed graph. If user D replies to what B posted, then (B, D) is also an edge.

C.1.2 StackExchange

We downloaded the StackExchange data dump with all posts from all 320 communities until December 2016 from https://archive.org/ and gathered meta-data from posts, answers, and comments. For further processing, we use only those posts that have a user ID assigned and are classified as either question, or answer, or comment. This pruning is necessary since in the beginning of StackExchange, comments could be made without an account. Also, deleted users' contributions are in the data dump with just the user name but no ID. We ignore the user with ID -1, which is a bot responsible for clean up tasks. As the time stamp for a contribution, we consider the creation date and ignore dates of later edits.

Every contribution, which can be either question, answer, or comment, comes with information about who created it and when. For answers and comments, we additionally know in response to which other contribution a record was created for. From this information, we construct graphs per community, where an edge is formed between two users if one created a contribution in response to the other user. Every edge is labelled with the corresponding time stamp.

C.1.3 HealthBoards

The HealthBoards dataset was collected from the site https://www.healthboards.com/ in 2013. We processed the meta-data of all 235 discussion boards, totalling 752 778 user contributions. The information we use to construct the user interaction graph per discussion board is who created a contribution in which thread at what time. This means, there is an edge between two users in the graph of one discussion board if both users have contributed to the same thread. Every edge is labelled with the corresponding time stamp of the interaction.

C.2 Edge Weight Distribution

We use unweighted data throughout the entire analysis. One alternative for representing the data would be to weight each link between users by the number of interactions they had within a given time interval and consider this weight when calculating the community model. A study comparing such an approach to ours is beyond the scope of this work. We, however, suspect that little additional information is gained by analysing weighted networks: for StackExchange, we observe that about 2/3 of the edges have a weight of 1, and 90 % of all edges are covered if we count edges of weight 1, 2, and 3 (see Figure C.1).

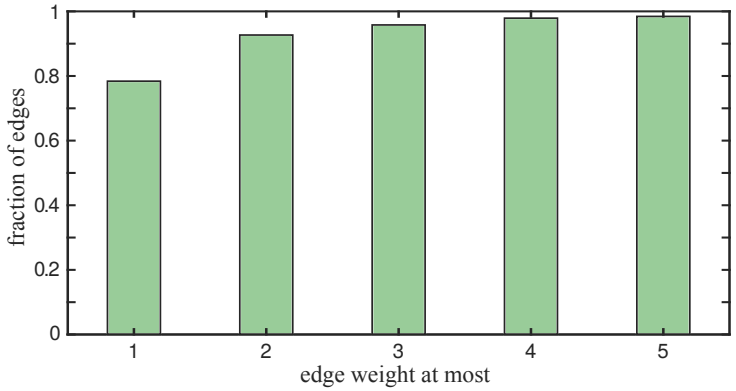
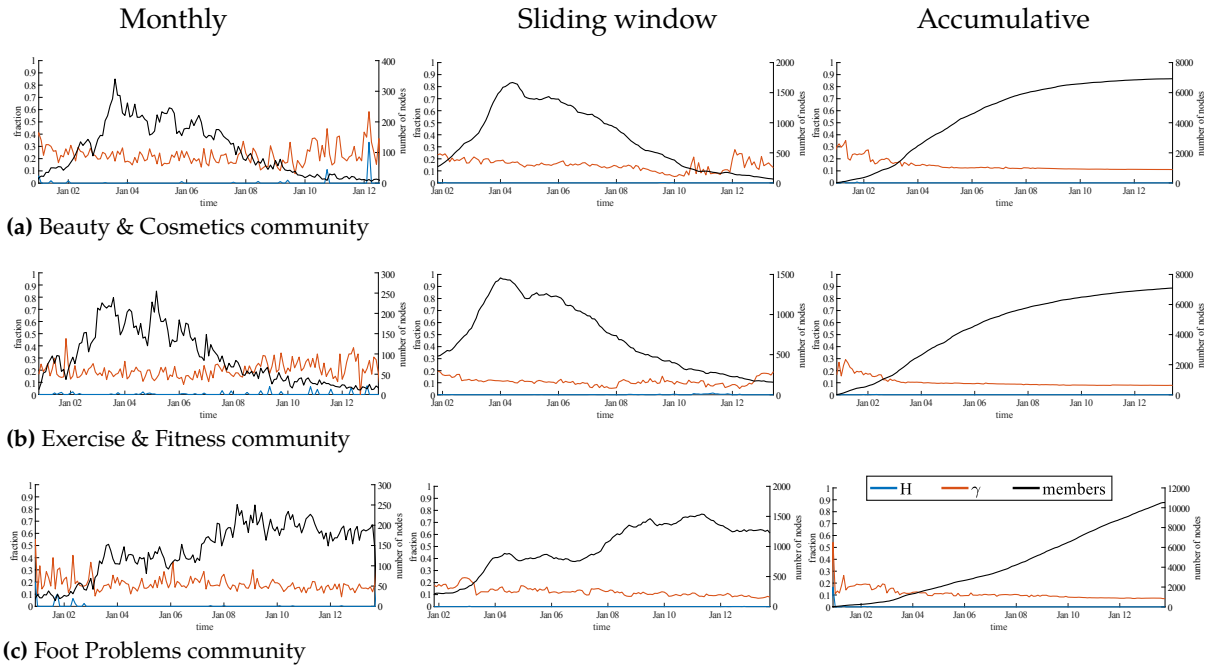


Figure C.1: Cumulative plot of the fraction of edges covered with at most weight x of all monthly time slices of StackExchange (including meta-communities).

C.3 Different Time Aggregation Options

Each interaction in the datasets is labelled with its date. As for the massive amount of data, it is computationally infeasible to re-model each community after every new interaction. Therefore, we chose monthly intervals to discretise the time axis as a trade-off between computational effort and a fine-grained view on the evolution of the communities.

This modelling decision, however, still leaves options for how to treat each monthly time interval. While in Chapter 8, we present the findings that consider each of these intervals individually – which is in fact the most difficult setting – we also examined two other options: a sliding window approach and an accumulative setting where edges are never removed. Here, we compare these three options against each other using the HealthBoards dataset, which has the smallest communities on average. We chose this dataset because smaller communities exhibit higher degrees of variance, and thus, the smoothing effect of the two alternative options is more pronounced. As a side effect, this dataset requires the least



(a) Beauty & Cosmetics community

(b) Exercise & Fitness community

(c) Foot Problems community

Figure C.2: Comparison of time development using different time aggregation options after the example of selected HealthBoards communities. Modelling results for all further communities for the respective time aggregation option are displayed in Figures C.15 to C.17.

amount of computation, which is advantageous as particularly the accumulative setting yields substantially larger communities to be modelled in every time step.

C.3.1 Monthly

When considering the interactions of users in every month individually, the data of one time step is independent of data from the previous time steps. As detailed in Chapter 8, even in this hardest setting, where the variance of the model parameters is the highest (compare Table C.1 and Figure C.3), we are able to identify the common pattern of a small group of users who are constantly responsible for the majority of the social interactions throughout each time line.

C.3.2 Sliding Window

For aggregation in a sliding window fashion, we consider all the interactions of the current month plus those of the preceding eleven months. With this window of one year, we go along the time dimension of the communities and compute the model month by month.

Examples of the model evolution over time can be seen in Figure C.2 in the middle column. Compared to the set-up where every month is regarded individually, we observe less fluctuation

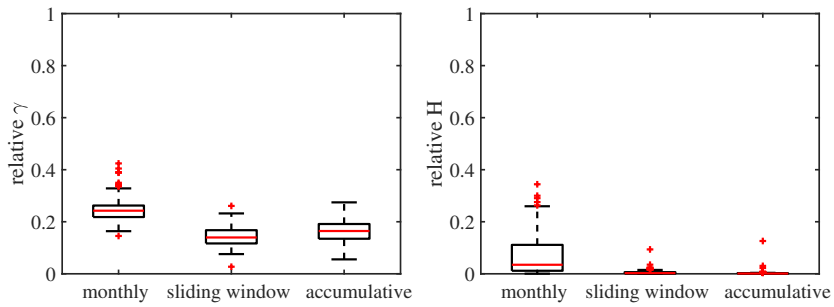
	monthly		sliding window		accumulative	
	average	SD	average	SD	average	SD
average γ	0.246	0.044	0.165	0.041	0.143	0.036
avg. of SD of γ	0.160	0.075	0.062	0.032	0.065	0.030
average H	0.071	0.081	0.002	0.009	0.005	0.008
avg. of SD of H	0.134	0.100	0.007	0.022	0.031	0.037
average slope of γ	−0.000	0.000	−0.000	0.000	−0.000	0.000
RMSE of fit	0.155	0.067	0.057	0.031	0.053	0.026
avg. correlation	1.000	0.000	1.000	0.000	1.000	0.000
p-value of corr.	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001

Table C.1: Statistics of model parameters for different time aggregation options on the HealthBoards data set. We consider only communities with more than 100 nodes in total that cover a time span of more than 12 months. The correlation is computed between the data and the constant model after transforming the variable space (as explained in Section 8.5).

in the parameters H and γ. This is expected for two reasons: First, the sliding window smooths the differences in the data from one month to the next. Second, as for every time step, the interactions of one year are aggregated, the effective size of the community for which the model is computed is much larger, and the model thus becomes more stable. Importantly, we observe the constant behaviour of the model parameters. This impression is backed up by the statistical analysis of all the model parameters summarised in Table C.1 (and Figure C.3): we verify that the size of the community core, indicated by the parameter γ, is near constant over time; furthermore, approximately 20 % of the community members constitute the core.

C.3.3 Accumulation Over Time

We analyse the community structure evolution when accumulating all interactions from the beginning of the time series to the current month. As observable in Figure C.2 in the rightmost column, the number of interactions steadily grows while the relative core size remains constantly small. Statistics summarising all datasets are given in Table C.1 (and Figure C.3). This analysis confirms what we observed with the monthly time steps and with the sliding



(a) Distribution of the core size γ **(b)** Distribution of tail height H

Figure C.3: Distribution of the model parameters H and γ on the HealthBoards data using different time aggregation approaches.

window approach: the relative size of the core of each community is constant and has approximately 20 % of the size of the whole community.

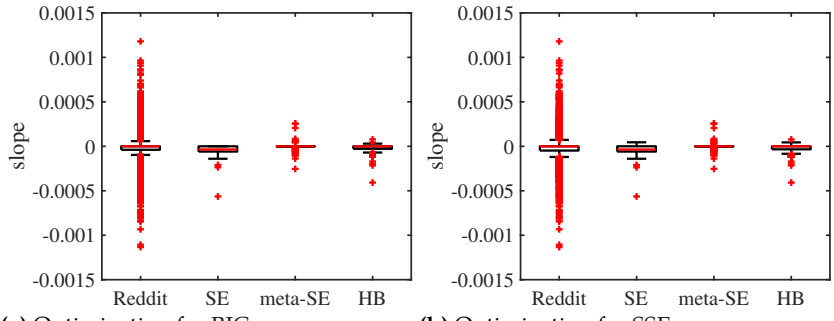
Thus, we conclude that our results, are not dependent on the way the data is aggregated, suggesting that the patterns we observe are a truly inherent property of the user interaction data.

Admittedly, for the HealthBoards data, the sliding window setting would lead to cleaner looking results in the sense that fluctuations would be smoothed, thus only leaving the potentially interesting discontinuities in the core size. Analysing these further might be interesting for practitioners. We nevertheless decided to use the monthly set-up as a common data preparation procedure for all the examined data sets. This decision not only accounts for the size of StackExchange and Reddit communities, but also opposes the argument that the observed schema of intra-interaction would be a result of the data preparation.

C.4 Constant Model versus Higher Order Polynomials

We validate that the constant model is an appropriate model to describe the time evolution of the core size γ. To that end, we fit polynomials to each time line, take the best, and compare its quality to that of the constant model. For fitting the polynomials, we use Matlab's `polyfit` function and optimise once for the Bayesian information criterion (BIC) and once for the sum of squares error (SSE). We evaluate the models with respect to the mean square error (MSE).

First of all, we notice that although we permit polynomials up to degree nine, the best models are never more complex than a first order polynomial. Often, the best model is exactly a constant model. In some cases, it is linear with a small slope in addition



(a) Optimization for BIC **(b)** Optimization for SSE

Figure C.4: Distribution of the slope parameter of the fitted models.

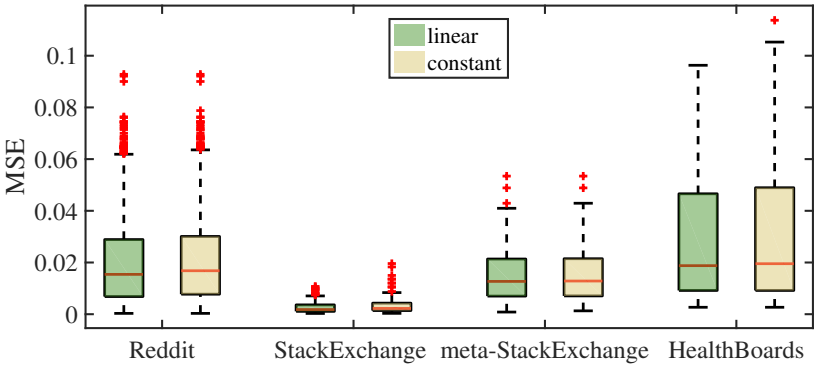


Figure C.5: Distribution of the MSE of linear models fitted to each community in comparison to that of constant models. Models where obtained after optimization for BIC.

