Data Mining in Complex Networks: Missing Link Prediction and Fuzzy Communities

A dissertation submitted for the degree of Doctor of Philosophy

Tamás Nepusz

Department of Measurement and Information Systems,
Budapest University of Technology and Economics,
Budapest, Hungary

Advisors:

Dr. Fülöp Bazsó, Ph.D.,
KFKI Research Institute for Particle and Nuclear Physics,
Hungarian Academy of Sciences, Budapest, Hungary

Dr. György Strausz, Ph.D.,
Department of Measurement and Information Systems,
Budapest University of Technology and Economics, Budapest, Hungary

Ph.D. School: Informatics
Head: Dr. Endre Selényi
Ph.D. Program: Intelligent systems

Budapest, 2008

Abstract

This dissertation is devoted to networks: complex interconnected systems where the individual components are connected by binary links arranged in seemingly random but intrinsically structured patterns. Networks are used to model various real-world phenomena ranging from protein interaction in living organisms to the large-scale organisation of human society or the structure of technological networks such as software systems or the Internet.

The first part of the dissertation studies a stochastic graph model, which can be considered as a possible extension of Erdős–Rényi random graphs. I discuss some basic statistical properties of the model and devise methods to find the best fit of the model to a given network instance. I also demonstrate how the fitted model can be used to predict previously unknown connections in the network.

The second part of the dissertation studies overlapping communities (i.e., dense subgraphs) in sparse networks. I introduce a method based on the concept of fuzzy partition matrices and vertex similarity to uncover meaningful communities with possible overlaps and to identify bridge vertices that belong to more than one community significantly. Finally, I present applications of the link prediction and community detection methods on real-world datasets.

Kivonat

Disszertációm témája a hálózatok világa. Az általam tanulmányozott hálózatok egyedi elemekből bináris relációk által képzett összetett rendszerek, ahol a kapcsolatok véletlenszerűnek tűnő mintázata mögött rendszerint belső szabályszerűség rejlik. A hálózatok elméletét számos valós jelenség modellezésére használták már az élő szervezetek fehérjéi közti interakcióktól kezdve az emberi társadalom nagyléptékű szerveződéséig, vagy éppen olyan, ember alkotta technológiai hálózatokig, mint a szoftverrendszerek vagy az Internet.

A disszertáció első felében egy sztochasztikus gráfmodellt vizsgálok, amely az Erdős–Rényi véletlen gráfok egy lehetséges kiterjesztésének tekinthető. A modell alapvető statisztikai jellemzése után két módszert ismertetek a modell adott hálózatra való legjobb illesztésének megkeresésére. Bemutatom, hogyan használható az illesztett modell a hálózat esetlegesen még nem ismert kapcsolatainak felderítésére.

A második rész az egymással átfedő csoportosulások (sűrű részgráfok) jelenségét vizsgálja ritka hálózatokban. Bevezetek egy, a fuzzy partíciós mátrixokon és a hálózat pontjainak egy megfelelő hasonlósági mértéken alapuló algoritmust, amely alkalmas a hálózat potenciálisan átfedő csoportosulásainak és a csoportosulások között átkötő hídpontjainak azonosítására. Végül bemutatok néhány esettanulmányt a leírt módszerek valós adathalmazokon történő alkalmazásáról.

Nyilatkozat

Alulírott Nepusz Tamás kijelentem, hogy ezt a doktori értekezést magam készítettem és abban csak a megadott forrásokat használtam fel. Minden olyan részt, amelyet szó szerint, vagy azonos tartalomban, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Budapest, 2008. július 5.

.....
Nepusz Tamás

Acknowledgments

I would like to thank my advisors, Fülöp Bazsó and György Strausz for guiding my work during the last few years. I learned a lot during this time and I am convinced that this knowledge will help me in the future.

I would like to thank my colleagues and co-authors whom I worked with on the projects presented in this dissertation: Andrea Petróczi for discussing issues related to social networks and for supervising my work during my visit at Kingston University; László Négyessy and László Kocsis for our work on cortical networks (which would be fairly incomplete without the additional biological conclusions that I could not have come to due to my limited insight on the field); Gábor Tusnády and Péter Erdős for introducing us to Szemerédi's regularity lemma and for initiating the whole research direction that led to one of the models presented in this dissertation; László Zalányi for always being ready for discussions and debates on my research.

My thanks to all the members of the Computational Neuroscience Group at KFKI. It was a pleasure to work there and to enjoy the friendly atmosphere of the group. My special thanks to Gábor Csárdi for starting the `igraph` project and letting me obfuscate and abuse his carefully written source code to aid me in my research.