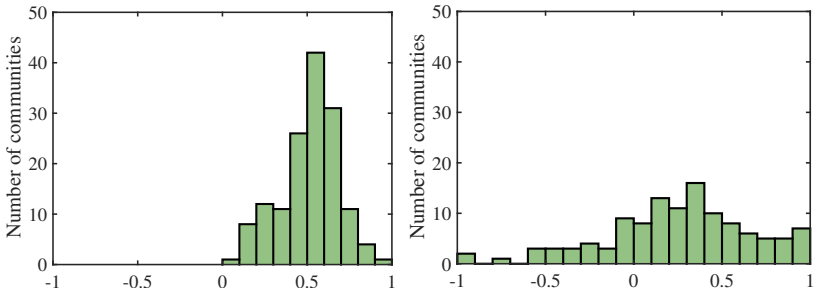
to the intercept. Figure C.4 shows the distribution of slopes in the best models of each community and per dataset. Notice that a slope of around 0.001, which is what we observe for the most extreme outliers in the Reddit data, would alter the parameter γ by 0.1 after 8.3 years. There is almost no difference between the results of optimization for BIC and for SSE.

In Chapter 8, we claim that the constant model is a good fit for every analysed community. To assess how much this simplification impacts the model quality, we compare the MSE of each best model to that of each respective constant model. The distributions are visualised in Figure C.5. We notice that the difference in MSE is typically negligibly close to zero, suggesting that the constant model is sufficient to explain the data.

C.5 Reputation of the Core Members

The StackExchange users gain or lose reputation points based on the quality of their interactions. Reputation determines a user's privileges within the system and reflects familiarity with the site as well as subject matter expertise. One might assume that the community cores identified by the hyperbolic model predominantly consist of highly reputed members. To see whether this assumption is true, we compare the hyperbolic model results to the result obtained from purely analysing the reputation scores of each user. To that end, we use the model of the last time step of every StackExchange community and compare its ranking of the users, which is the degree order of the nodes, to the ranking of the users by their reputation.

Supported by a Pearson correlation test, we observe a mediocre connection between the rankings. The mean correlation is 0.4 with a noticeable SD of 0.33 (see Figure C.6). This finding shows that an analysis of the reputation is not enough to identify the community



(a) StackExchange **(b)** Meta-communities of StackExchange

Figure C.6: Distribution of Pearson correlation coefficients calculated for every StackExchange community between the users ordered by their reputation score and ordered by their activity (*i. e.* node degree) in the last month of the time series. We display the distribution within the meta-communities separately.

core. The low but positive correlation between activity and reputation underlines that reputation is gained through activity, but the activity of a user cannot be inferred from the user's reputation.

C.6 Numeric Results for Fitting Community Models and Core Stability

In Chapter 8, we argue that the shape of the community models remains constant throughout time and despite changes in community size in each of the data sets. In particular, we state that the slope of the line fitted to γ over its time and size evolution is 0 (see Sections 8.5 and 8.6). Furthermore, we observe that a substantial fraction of users remains in the core of the community from one time step to the next (see Section 8.7).

Table C.2 shows supporting statistics and summarises the amounts of overlap we observe between the cores.

While the averages and SDs of the model parameters as well as the fit of a linear function on the time evolution of γ are straightforward to compute, the p-values to assess the significance of the results require further explanation: the p-values of the coefficients in the linear regression are very high. This is unsurprising since the tested null hypothesis for the slope coefficient is 'the slope is equal to 0'. We find no evidence against this hypothesis. In an additional test, explained in Section 8.5, we assess the correlation between the parameter γ over time and community size and a constant function to confirm the significance.

Table C.2: Statistics of model parameters. We consider all communities with more than 100 nodes in total that cover a time span of more than 12 months. Linear regression is computed with respect to time and community size. The correlation is between the data and the constant model after transforming the variable space. Core overlap is computed between time-wise subsequent cores and cores 2, 4, and 8 months apart.

	averages				standard deviations			
	Reddit	StackExchange forums	StackExchange meta	HealthBoards	Reddit	StackExchange forums	StackExchange meta	HealthBoards
average γ	0.242	0.145	0.267	0.246	0.060	0.034	0.039	0.044
avg. of SD of γ	0.132	0.054	0.119	0.160	0.053	0.024	0.046	0.075
average H	0.065	0.011	0.057	0.071	0.060	0.009	0.034	0.081
avg. of SD of H	0.124	0.026	0.090	0.134	0.082	0.029	0.048	0.100
Regression on time as x-axis								
avg. slope of γ	−0.000	−0.000	−0.000	−0.000	0.000	0.000	0.000	0.000
avg. p-value of slope	0.312	0.123	0.363	0.224	0.313	0.228	0.309	0.304
RMSE of fit	0.128	0.048	0.118	0.155	0.054	0.021	0.047	0.067
avg. correlation	1.000	1.000	1.000	1.000	0.000	0.000	0.000	0.000
p-value of correlation	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001
Regression on size as x-axis								
avg. slope of γ	−0.001	−0.000	0.000	0.007	0.003	0.000	0.004	0.014
avg. p-value of slope	0.277	0.071	0.287	0.142	0.300	0.020	0.325	0.234
RMSE of fit	0.128	0.046	0.117	0.155	0.055	0.177	0.047	0.072
avg. correlation	1.000	1.000	1.000	0.999	0.000	0.000	0.000	0.003
p-value of correlation	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001
Average core overlap								
1 month apart	0.397	0.564	0.603	0.467	0.135	0.072	0.063	0.141
2 months apart	0.358	0.494	0.557	0.410	0.138	0.078	0.068	0.157
4 months apart	0.334	0.443	0.531	0.362	0.147	0.080	0.069	0.161
8 months apart	0.315	0.391	0.493	0.321	0.161	0.082	0.075	0.165

C.7 Regression with Respect to Community Size

Figure C.7 provides additional insights for the distributions of slopes summarised as boxplots in Figure 8.7. We observe that for communities with a sufficient average monthly size, there exists very little variation in the slope. Outliers primarily occur when the communities have too few members to derive a reliable model. It appears that the regression on community size as the x-axis is more severely affected, showing stronger deviations from zero in the slope than when the regression is computed on time as the x-axis.

It appears to be a particularity of the HealthBoards data that outliers for the slope of γ with respect to the community size show primarily positive values, while being much more evenly spread in the other

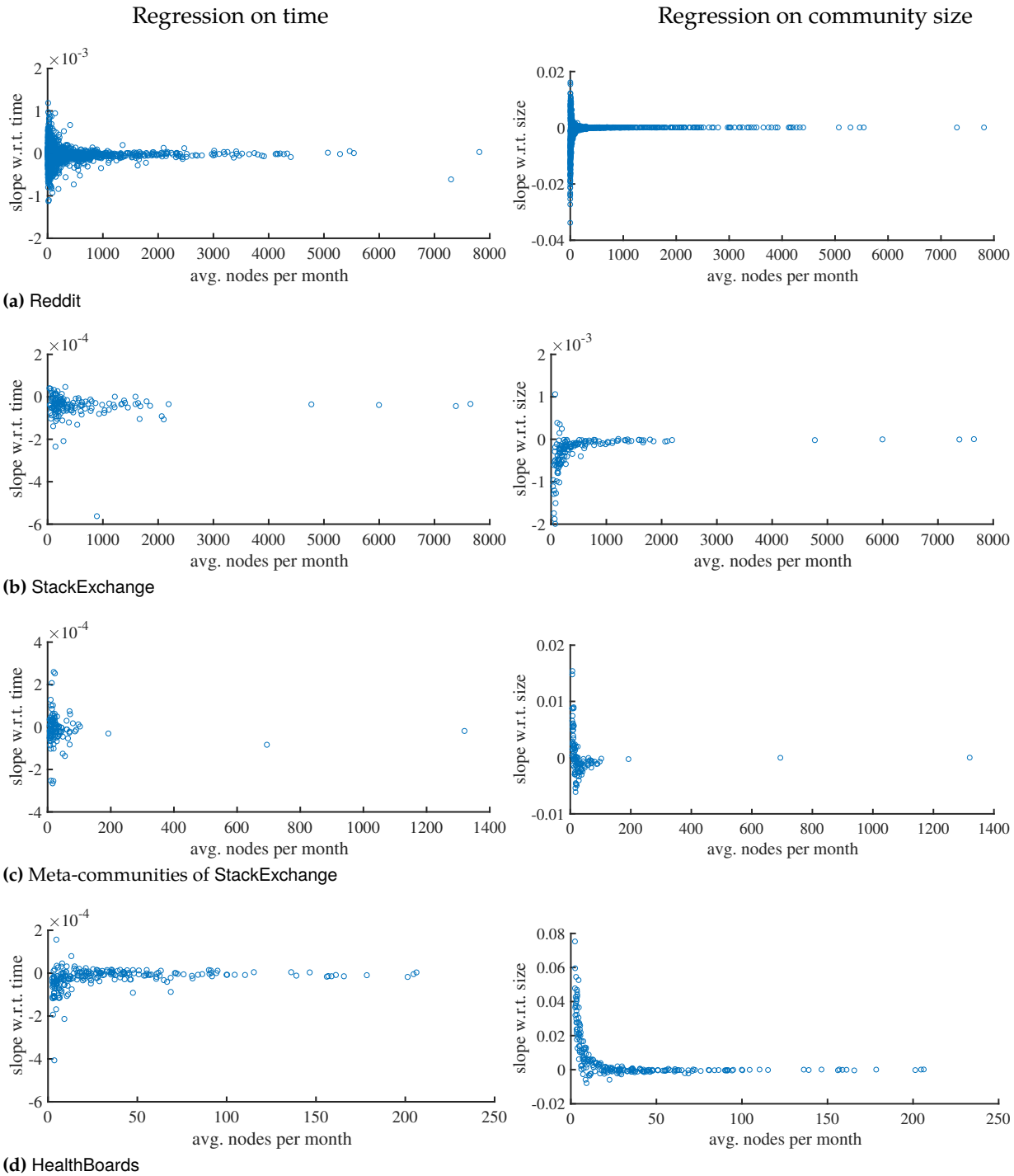


Figure C.7: Scatter plots of the slope for γ versus the average monthly community size. Note that axes are not scaled to a common size.

data sets (compare Figure 8.7b). It seems more likely that this observation is an artefact of inaccurate hyperbolic models on such small data – on average, below 20 nodes – rather than a hint on a special pattern where the smallest of the analysed HealthBoards communities have an actual tendency to grow larger cores as their sizes increase.

C.8 Time Evolution of All Communities

We display the time evolution of the communities of Reddit in Figures C.8 to C.13, of StackExchange in Figure C.14, and of HealthBoards in Figure C.15.

Figures C.16 and C.17 show the time evolution for the HealthBoards communities when using the accumulative and sliding window time aggregation approaches, respectively. As for the number of individual communities, we display only the larger ones with sufficient size in these collages and encourage the interested reader to zoom in using the electronic version of this document.

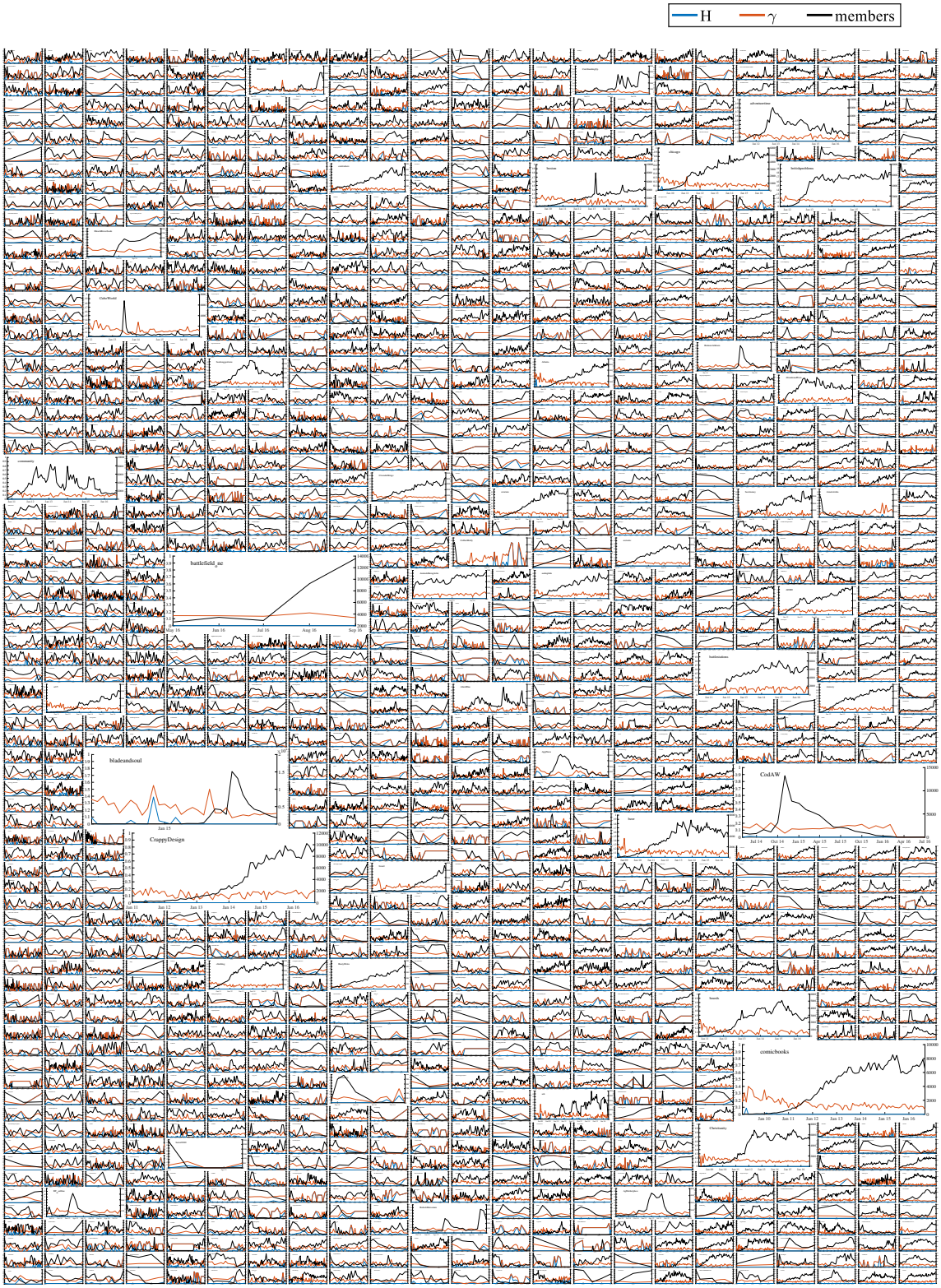


Figure C.8: Time evolution graphs of all Reddit communities starting with A, B, C, or a non-letter. In every diagram, the x-axis indicates time, the left y-axis the model parameters H and γ, and the right y-axis the size of the respective community.

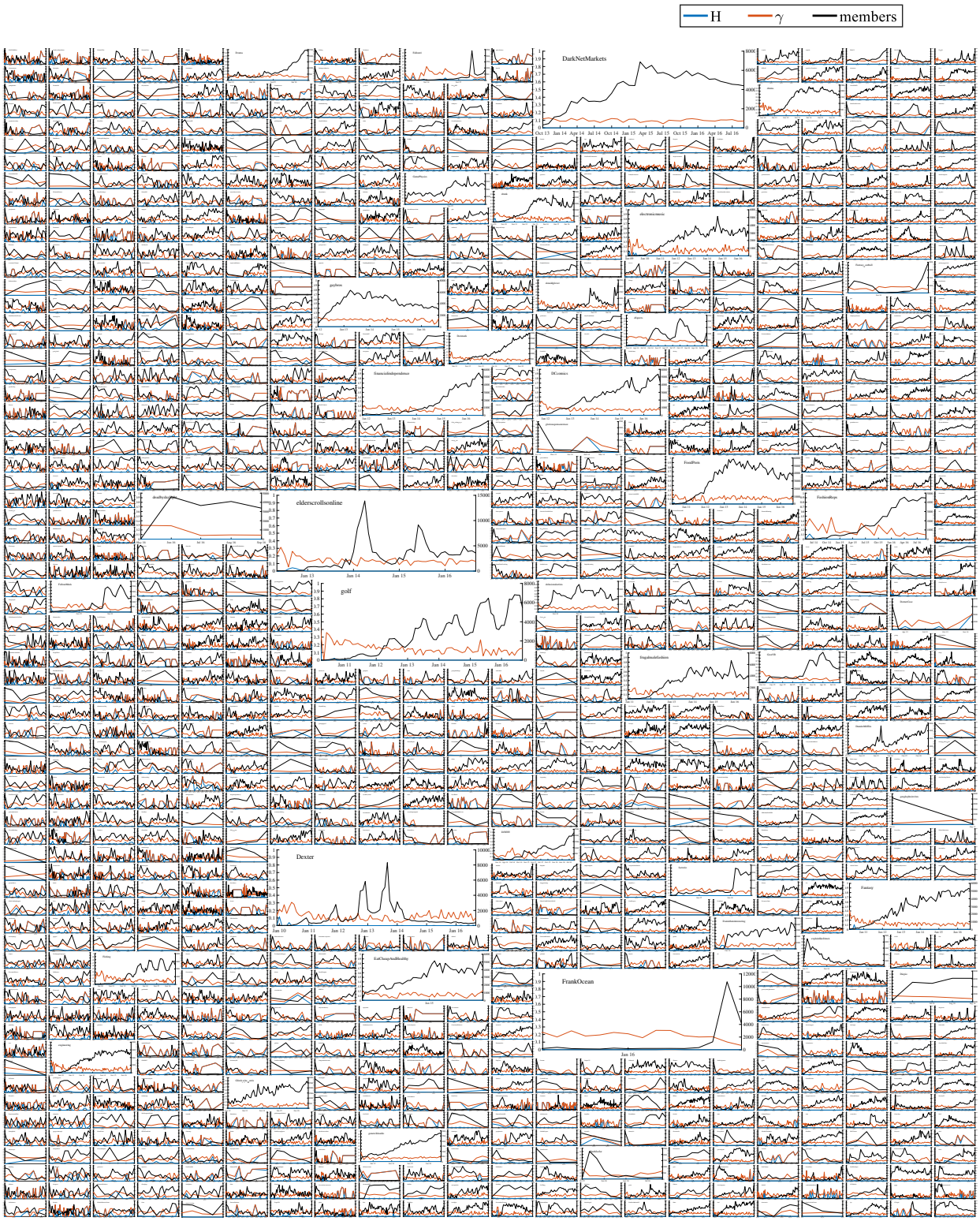


Figure C.9: Time evolution graphs of all Reddit communities starting with letters D–G. In every diagram, the x-axis indicates time, the left y-axis the model parameters H and γ, and the right y-axis the size of the respective community.

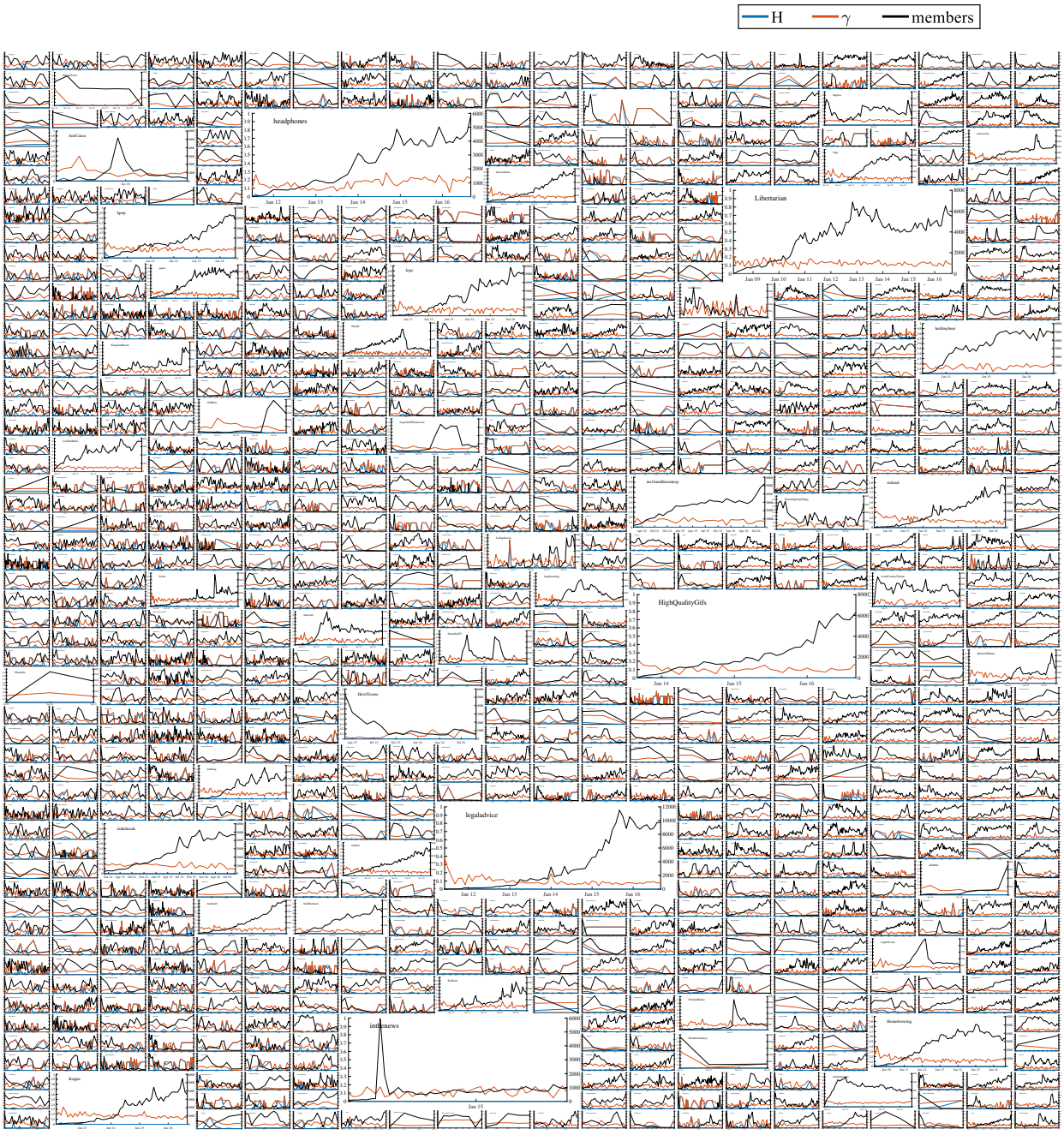


Figure C.10: Time evolution graphs of all Reddit communities starting with letters H–L. In every diagram, the x-axis indicates time, the left y-axis the model parameters H and γ, and the right y-axis the size of the respective community.

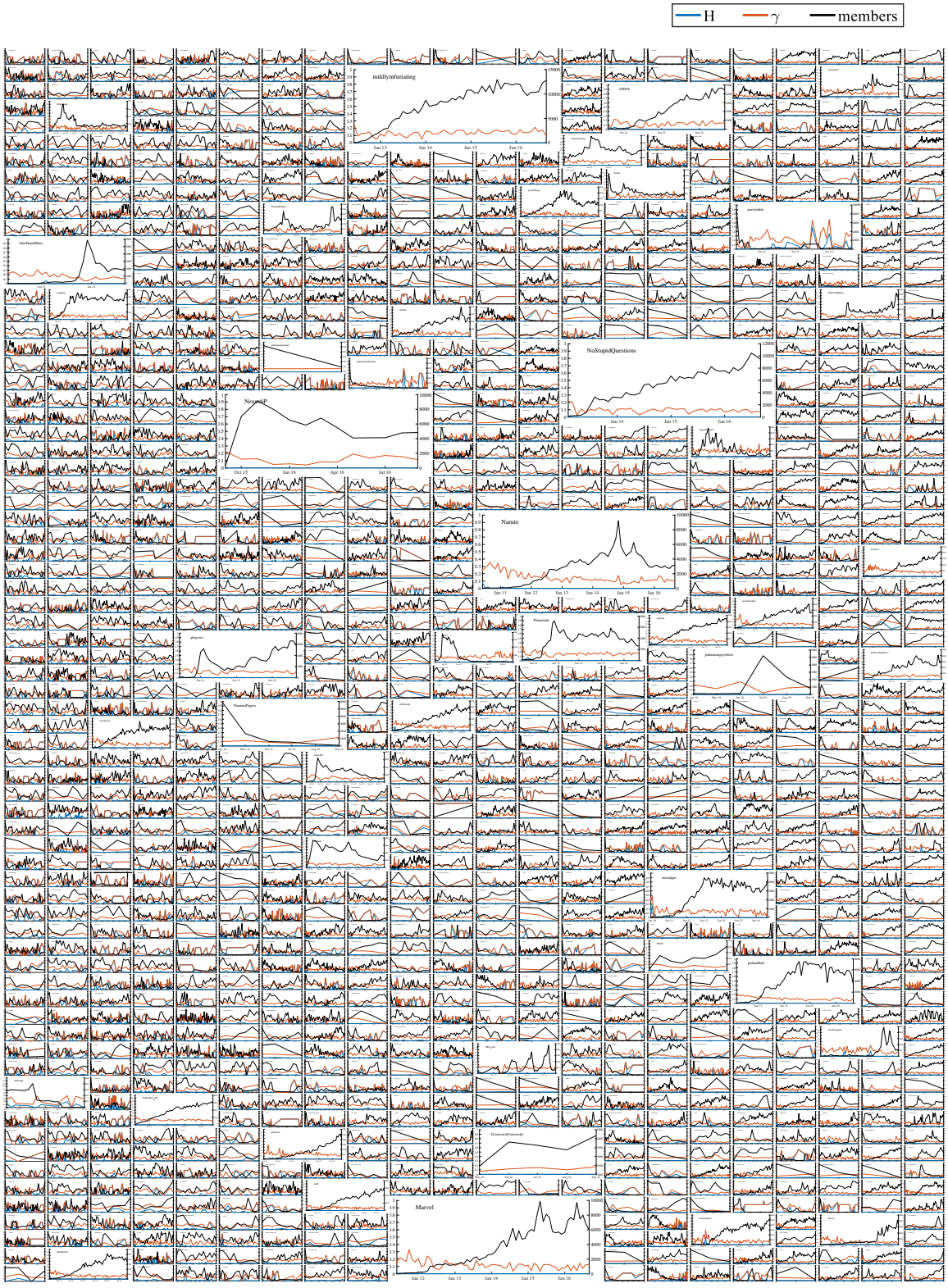


Figure C.11: Time evolution graphs of all Reddit communities starting with letters M–P. In every diagram, the x-axis indicates time, the left y-axis the model parameters H and γ, and the right y-axis the size of the respective community.