Last, but not least, I am grateful to my family and Ági for their continuous and unconditional support. This dissertation could not have been brought to completion without their patience, understanding and love.

Köszönetnyilvánítás

Köszönettel tartozom témavezetőimnek, Bazsó Fülöpnek és Strausz Györgynek, amiért irányították munkámat az utóbbi néhány évben. Sokat tanultam tőlük ez idő alatt, és biztos vagyok benne, hogy ez a tudás a későbbiekben még hasznomra válik.

Szeretném megköszönni kollégáim és szerzőtársaim segítségét, akikkel a jelen disszertációban bemutatott témákon együtt dolgoztam. Hálás vagyok Petróczi Andreának a közösségi hálózatokon végzett közös munkáért és a lehetőségért, hogy fél évet eltölthettem a Kingston University-n; Négyessy Lászlónak és Kocsis Lászlónak az agykérgi hálózatokkal kapcsolatos kutatásainkért (amely nélkülük bizonyára csak félkész állapotban lenne még most is, hiszen a munka biológiai következményeinek vizsgálata az ő érdemük); Tusnády Gábornak és Erdős Péternek, amiért megismertettek engem és kollégáim a Szemerédi regularitási lemmával és elindították azt a kutatási irányt, amely eredménye a jelen disszertáció egyik teljes fejezete lett; Zalányi Lászlónak, amiért mindig készen állt arra, hogy a kutatásaimmal kapcsolatos kérdéseket megvitassuk.

Köszönetet mondok a KFKI Részecske- és Magfizika Kutatóintézete Biofizikai osztályának a kutatócsoport barátságos légköréért és a kutatásaim során nyújtott segítségükért az utóbbi négy év során. Külön köszönet illeti Csárdi Gábort, amiért belevágott az `igraph` projektbe és rendelkezésemre bocsájtotta a forráskódot, hogy a saját munkámhoz felhasználjam.

Végül, de nem utolsósorban hálás vagyok családomnak és Áginak a folyamatos és feltétel nélküli támogatásukért. Ezt a disszertációt nem tudtam volna befejezni szeretetük, megértésük és végtelen türelmük nélkül.

Contents

1 Introduction **1**
- 1.1 Basics of graph theory . . . 1
- 1.2 Random graph models . . . 4
  - 1.2.1 Erdős-Rényi graphs . . . 4
  - 1.2.2 Small world networks . . . 5
  - 1.2.3 Scale-free networks and the principle of preferential attachment . . . 7
  - 1.2.4 Community structure . . . 9
- 1.3 Further reading . . . 12

2 Link prediction in complex networks **13**
- 2.1 Overview . . . 15
  - 2.1.1 Prediction by local similarity indices . . . 15
  - 2.1.2 Prediction by path ensembles and random walks . . . 17
  - 2.1.3 Prediction based on stochastic network models . . . 18
- 2.2 The model framework . . . 19
  - 2.2.1 Formal description . . . 19
  - 2.2.2 Extended preference model . . . 21
- 2.3 Basic statistical properties . . . 22
  - 2.3.1 Analytical results . . . 22
  - 2.3.2 Numerical simulations . . . 26
- 2.4 Fitting the model to data . . . 27
  - 2.4.1 The goal function for model fitting . . . 28
  - 2.4.2 Fitting by expectation-maximisation . . . 30
  - 2.4.3 Fitting by Markov chain Monte Carlo methods . . . 33
  - 2.4.4 Combining EM and MCMC methods . . . 39
  - 2.4.5 Choosing the number of vertex types . . . 40
- 2.5 Running time considerations . . . 44
  - 2.5.1 Network generation . . . 44
  - 2.5.2 Model fitting . . . 46
- 2.6 Performance measurements . . . 48

2.6.1 Fitting the model with given number of groups 49
2.6.2 Choosing the number of groups 51
2.6.3 Rapid mixing of the Markov chain 54
2.7 Using the preference model for predicting unknown links . . . 54
2.8 Conclusion 57

3 Fuzzy community structure in complex networks 58
3.1 Overview 61
3.2 Basic concepts 63
3.2.1 Fuzzy partition matrices 63
3.2.2 Similarity and the goal function 64
3.3 Finding fuzzy communities in undirected networks 66
3.3.1 Outline of the algorithm 66
3.3.2 Connection weights 73
3.3.3 Choosing the number of communities 74
3.4 Identifying bridge vertices 75
3.4.1 Bridgeness 75
3.4.2 Centrality-weighted bridgeness 76
3.4.3 Exponentiated entropy 77
3.5 Benchmark results 78
3.5.1 Nonoverlapping community structure 79
3.5.2 Overlapping community structure 79
3.5.3 Running time 81