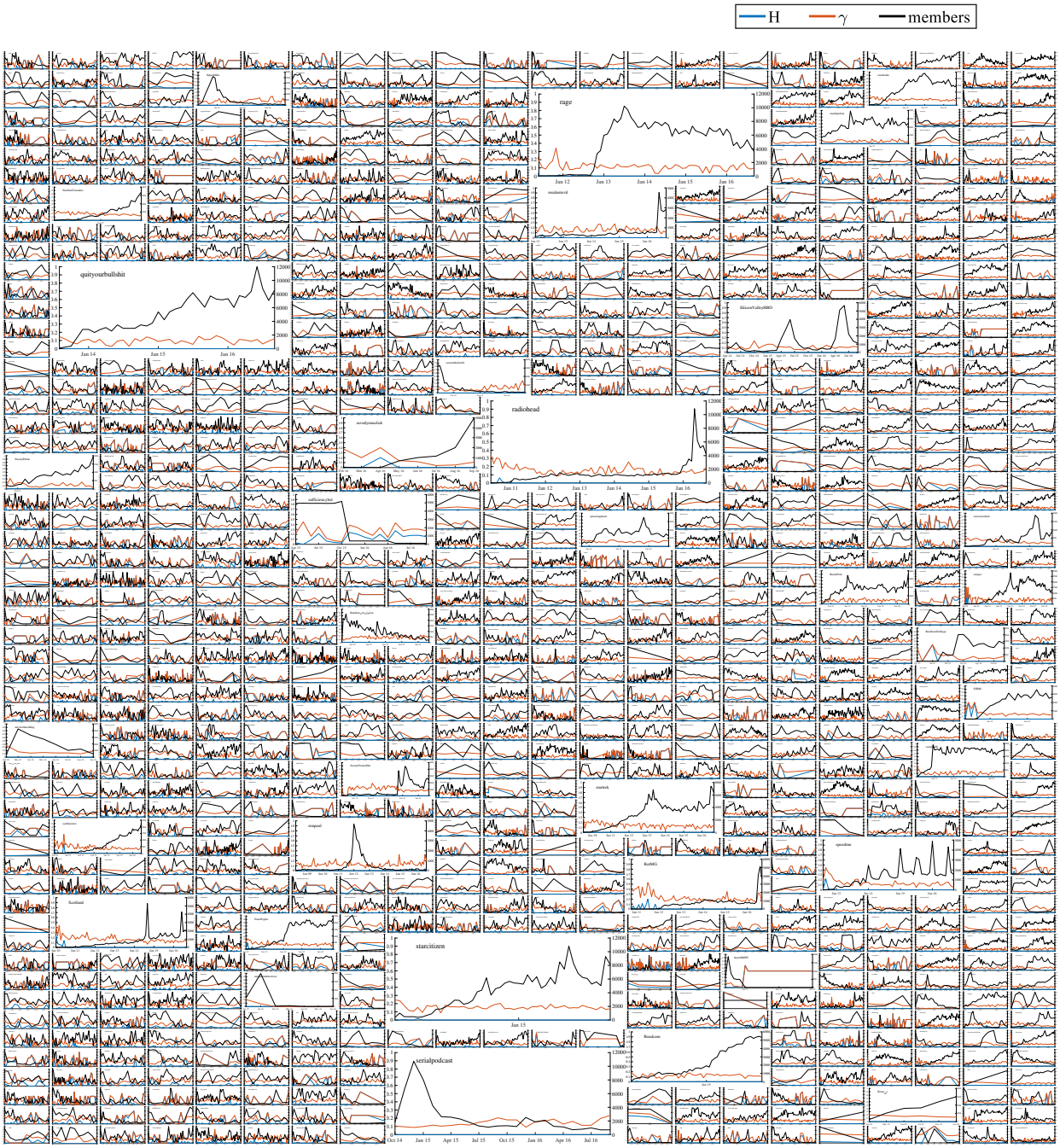


Figure C.12: Time evolution graphs of all Reddit communities starting with letters Q–S. In every diagram, the x-axis indicates time, the left y-axis the model parameters H and γ, and the right y-axis the size of the respective community.

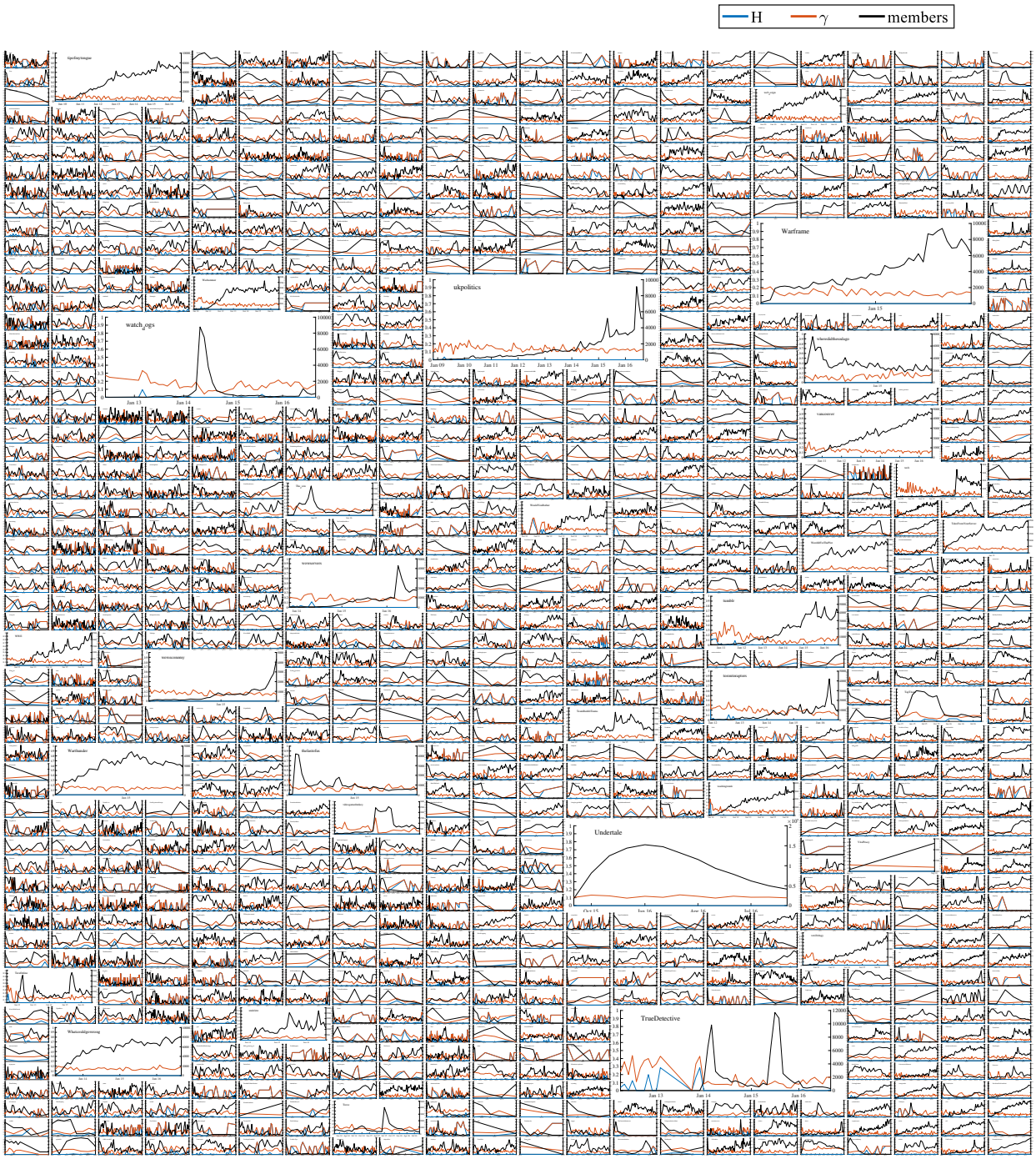


Figure C.13: Time evolution graphs of all Reddit communities starting with letters T–Z. In every diagram, the x-axis indicates time, the left y-axis the model parameters H and γ, and the right y-axis the size of the respective community.

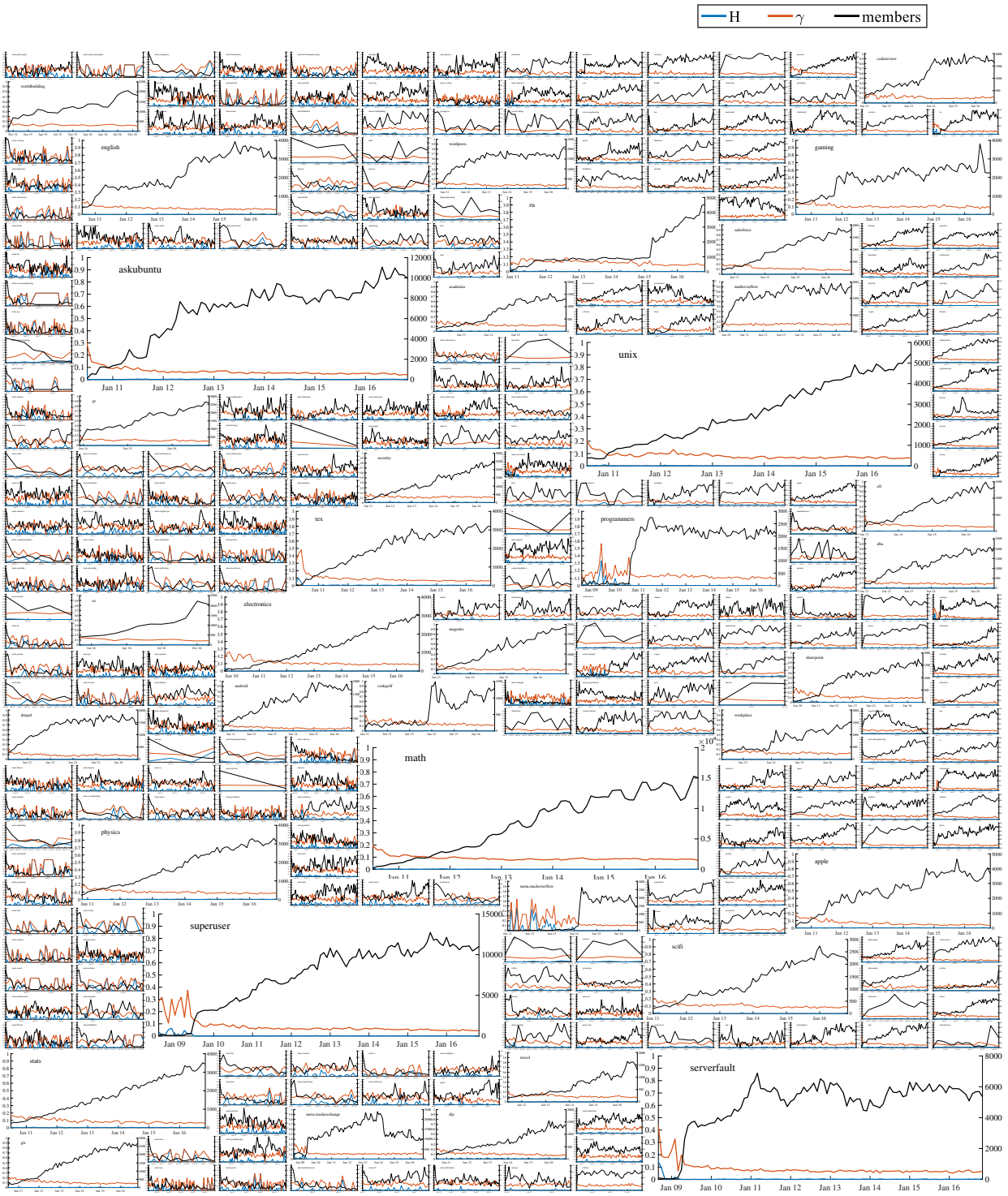


Figure C.14: Time evolution graphs of all StackExchange communities. In every diagram, the x-axis indicates time, the left y-axis the model parameters H and γ, and the right y-axis the size of the respective community.

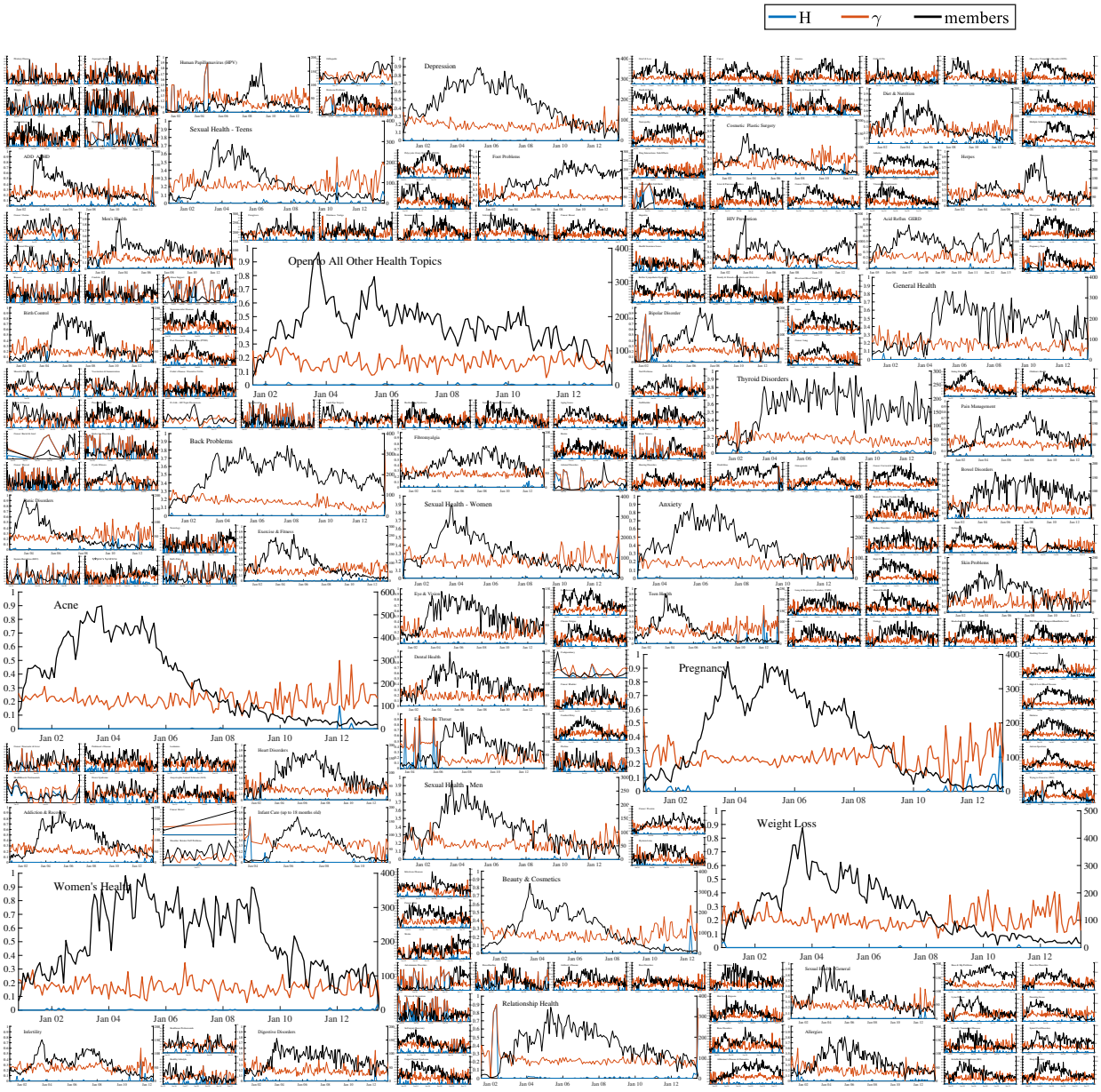


Figure C.15: Time evolution graphs of the HealthBoards communities with monthly time aggregation. In every diagram, the x-axis indicates time, the left y-axis the model parameters H and γ, and the right y-axis the size of the respective community.

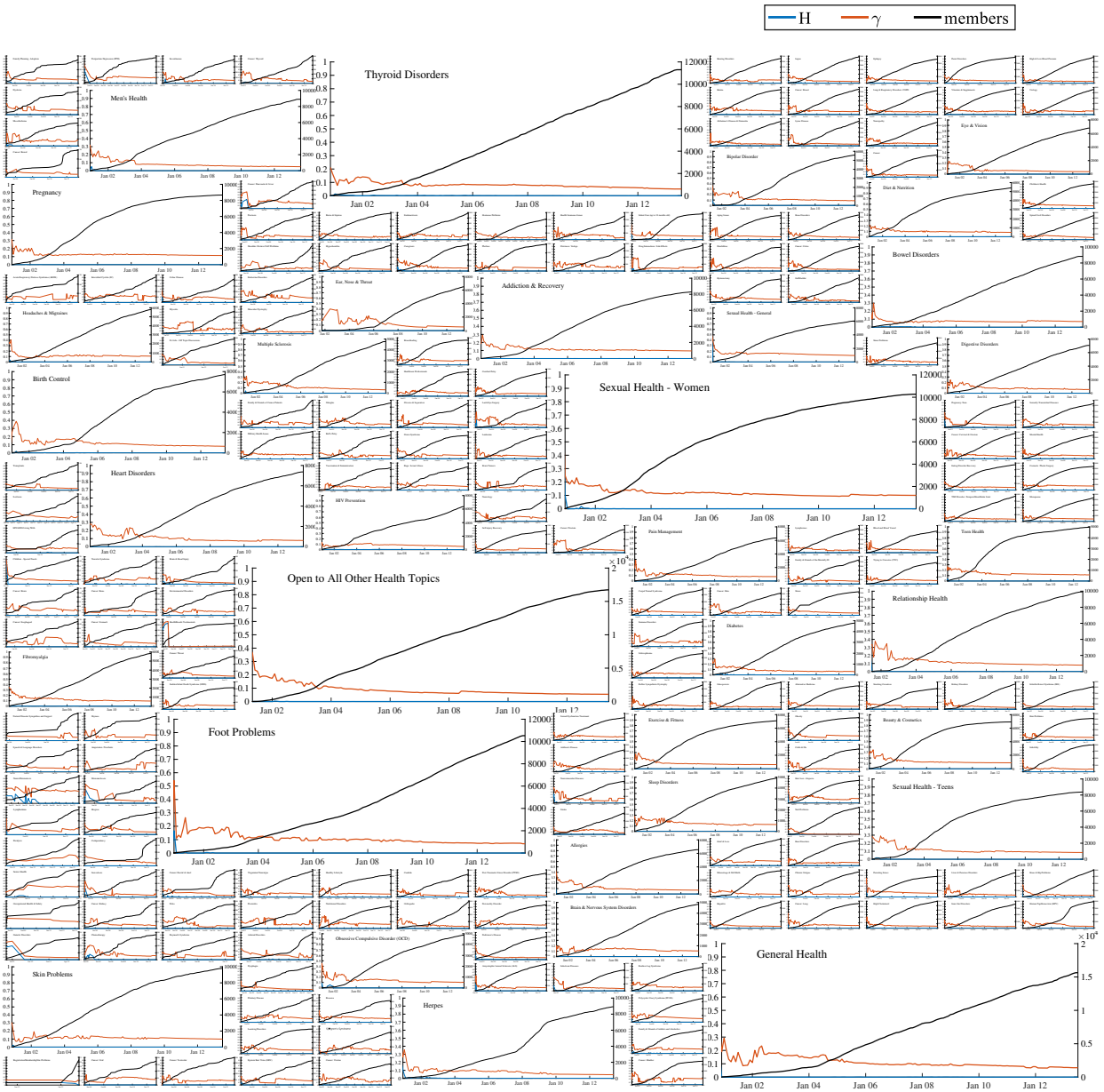


Figure C.16: Time evolution graphs of all HealthBoards communities with accumulative time aggregation. In every diagram, the x-axis indicates time, the left y-axis the model parameters H and γ, and the right y-axis the size of the respective community.

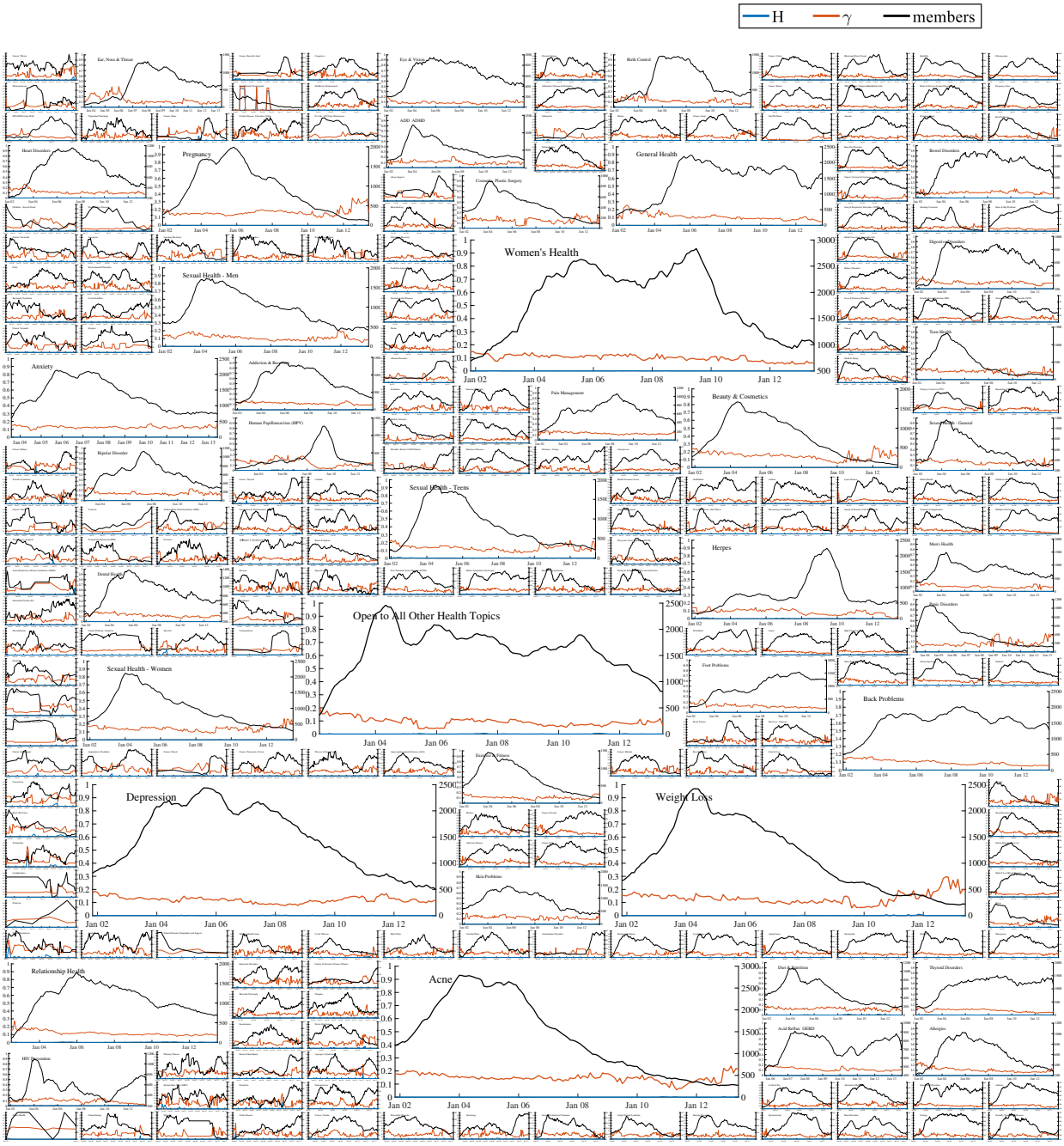


Figure C.17: Time evolution graphs of all HealthBoards communities with sliding window time aggregation. In every diagram, the x-axis indicates time, the left y-axis the model parameters H and γ, and the right y-axis the size of the respective community.

Bibliography

We list the references in citation order. In the electronic version, titles are hyperlinks, using the DOI whenever available; alternatively links refer to arXiv, or further (hopefully) stable resources. Page numbers are hyperlinks to allow navigating back to the occurrence of a citation.

[1] N. P. Gopalan and B. Sivaselvan. *Data Mining: Techniques and Trends*. First Edition. New Delhi, India: PHI Learning, 2009 (cited on page 1).

[2] Heikki Mannila. 'Local and Global Methods in Data Mining: Basic Techniques and Open Problems'. In: *Proceedings of the 29th International Colloquium on Automata, Languages and Programming*. (ICALP'02 in Malaga, Spain). Edited by Peter Widmayer, Francisco Triguero Ruiz, Rafael Morales Bueno, Matthew Hennessy, Stephan J. Eidenbenz, and Ricardo Conejo. Volume 2380. Lecture Notes in Computer Science. Berlin Heidelberg: Springer, 2002, pages 57–68 (cited on page 1).

[3] Miguel Araujo, Stephan Günnemann, Gonzalo Mateos, and Christos Faloutsos. 'Beyond blocks: Hyperbolic community detection'. In: *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases*. (ECML PKDD'14 in Nancy, France). Edited by Toon Calders, Floriana Esposito, Eyke Hüllermeier, and Rosa Meo. Berlin Heidelberg: Springer, 2014, pages 50–65 (cited on pages 4, 11, 17, 18, 21, 33, 37, 44, 54, 57).

[4] Reinhard Diestel. *Graph Theory*. Fourth Edition. Volume 173. Graduate Texts in Mathematics. Berlin Heidelberg: Springer Nature, 2017 (cited on page 7).

[5] Albert-László Barabási and Márton Pósfai. *Network Science*. Cambridge, UK: Cambridge University Press, 31, 2016. 475 pages (cited on pages 7–9, 11).

[6] Charo I. Del Genio, Thilo Gross, and Kevin E. Bassler. 'All Scale-Free Networks Are Sparse'. In: *Physical Review Letters* 107.17 (2011), page 178701 (cited on page 9).

[7] Peter J. Mucha, Thomas Richardson, Kevin Macon, Mason A. Porter, and Jukka-Pekka Onnela. 'Community structure in time-dependent, multiscale, and multiplex networks'. In: *Science* 328.5980 (2010), pages 876–878 (cited on page 10).

[8] Danai Koutra, U Kang, Jilles Vreeken, and Christos Faloutsos. 'VoG: Summarizing and Understanding Large Graphs'. In: *Proceedings of the 14th SIAM International Conference on Data Mining*. (SDM'14 in Philadelphia, PA, USA). Edited by Mohammed Zaki, Zoran Obradovic, Pang Ning Tan, Arindam Banerjee, Chandrika Kamath, and Srinivasan Parthasarathy. Volume abs/1406.3411. Philadelphia: Society for Industrial and Applied Mathematics, 2014, pages 91–99 (cited on page 10).

[9] Stephen P. Borgatti and Martin G. Everett. 'Models of core/periphery structures'. In: *Social Networks* 21 (1999), pages 375–395 (cited on pages 11, 18, 20, 21, 33).

[10] M. Girvan and M. E. J. Newman. 'Community structure in social and biological networks'. In: *Proceedings of the National Academy of Sciences* 99.12 (2002), pages 7821–7826 (cited on pages 11, 23).

[11] Vincenzo Leo, Giovanni Santoboni, Federica Cerina, Mario Mureddu, Luca Secchi, and Alessandro Chessa. 'Community core detection in transportation networks'. In: *Physical Review E* 88.4 (2013), page 042810 (cited on page 11).

[12] Muhammad Aqib Javed, Muhammad Shahzad Younis, Siddique Latif, Junaid Qadir, and Adeel Baig. 'Community detection in networks'. In: *Journal of Network and Computer Applications* 108 (2018), pages 87–111 (cited on pages 11, 19, 90).

[13] Krzysztof Nowicki and Tom A. B. Snijders. 'Estimation and prediction for stochastic blockstructures'. In: *Journal of the American Statistical Association* 96.455 (2001), pages 1077–1087 (cited on pages 11, 20).

[14] Edoardo M. Airoldi, David M. Blei, Stephen E. Fienberg, and Eric P. Xing. 'Mixed Membership Stochastic Blockmodels'. In: *Journal of Machine Learning Research* 9.65 (2008), pages 1981–2014 (cited on pages 11, 20, 21).

[15] Jaewon Yang and Jure Leskovec. 'Community-affiliation graph model for overlapping network community detection'. In: *Proceedings of the 12th IEEE International Conference on Data Mining*. (ICDM'12 in Brussels, Belgium). Edited by Mohammed J. Zaki, Arno Siebes, Jeffrey Xu Yu, Bart Goethals, Geoff Webb, and Xindong Wu. Los Alamitos: IEEE Computer Society, 2012, pages 1170–1175 (cited on pages 11, 20, 22).

[16] Stephan Günnemann, Ines Färber, Brigitte Boden, and Thomas Seidl. 'GAMer: a synthesis of subspace clustering and dense subgraph mining'. In: *Knowledge and Information Systems* 40.2 (2014), pages 243–278 (cited on pages 11, 20).

[17] Brigitte Boden, Stephan Günnemann, Holger Hoffmann, and Thomas Seidl. 'Mining coherent subgraphs in multi-layer graphs with edge labels'. In: *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. (KDD'12 in Beijing, China). Edited by Qiang Yang, Deepak Agarwal, and Jian Pei. New York: Association for Computing Machinery, 2012, pages 1258–1266 (cited on pages 11, 20).

[18] Deepayan Chakrabarti, Spiros Papadimitriou, Dharmendra S. Modha, and Christos Faloutsos. 'Fully Automatic Cross-Associations'. In: *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. (KDD'04 in Seattle, WA, USA). Edited by Won Kim, Ronny Kohavi, Johannes Gehrke, and William DuMouchel. New York: Association for Computing Machinery, 2004, pages 79–88 (cited on pages 11, 20).

[19] P. Erdős and A. Rényi. 'On random graphs I'. In: *Publicationes Mathematicae Debrecen* 6 (1959), pages 290–297 (cited on pages 11, 23).

[20] E. N. Gilbert. 'Random Graphs'. In: *Annals of Mathematical Statistics* 30.4 (1959), pages 1141–1144 (cited on page 11).

[21] Duncan J. Watts and Steven H. Strogatz. 'Collective dynamics of 'small-world' networks'. In: *Nature* 393.6684 (1998), pages 440–442 (cited on pages 12, 45, 47).

[22] Réka Albert and Albert-László Barabási. 'Statistical mechanics of complex networks'. In: *Reviews of Modern Physics* 74.1 (2002), pages 47–97 (cited on pages 12, 23, 45).

[23] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 'Graphs over Time: Densification Laws, Shrinking Diameters and Possible Explanations'. In: *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. (KDD'05, Chicago, IL, USA). Edited by Robert Grossman, Roberto Bayardo, and Kristin Bennett. New York: Association for Computing Machinery, 2005, pages 177–187 (cited on pages 12, 23, 74, 85, 86).

[24] Saskia Metzler and Pauli Miettinen. 'Join Size Estimation on Boolean Tensors of RDF Data'. In: *Proceedings of the 24th International Conference on World Wide Web*. (WWW'15 in Florence, Italy). Edited by Aldo Gangemi, Stefano Leonardi, and Alessandro Panconesi. WWW'15 Companion. New York: Association for Computing Machinery, 2015, pages 77–78 (cited on page 13).

[25] Barry Wellman, Janet Salaff, Dimitrina Dimitrova, Laura Garton, Milena Gulia, and Caroline Haythornthwaite. 'Computer Networks as Social Networks: Collaborative Work, Telework, and Virtual Community'. In: *Annual Review of Sociology* 22.1 (1996), pages 213–238 (cited on pages 19, 73).

[26] E. Ravasz, A. L. Somera, D. A. Mongru, Z. N. Oltvai, and A.-L. Barabási. 'Hierarchical Organization of Modularity in Metabolic Networks'. In: *Science* 297.5586 (2002), pages 1551–1555 (cited on page 19).

[27] Sergei Maslov and Kim Sneppen. 'Specificity and Stability in Topology of Protein Networks'. In: *Science* 296.5569 (2002), pages 910–913 (cited on page 19).

[28] Shai S. Shen-Orr, Ron Milo, Shmoolik Mangan, and Uri Alon. 'Network motifs in the transcriptional regulation network of *Escherichia coli*'. In: *Nature Genetics* 31.1 (2002), pages 64–68 (cited on page 19).

[29] Ann E. Krause, Kenneth A. Frank, Doran M. Mason, Robert E. Ulanowicz, and William W. Taylor. 'Compartments revealed in food-web structure'. In: *Nature* 426.6964 (2003), pages 282–285 (cited on page 19).

[30] R. Guimerà, D. B. Stouffer, M. Sales-Pardo, E. A. Leicht, M. E.J. Newman, and L. A.N. Amaral. 'Origin of compartmentalization in food webs'. English (US). In: *Ecology* 91.10 (2010), pages 2941–2951 (cited on page 19).

[31] Hocine Cherifi, Gergely Palla, Boleslaw K. Szymanski, and Xiaoyan Lu. 'On community structure in complex networks: challenges and opportunities'. In: *Applied Network Science* 4.117 (2019), pages 1–35 (cited on page 20).

[32] Pauli Miettinen, Taneli Mielikäinen, Aristides Gionis, Gautam Das, and Heikki Mannila. 'The Discrete Basis Problem'. In: *IEEE Transactions on Knowledge and Data Engineering* 20.10 (2008), pages 1348–1362 (cited on pages 20, 57, 98, 103, 128).

[33] Edward O. Laumann and Franz U. Pappi. *Networks of Collective Action: A Perspective on Community Influence Systems*. Edited by Peter H. Rossi. Quantitative Studies in Social Relations. New York: Academic Press, 1976 (cited on pages 20, 74).

[34] Richard D. Alba and Gwen Moore. 'Elite Social Circles'. In: *Sociological Methods & Research* 7.2 (1978), pages 167–188 (cited on pages 20, 74).

[35] David L. Morgan, Margaret B. Neal, and Paula Carder. 'The stability of core and peripheral networks over time'. In: *Social Networks* 19.1 (1997), pages 9–25 (cited on pages 20, 74).

[36] Paul B. Reed and L. Kevin Selbee. 'The Civic Core in Canada: Disproportionality in Charitable Giving, Volunteering, and Civic Participation'. In: *Nonprofit and Voluntary Sector Quarterly* 30.4 (2001), pages 761–780 (cited on pages 20, 73, 74).

[37] Pietro Panzarasa, Tore Opsahl, and Kathleen M. Carley. 'Patterns and dynamics of users' behavior and interaction: Network analysis of an online community'. In: *Journal of the Association for Information Science and Technology* 60.5 (2009), pages 911–932 (cited on pages 20, 74).

[38] C. Corradino. 'Proximity structure in a captive colony of Japanese monkeys (Macaca fuscata fuscata): An application of multidimensional scaling'. In: *Primates* 31.3 (1990), pages 351–362 (cited on pages 20, 74).

[39] M. Puck Rombach, Mason A. Porter, James H. Fowler, and Peter J. Mucha. 'Core-Periphery Structure in Networks'. In: *SIAM Journal on Applied Mathematics* 74.1 (2014), pages 167–190 (cited on page 20).

[40] Sanjar Karaev, Saskia Metzler, and Pauli Miettinen. 'Logistic-Tropical Decompositions and Nested Subgraphs'. In: *Proceedings of the 14th International Workshop on Mining and Learning with Graphs*. (MLG@KDD'18 in London, UK). London: http://www.mlgworkshop.org/2018, 2018, pages 1–8 (cited on pages 21, 50, 90, 91, 133, 134).

[41] Stefan Neumann, Rainer Gemulla, and Pauli Miettinen. 'What You Will Gain By Rounding: Theory and Algorithms for Rounding Rank'. In: *Proceedings of the 16th IEEE International Conference on Data Mining*. (ICDM'16 in Barcelona, Spain). Edited by Francesco Bonchi, Josep Domingo-Ferrer, Ricardo A. Baeza-Yates, Zhi-Hua Zhou, and Xindong Wu. Los Alamitos: IEEE Computer Society, 2016, pages 380–389 (cited on pages 21, 90, 134).

[42] Paul W. Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. 'Stochastic blockmodels: First steps'. In: *Social Networks* 5.2 (1983), pages 109–137 (cited on pages 21, 23).

[43] Clement Lee and Darren J. Wilkinson. 'A review of stochastic block models and extensions for graph clustering'. In: *Applied Network Science* 4.1 (2019), page 122 (cited on page 21).

[44] Tiago P. Peixoto. 'Model Selection and Hypothesis Testing for Large-Scale Network Models with Overlapping Groups'. In: *Physical Review X* 5 (1 2015), page 011033 (cited on page 21).

[45] Tiago P. Peixoto. 'Nonparametric Bayesian inference of the microcanonical stochastic block model'. In: *Physical Review E* 95 (1 2017), page 012317 (cited on pages 22, 90).

[46] Brian Karrer and M. E. J. Newman. 'Stochastic blockmodels and community structure in networks'. In: *Physical Review E* 83.1 (1 2011), page 016107 (cited on pages 22, 23).

[47] Gergely Palla, Imre Derényi, Illés Farkas, and Tamás Vicsek. 'Uncovering the overlapping community structure of complex networks in nature and society'. In: *Nature* 435.7043 (2005), pages 814–818 (cited on page 22).

[48] Andrea Lancichinetti, Santo Fortunato, and János Kertész. 'Detecting the overlapping and hierarchical community structure in complex networks'. In: *New Journal of Physics* 11.3 (2009), page 033015 (cited on page 22).

[49] Yongsub Lim, U Kang, and Christos Faloutsos. 'SlashBurn: Graph Compression and Mining beyond Caveman Communities'. In: *IEEE Transactions on Knowledge and Data Engineering* 26.12 (2014), pages 3077–3089 (cited on page 22).

[50] Danai Koutra, U Kang, Jilles Vreeken, and Christos Faloutsos. 'Summarizing and understanding large graphs'. In: *Statistical Analysis and Data Mining* 8.3 (2015), pages 183–202 (cited on page 23).

[51] Himchan Park and Min-Soo Kim. 'EvoGraph: An Effective and Efficient Graph Upscaling Method for Preserving Graph Properties'. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. (KDD'18 in London, UK). Edited by Yike Guo and Faisal Farooq. New York: Association for Computing Machinery, 2018, pages 2051–2059 (cited on page 23).

[52] J. W. Zhang and Y. C. Tay. 'GSCALER: Synthetically Scaling A Given Graph'. In: *Proceedings of the 19th International Conference on Extending Database Technology*. (EDBT'16 in Bordeaux, France). Edited by Evaggelia Pitoura, Sofian Maabout, Georgia Koutrika, Amélie Marian, Letizia Tanca, Ioana Manolescu, and Kostas Stefanidis. Konstanz, Germany: OpenProceedings.org, 2016, pages 53–64 (cited on page 23).

[53] Dmitri Krioukov, Fragkiskos Papadopoulos, Maksim Kitsak, Amin Vahdat, and Marián Boguñá. 'Hyperbolic geometry of complex networks'. In: *Physical Review E* 82.3 (3 2010), page 036106 (cited on page 23).

[54] Jure Leskovec, Deepayan Chakrabarti, Jon Kleinberg, Christos Faloutsos, and Zoubin Ghahramani. 'Kronecker Graphs: An Approach to Modeling Networks'. In: *Journal of Machine Learning Research* 11 (2010), pages 985–1042 (cited on page 23).

[55] Emmanuel Abbe. 'Community Detection and Stochastic Block Models: Recent Developments'. In: *Journal of Machine Learning Research* 18.177 (2018), pages 6446–6531 (cited on page 23).

[56] Yaojia Zhu, Xiaoran Yan, and Cristopher Moore. 'Oriented and degree-generated block models: generating and inferring communities with inhomogeneous degree distributions'. In: *Journal of Complex Networks* 2.1 (2014), pages 1–18 (cited on page 23).

[57] Deepayan Chakrabarti, Yiping Zhan, and Christos Faloutsos. 'R-MAT: A Recursive Model for Graph Mining'. In: *Proceedings of the 4th SIAM International Conference on Data Mining*. (SDM'04 in Lake Buena Vista, FL, USA). Edited by Michael W. Berry, Umeshwar Dayal, Chandrika Kamath, and David B. Skillicorn. Philadelphia: Society for Industrial and Applied Mathematics, 2004, pages 442–446 (cited on page 23).

[58] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. 'Benchmark graphs for testing community detection algorithms'. In: *Physical Review E* 78.4 (2008), page 046110 (cited on pages 23, 42, 64).

[59] Günce Keziban Orman, Vincent Labatut, and Hocine Cherifi. 'Towards realistic artificial benchmark for community detection algorithms evaluation'. In: *International Journal of Web Based Communities* 9.3 (2013), page 349 (cited on page 24).

[60] Justin Fagnan, Afra Abnar, Reihaneh Rabbany, and Osmar R. Zaïane. 'Modular Networks for Validating Community Detection Algorithms'. In: *Computing Research Repository* 1801.01229 (2018), pages 1–20 (cited on page 24).

[61] Zhao Yang, Juan I. Perotti, and Claudio J. Tessone. 'Hierarchical benchmark graphs for testing community detection algorithms'. In: *Physical Review E* 96.5 (2017), page 052311 (cited on page 24).

[62] Peter Orbanz and Daniel M. Roy. 'Bayesian Models of Graphs, Arrays and Other Exchangeable Random Structures'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.2 (2015), pages 437–461 (cited on pages 31, 43).

[63] Michael R. Garey and David S. Johnson. *Computers and intractability: A guide to the theory of NP-Completeness*. San Francisco: W. H. Freeman & Co., 1, 1979. 340 pages (cited on page 35).

[64] Jeffrey Pattillo, Alexander Veremyev, Sergiy Butenko, and Vladimir Boginski. 'On the maximum quasi-clique problem'. In: *Discrete Applied Mathematics* 161.1-2 (2013), pages 244–257 (cited on page 35).

[65] Pauli Miettinen. 'Generalized Matrix Factorizations as a Unifying Framework for Pattern Set Mining: Complexity Beyond Blocks'. In: *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases*. (ECML PKDD'15 in Porto, Portugal). Edited by Annalisa Appice, Pedro Pereira Rodrigues, Vitor Santos Costa, João Gama, Alípio Jorge, and Carlos Soares. Volume 9285. Lecture Notes in Computer Science. Cham, Switzerland: Springer International Publishing, 2015, pages 36–52 (cited on page 36).

[66] Donald E. Knuth. *The Art of Computer Programming Volume 2: Seminumerical Algorithms*. Third Edition. Reading, MA: Addison-Wesley, 1, 1997. 784 pages (cited on page 43).

[67] Charu C. Aggarwal and Haixun Wang, editors. *Managing and Mining Graph Data*. Volume 40. Springer Texts in Statistics. New York: Springer Science and Business Media, 2010 (cited on pages 44, 47, 50).

[68] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. 'On power-law relationships of the Internet topology'. In: *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*. (SIGCOMM'99 in Cambridge, MA, USA). Edited by Lyman Chapin, James P. G. Sterbenz, Guru M. Parulkar, and Jonathan S. Turner. Volume 29. 4. New York: Association for Computing Machinery, 1999, pages 251–262 (cited on page 45).

[69] Samuel Jonhson, Virginia Domínguez-García, and Miguel A. Muñoz. 'Factors Determining Nestedness in Complex Networks'. In: *PLoS ONE* 8.9 (2013). Edited by Yamir Moreno, e74025 (cited on page 50).

[70] Jure Leskovec and Andrej Krevl. *SNAP Datasets: Stanford Large Network Dataset Collection*. 2014. URL: http://snap.stanford.edu/data (visited on 02/11/2016) (cited on pages 51, 59).

[71] Timothy A. Davis and Yifan Hu. 'The University of Florida Sparse Matrix Collection'. In: *ACM Transactions on Mathematical Software* 38.1 (2011), pages 1–25 (cited on page 52).

[72] Larry Wasserman. *All of Statistics: A Concise Course in Statistical Inference*. Edited by George Casella, Stephen Fienberg, and Ingram Olkin. New York: Springer Science and Business Media, 2004 (cited on pages 54, 78, 82).

[73] Ulrike von Luxburg. 'A Tutorial on Spectral Clustering'. In: *Statistics and Computing* 17.4 (2007), pages 395–416 (cited on page 56).

[74] Stack Exchange, Inc. *Stack Exchange Data Dump*. 2016. URL: https://archive.org/details/stackexchange (visited on 01/24/2017) (cited on pages 59, 76).

[75] Tamas Nepusz. *Blockmodel: Fitting stochastic blockmodels to empirical networks*. 2015. URL: https://github.com/ntamas/blockmodel (visited on 03/04/2019) (cited on page 66).

[76] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Hoboken, NJ: John Wiley & Sons, 2006 (cited on page 68).

[77] Nancy K. Baym, Yan Bing Zhang, and Mei-Chen Lin. 'Social Interactions Across Media: Interpersonal Communication on the Internet, Telephone and Face-to-Face'. In: *New Media & Society* 6.3 (2004), pages 299–318 (cited on page 73).

[78] Valerio Arnaboldi, Andrea Guazzini, and Andrea Passarella. 'Egocentric online social networks: Analysis of key features and prediction of tie strength in Facebook'. In: *Computer Communications* 36.10-11 (2013), pages 1130–1144 (cited on pages 73, 86).

[79] R. I. M. Dunbar, Valerio Arnaboldi, Marco Conti, and Andrea Passarella. 'The structure of online social networks mirrors those in the offline world'. In: *Social Networks* 43 (2015), pages 39–47 (cited on pages 73, 86).

[80] Rebecca Nesbit and Beth Gazley. 'Patterns of Volunteer Activity in Professional Associations and Societies'. In: *Voluntas* 23.3 (2012), pages 558–583 (cited on pages 73, 74, 86).

[81] Axel Bruns. *Blogs, Wikipedia, Second Life, and Beyond: From Production to Produsage*. New York: Peter Lang, 2008 (cited on page 73).

[82] Eszter Hargittai and Gina Walejko. 'The Participation Divide: Content creation in the digital age'. In: *Information, Communication & Society* 11.2 (2008), pages 239–256 (cited on page 73).

[83] Sorin Adam Matei and Robert J. Bruno. 'Pareto's 80/20 law and social differentiation: A social entropy perspective'. In: *Public Relations Review* 41.2 (2015), pages 178–186 (cited on page 74).

[84] Felipe Ortega, Jesus M. Gonzalez-Barahona, and Gregorio Robles. 'On the Inequality of Contributions to Wikipedia'. In: *Proceedings of the 41st Annual Hawaii International Conference on System Sciences*. (HICSS'08 in Waikoloa, HI, USA). Edited by Ralph H. Sprague Jr. Los Alimatos: IEEE Computer Society, 2008, pages 304–310 (cited on page 74).

[85] Aniket Kittur and Robert E. Kraut. 'Harnessing the Wisdom of Crowds in Wikipedia: Quality Through Coordination'. In: *Proceedings of the ACM Conference on Computer Supported Cooperative Work*. (CSCW'08 in San Diego, CA, USA). Edited by Bo Begole and David W. McDonald. New York: Association for Computing Machinery, 2008, pages 37–46 (cited on page 74).

[86] Mordechai Haklay. 'Why is participation inequality important?' In: *European Handbook of Crowdsourced Geographic Information*. Edited by C. Capineri, M. Haklay, H. Huang, V. Antoniou, J. Kettunen, F. Ostermann, and R. Purves. London: Ubiquity Press, 2016, pages 35–44 (cited on page 74).

[87] Brian Butler, Lee Sproull, Sara Kiesler, and Robert Kraut. 'Community Effort in Online Groups: Who Does the Work and Why?' In: *Leadership at a Distance: Research in Technically-Supported Work*. Edited by Suzanne P. Weisband. New York: Psychology Press, 2007. Chapter 9, pages 171–194 (cited on pages 74, 86).

[88] Jure Leskovec, Kevin J Lang, Anirban Dasgupta, and Michael W Mahoney. 'Statistical properties of community structure in large social and information networks'. In: *Proceedings of the 17th international conference on World Wide Web*. (WWW'08 in Beijing, China). Edited by Jinpeng Huai, Robin Chen, Hsiao-Wuen Hon, Yunhao Liu, Wei-Ying Ma, Andrew Tomkins, and Xiaodong Zhang. New York: Association for Computing Machinery, 2008, pages 695–704 (cited on page 74).

[89] Ciro Cattuto. 'Semiotic dynamics in online social communities'. In: *European Physical Journal C* 37 (2006), pages 33–37 (cited on page 74).

[90] Yong-Yeol Ahn, Seungyeop Han, Haewoon Kwak, Sue Moon, and Hawoong Jeong. 'Analysis of topological characteristics of huge online social networking services'. In: *Proceedings of the 16th international conference on World Wide Web*. (WWW'07 in Banff, AB, Canada). Edited by Carey Williamson, Mary Ellen Zurko, Peter Patel-Schneider, and Prashant Shenoy. New York: Association for Computing Machinery, 2007, pages 835–844 (cited on page 74).

[91] M. E. J. Newman and Juyong Park. 'Why social networks are different from other types of networks'. In: *Physical Review E* 68.3 (2003), page 036122 (cited on page 74).

[92] Vedran Sekara, Arkadiusz Stopczynski, and Sune Lehmann. 'Fundamental structures of dynamic social networks'. In: *Proceedings of the National Academy of Sciences* 113.36 (2016), pages 9977–9982 (cited on page 74).

[93] Tianyang Zhang, Peng Cui, Christos Faloutsos, Yunfei Lu, Hao Ye, Wenwu Zhu, and Shiqiang Yang. 'comeNgo: A Dynamic Model for Social Group Evolution'. In: *ACM Transactions on Knowledge Discovery from Data* 11.4 (2017), pages 1–22 (cited on page 74).

[94] Ravi Kumar, Jasmine Novak, and Andrew Tomkins. 'Structure and evolution of online social networks'. In: *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. (KDD'06 Philadelphia, PA, USA). Edited by Lyle Ungar, Mark Craven, Dimitrios Gunopulos, and Tina Eliassi-Rad. New York: Association for Computing Machinery, 2006, pages 611–617 (cited on pages 74, 85, 86).

[95] Richard Koch. *The 80/20 Principle: The Secret of Achieving More with Less*. Second Edition. New York: Nicholas Brealey Publishing, 2011 (cited on pages 75, 86).

[96] Mathieu Bastian, Sebastien Heymann, and Mathieu Jacomy. 'Gephi: An Open Source Software for Exploring and Manipulating Networks'. In: *Proceedings of the Third International AAAI Conference on Weblogs and Social Media*. (ICWSM'09 in San Jose, CA, USA). Edited by Eytan Adar, Matthew Hurst, Tim Finin, Natalie Glance, Nicolas Nicolov, and Belle Tseng. Menlo Park, CA: AAAI Press, 2009, pages 361–362 (cited on page 76).

[97] Gonzalo Fernández-Sánchez and Fernando Rodríguez-López. 'A methodology to identify sustainability indicators in construction project management—Application to infrastructure projects in Spain'. In: *Ecological Indicators* 10.6 (2010), pages 1193–1201 (cited on page 86).

[98] Herman E. Daly and Joshua Farley. *Ecological economics: principles and applications*. Second Edition. Washington: Island press, 2011 (cited on page 86).

[99] Mark D. Jankowski, Christopher J. Williams, Jeanne M. Fair, and Jennifer C. Owen. 'Birds Shed RNA-Viruses According to the Pareto Principle'. In: *PLoS ONE* 8.8 (2013). Edited by Xiaofeng Ren, e72611 (cited on page 86).

[100] Gergely Palla, Albert-László Barabási, and Tamás Vicsek. 'Quantifying social group evolution'. In: *Nature* 446.7136 (2007), pages 664–667 (cited on page 86).

[101] Sam G. B. Roberts, Robin I. M. Dunbar, Thomas V. Pollet, and Toon Kuppens. 'Exploring variation in active network size: Constraints and ego characteristics'. In: *Social Networks* 31.2 (2009), pages 138–146 (cited on page 86).

[102] R. I. M. Dunbar. 'Neocortex size as a constraint on group size in primates'. In: *Journal of Human Evolution* 22.6 (1992), pages 469–493 (cited on page 86).

[103] Andrea De Salve, Marco Dondio, Barbara Guidi, and Laura Ricci. 'The impact of user's availability on On-line Ego Networks: A Facebook analysis'. In: *Computer Communications* 73 (2016), pages 211–218 (cited on page 86).

[104] J. Saramäki, E. A. Leicht, E. López, S. G. B. Roberts, F. Reed-Tsochas, and R. I. M. Dunbar. 'Persistence of social signatures in human communication'. In: *Proceedings of the National Academy of Sciences* 111.3 (2014), pages 942–947 (cited on page 86).

[105] Raj Rao Nadakuditi and M. E. J. Newman. 'Graph Spectra and the Detectability of Community Structure in Networks'. In: *Physical Review Letters* 108 (18 2012), page 188701 (cited on page 91).

[106] Aurelien Decelle, Florent Krzakala, Cristopher Moore, and Lenka Zdeborová. 'Inference and Phase Transitions in the Detection of Modules in Sparse Networks'. In: *Physical Review Letters* 107 (6 2011), page 065701 (cited on page 91).

[107] Dóra Erdős and Pauli Miettinen. 'Discovering Facts with Boolean Tensor Tucker Decomposition'. In: *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*. (CIKM'13 in San Francisco, CA, USA). Edited by Qi He, Arun Iyengar, Wolfgang Nejdl, Jian Pei, and Rajeev Rastogi. New York: Association for Computing Machinery, 2013, pages 1569–1572 (cited on pages 95, 98).

[108] Dóra Erdős and Pauli Miettinen. 'Walk'n'Merge: A Scalable Algorithm for Boolean Tensor Factorization'. In: *Proceedings of the 13th IEEE International Conference on Data Mining*. (IDCM'13 in Dallas, TX, USA). Edited by Hui Xiong, George Karypis, Bhavani Thuraisingham, Diane Cook, and Xindong Wu. Los Alamitos: IEEE Computer Society, 2013, pages 1037–1042 (cited on pages 95, 98, 115, 125).

[109] Lizhuang Zhao and Mohammed J. Zaki. 'triCluster: An Effective Algorithm for Mining Coherent Clusters in 3D Microarray Data'. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. (SIGMOD'05 in Baltimore, MD, USA). Edited by Fatma Ozcan. New York: Association for Computing Machinery, 2005, pages 694–705 (cited on pages 97, 105).

[110] Stefanie Jegelka, Suvrit Sra, and Arindam Banerjee. 'Approximation Algorithms for Tensor Clustering'. In: *Proceedings of the International Conference on Algorithmic Learning Theory*. (ALT'09 in Porto, Portugal). Edited by Ricard Gavaldà, Gábor Lugosi, Thomas Zeugmann, and Sandra Zilles. Volume 5809. Lecture Notes in Computer Science. Berlin Heidelberg: Springer, 2009, pages 368–383 (cited on pages 97, 102).

[111] Evangelos E. Papalexakis, Nicholas Sidiropoulos, and Rasmus Bro. 'From *K*-Means to Higher-Way Co-Clustering: Multilinear Decomposition With Sparse Latent Factors'. In: *IEEE Transactions on Signal Processing* 61.2 (2013), pages 493–506 (cited on page 97).

[112] Heng Huang, Chris Ding, Dijun Luo, and Tao Li. 'Simultaneous Tensor Subspace Selection and Clustering: The Equivalence of High Order SVD and *K*-Means Clustering'. In: *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. (KDD'08 in Las Vegas, NV, USA). Edited by Ying Li, Bing Liu, and Sunita Sarawagi. New York: Association for Computing Machinery, 2008, pages 327–335 (cited on pages 97, 102).

[113] Xinhai Liu, Lieven De Lathauwer, Frizo Janssens, and Bart De Moor. 'Hybrid Clustering of Multiple Information Sources via HOSVD'. In: *Proceedings of the 7th International Conference on Advances in Neural Networks - Part II*. (ISNN'10 in Shanghai, China). Edited by Liqing Zhang, Bao-Liang Lu, and James Kwok. Volume 6064. Lecture Notes in Computer Science. Berlin Heidelberg: Springer, 2010, pages 337–345 (cited on page 97).

[114] Ledyard R. Tucker. 'Some mathematical notes on three-mode factor analysis'. In: *Psychometrika* 31.3 (1966), pages 279–311 (cited on page 97).

[115] J. Douglas Carroll and Jih-Jie Chang. 'Analysis of individual differences in multidimensional scaling via an N-way generalization of "Eckart–Young" decomposition'. In: *Psychometrika* 35.3 (1970), pages 283–319 (cited on page 97).

[116] Richard A. Harshman. 'Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multimodal factor analysis'. In: *UCLA Working Papers in Phonetics* 16 (1970), pages 1–84 (cited on page 97).

[117] Tamara G. Kolda and Brett W. Bader. 'Tensor decompositions and applications'. In: *SIAM Review* 51.3 (2009), pages 455–500 (cited on pages 97, 99, 101).

[118] Evangelos E. Papalexakis, Christos Faloutsos, and Nicholas D. Sidiropoulos. 'ParCube: Sparse Parallelizable Tensor Decompositions'. In: *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases*. (ECML PKDD'12 in Bristol, UK). Edited by Peter A. Flach, Tijl De Bie, and Nello Cristianini. Volume 7523. Lecture Notes in Computer Science. Berlin Heidelberg: Springer, 2012, pages 521–536 (cited on pages 97, 116).

[119] Iwin Leenen, Iven Van Mechelen, Paul De Boeck, and Seymour Rosenberg. 'Indclas: A three-way hierarchical classes model'. In: *Psychometrika* 64.1 (1999), pages 9–24 (cited on page 97).

[120] Radim Bělohlávek, Cynthia Glodeanu, and Vilém Vychodil. 'Optimal Factorization of Three-Way Binary Data Using Triadic Concepts'. In: *Order. A Journal on the Theory of Ordered Sets and its Applications* 30.2 (2012), pages 437–454 (cited on page 97).

[121] Dmitry I. Ignatov, Sergei O. Kuznetsov, Ruslan A. Magizov, and Leonid E. Zhukov. 'From Triconcepts to Triclusters'. In: *Proceedings of the 13th International Conference on Rough Sets, Fuzzy Sets, Data Mining and Granular Computing*. (RSFDGrC'11 in Moscow, Russia). Edited by Sergei O. Kuznetsov, Dominik Ślęzak, Daryl H. Hepting, and Boris G. Mirkin. Volume 6743. Lecture Notes in Computer Science. Berlin Heidelberg: Springer, 2011, pages 257–264 (cited on page 97).

[122] Pauli Miettinen. 'Boolean Tensor Factorizations'. In: *Proceedings of the 11th IEEE International Conference on Data Mining*. (ICDM'11 in Vancouver, BC, Canada). Edited by Diane Cook, Jian Pei, Wei Wang, Osmar Zaïane, and Xindong Wu. Los Alamitos: IEEE Computer Society, 2011, pages 447–456 (cited on pages 98, 99, 101, 115, 123).

[123] Loïc Cerf, Jérémy Besson, Kim-Ngan T. Nguyen, and Jean-François Boulicaut. 'Closed and noise-tolerant patterns in n-ary relations'. In: *Data Mining and Knowledge Discovery* 26.3 (2013), pages 574–619 (cited on page 98).

[124] Loïc Cerf, Jérémy Besson, Céline Robardet, and Jean-François Boulicaut. 'Closed patterns meet n-ary relations'. In: *ACM Transactions on Knowledge Discovery from Data* 3.1 (2009), pages 1–36 (cited on page 98).

[125] Peng Jiang. 'Pattern extraction and clustering for high-dimensional discrete data'. PhD thesis. University of Illinois at Urbana-Champaign, 2014 (cited on pages 98, 103, 106).

[126] Jon M. Kleinberg, Christos H. Papadimitriou, and Prabhakar Raghavan. 'A Microeconomic View of Data Mining'. In: *Data Mining and Knowledge Discovery* 2.4 (1998), pages 311–324 (cited on page 98).

[127] Jon M. Kleinberg, Christos H. Papadimitriou, and Prabhakar Raghavan. 'Segmentation problems'. In: *Journal of the ACM* 51.2 (2004), pages 263–280 (cited on pages 98, 105, 107, 108, 110).

[128] Jouni K. Seppänen. 'Upper bound for the approximation ratio of a class of hypercube segmentation algorithms'. In: *Information Processing Letters* 93.3 (2005), pages 139–141 (cited on page 98).

[129] Noga Alon and Benny Sudakov. 'On two segmentation problems'. In: *Journal of Algorithms* 33.1 (1999), pages 173–184 (cited on pages 98, 106–108, 110).

[130] Mijung Kim and K. Selçuk Candan. 'Approximate tensor decomposition within a tensor-relational algebraic framework'. In: *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*. (CIKM'11 in Glasgow, UK). Edited by Bettina Berendt, Arjen de Vries, Wenfei Fan, Craig Macdonald, Iadh Ounis, and Ian Ruthven. New York: Association for Computing Machinery, 2011, pages 1737–1742 (cited on page 98).

[131] Mijung Kim and K. Selçuk Candan. 'Decomposition-by-Normalization (DBN): Leveraging Approximate Functional Dependencies for Efficient Tensor Decomposition'. In: *Proceedings of the 21st ACM International Conference on Information & Knowledge Management*. (CIKM'12 in Maui, HI, USA). Edited by Xuewen Chen, Guy Lebanon, Haixun Wang, and Mohammed J. Zaki. New York: Association for Computing Machinery, 2012, pages 355–364 (cited on page 98).

[132] Mijung Kim and K. Selçuk Candan. 'Pushing-Down Tensor Decompositions over Unions to Promote Reuse of Materialized Decompositions'. In: *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases*. (ECML PKDD'14 in Nancy, France). Edited by Toon Calders, Floriana Esposito, Eyke Hüllermeier, and Rosa Meo. Volume 8724. Lecture Notes in Computer Science. Berlin Heidelberg: Springer, 2014, pages 688–704 (cited on page 98).

[133] Pauli Miettinen. 'Matrix Decomposition Methods for Data Mining: Computational Complexity and Algorithms'. PhD thesis. Department of Computer Science, University of Helsinki, 2009 (cited on page 104).

[134] Leonardo Dagum and Ramesh Menon. 'OpenMP: an industry standard API for shared-memory programming'. In: *IEEE Computational Science and Engineering Magazine* 5.1 (1998), pages 46–55 (cited on page 112).

[135] Jorma Rissanen. 'Modeling by shortest data description'. In: *Automatica* 14.5 (1978), pages 465–471 (cited on page 112).

[136] Pauli Miettinen and Jilles Vreeken. 'MDL4BMF: Minimum Description Length for Boolean Matrix Factorization'. In: *ACM Transactions on Knowledge Discovery from Data* 8.4 (2014), pages 1–31 (cited on pages 112, 113).

[137] N. K. Vereshchagin and P. M. B. Vitanyi. 'Kolmogorov's structure functions and model selection'. In: *IEEE Transactions on Information Theory* 50.12 (2004), pages 3265–3290 (cited on page 113).

[138] Eric C. Chi and Tamara G. Kolda. 'On Tensors, Sparsity, and Nonnegative Factorizations'. In: *SIAM Journal on Matrix Analysis and Applications* 33.4 (2012), pages 1272–1299 (cited on page 115).

[139] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Upper Saddle River, NJ: Prentice-Hall, 1982 (cited on page 121).

[140] Iván Cantador, Peter Brusilovsky, and Tsvi Kuflik. 'Second Workshop on Information Heterogeneity and Fusion in Recommender Systems'. In: *Proceedings of the 5th ACM Conference on Recommender Systems (RecSys)*. (HetRec'11 in Chicago, IL, USA). Edited by Bamshad Mobasher, Robin Burke, Dietmar Jannach, and Gediminas Adomavicius. New York: Association for Computing Machinery, 2011, pages 387–388 (cited on page 123).

[141] Bryan Klimt and Yiming Yang. 'The Enron Corpus: A New Dataset for Email Classification Research'. In: *Proceedings of the 15th European Conference on Machine Learning*. Edited by Jean-François Boulicaut, Floriana Esposito, Fosca Giannotti, and Dino Pedreschi. Volume 3201. Lecture Notes in Computer Science. Berlin Heidelberg: Springer, 2004, pages 217–226 (cited on page 123).

[142] Bimal Viswanath, Alan Mislove, Meeyoung Cha, and Krishna P. Gummadi. 'On the Evolution of User Interaction in Facebook'. In: *Proceedings of the 2nd ACM Workshop on Online Social Networks*. (WOSN'09 in Barcelona, Spain). Edited by Jon Crowcroft and Balachander Krishnamurthy. New York: Association for Computing Machinery, 2009, pages 37–42 (cited on page 123).

[143] Alexander Yates and Oren Etzioni. 'Unsupervised methods for determining object and relation synonyms on the web'. In: *Journal of Artificial Intelligence Research* 34 (2009), pages 255–296 (cited on page 123).

[144] Center for Applied Internet Data Analysis. *The CAIDA UCSD Anonymized Internet Traces*. 2009. URL: https://www.caida.org/data/passive/passive_2009_dataset.xml (visited on 10/03/2020) (cited on page 123).

[145] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 'Yago: A Core of Semantic Knowledge'. In: *Proceedings of the 16th International Conference on World Wide Web*. (WWW'07 in Banff, AB, Canada). Edited by Carey Williamson, Mary Ellen Zurko, Peter Patel-Schneider, and Prashant Shenoy. New York: Association for Computing Machinery, 2007, pages 697–706 (cited on page 123).

[146] Pauli Miettinen. 'Sparse Boolean Matrix Factorizations'. In: *Proceedings of the 10th IEEE International Conference on Data Mining*. (ICDM'10 in Sydney, NSW, Australia). Edited by Geoffrey I. Webb, Bing Liu, Chengqi Zhang, Dimitrios Gunopulos, and Xindong Wu. Los Alamitos: IEEE Computer Society, 2010, pages 935–940 (cited on page 125).

List of Abbreviations

BCPC Boolean CP clustering. 96, 103, 104, 105, 106, 107, 108, 111, 113, 116, 131, 132, 133
BIC Bayesian information criterion. 64, 152
BMF Boolean matrix factorisation. 19, 55, 56, 57, 112, 113, 143
BTC Boolean tensor clustering. 13, 96, 99, 102, 103, 131, 132, 133, 181

CANDECOMP canonical decomposition. 181
CP CANDECOMP/PARAFAC. 96, 97, 99, 100, 101, 102, 103, 115, 116, 123, 125, 129, 131, 181
CPU central processing unit. 54, 118

DBPP discrete basis partitioning problem. 98, 102, 103, 105, 106
DC-SBM degree-corrected SBM. 21, 23, 59, 60, 61, 62, 63, 66, 67, 68, 90
DtM data-to-model. 113

GEV generalised extreme value. 64, 65

HB HealthBoards. 76, 80

IP Internet protocol. 123

LFR Lancichinetti–Fortunato–Radicchi. 23, 24, 59, 60, 61, 62, 63, 66, 67, 68
LL log-likelihood. 79

MDL minimum description length. 21, 90, 112, 113, 122, 127, 128, 129
MSE mean square error. 152, 153

PARAFAC parallel factors. 181
PM Pauli Miettinen. 4, 5, 6
PTAS polynomial-time approximation scheme. 6, 97, 98, 105, 107, 108, 110, 131

R-MAT recursive matrix. 23, 59, 60, 61, 62, 63
RDF resource description framework. 13
RMSE root mean square error. 151, 154

SBM stochastic block model. 21, 23, 42, 44, 59, 60, 61, 62, 61, 91, 181
SD standard deviation. 61, 67, 70, 118, 119, 151, 153, 154
SE StackExchange. 76, 80
SG Stephan Günnemann. 4, 5
SM Saskia Metzler. 4, 5, 6
SSE sum of squares error. 152

U-BTC unconstrained BTC. 102, 103, 104, 105, 107, 108, 119, 132
URL uniform resource locator. 7, 127