4 Applications 83
4.1 Predicting missing neural connections in cortical networks . . 83
4.1.1 The dataset 84
4.1.2 Results 85
4.1.3 Other prediction approaches 94
4.2 Higher level brain areas in the visuo-tactile cortex 95
4.2.1 Results 96
4.2.2 Comparison with other approaches 98
4.3 Detection of social bridges via fuzzy communities 99
4.3.1 The UK university faculty dataset 99
4.3.2 The network science co-authorship graph 100

5 Conclusions 105
5.1 Link prediction in complex networks 105
5.2 Fuzzy communities in complex networks 106

A Technical background **109**
A.1 Generating random numbers 109
A.2 The `igraph` library 110
A.2.1 The basic graph representation in `igraph` 111
A.2.2 An example: calculating SimRank scores 113

Bibliography **114**

List of Figures

1.1 An example graph and one of its diameters 3
1.2 A comparison of regular lattices, small world networks and Erdős-Rényi random graphs 7
1.3 Comparison of Poisson and power-law distributions 9
1.4 A social network with strong community structure 10

2.1 Two graphs generated by the preference model 21
2.2 Expected and observed degree distribution of a random graph according to the undirected preference model 27
2.3 Acceptance rates (left) and log-likelihoods (right) during a typical run of the MCMC algorithm in the function of time. . 36
2.4 Illustration of SVD-based matrix approximation 42
2.5 Performance assessment of the preference model fitting method 51
2.6 Eigenvalues of the directed Laplacian matrix for graphs generated by the preference model 52
2.7 Singular values of the adjacency matrix of a graph generated by the preference model 53
2.8 Rapid mixing of the Markov chain in the fitting process of the preference model 55

3.1 Illustration of hard and fuzzy partitions of a graph 60
3.2 Drawing Dirichlet-distributed random vectors from the range allowed in the fuzzy community detection problem. 69
3.3 Local goal function of the fuzzy community detection algorithm 71
3.4 Performance of the fuzzy community detection algorithm. . . . 80
3.5 Running time of the fuzzy community detection algorithm . . 82

4.1 Adjacency matrix of the visuo-tactile cortex dataset. 86
4.2 The predicted adjacency matrix of the visual cortex 89
4.3 Predicted pobability of connections in the visuo-tactile cortex. 93
4.4 Comparison of the prediction method to alternative approaches 95
4.5 Bridge areas in the visuo-tactile cortex of the macaque monkey 96

4.6 Bridgeness scores versus vertex degrees in the cortical network dataset 97
4.7 Fuzzy communities of the UK university dataset 100
4.8 Comparison of the unweighted and degree-weighted bridgeness scores in the UK university dataset 101
4.9 Vertex pair weight distribution in different networks. 102
4.10 Visualisation of the network science co-authorship dataset. . . 103

List of Tables

4.1 Basic properties of the cortical networks 85
4.2 Log-likelihoods, AIC values, $\sqrt{r_p r_n}$ and MCC in the visual cortex 87
4.3 Comparison of the reconstruction of the known parts of the cortical network base on the preference model and the results of Costa et al. [25]. 88
4.4 Comparison of the predictions of Costa et al. and Jouve et al. with the preference model regarding the unknown parts of the cortical network dataset. 90
4.5 Likelihoods, AIC values, $\sqrt{r_p r_n}$ and MCC in the visuo-tactile cortex 91
4.6 Identified bridge vertices in the cortical network dataset by various overlapping community detection algorithms. 98

List of Algorithms

1 Generating graphs according to the $G_{n,p}$ model 4
2 Generating graphs according to the preference model 20
3 Fitting the preference model by expectation-maximisation . . 33
4 Canonical rearrangement of vertex type assignments 34
5 Fitting the preference model by the Metropolis–Hastings algorithm 38
6 Post-processing fuzzy community detection results based on the exponentiated entropy 78

1

Introduction

This dissertation is about networks: complex interconnected systems, where individual entities are connected by binary links arranged in seemingly random but intrinsically structured patterns. These entities can practically be anything that is of interest to science: proteins in the human genome, related to each other by their possible interactions [58]; modules or functions of a large software system connected by dependence relations [84]; financial networks [100]; even human beings linked together by their everyday social intercourse [105]. Real computer networks also fit in this framework in a very straightforward way; in fact, the router-level network model of the Internet was in the focus of numerous studies since the emergence of network theory as a separate field in the late nineties (see [38] for an introductory paper). Excellent overviews of the advances in the study of complex networks are given in [6, 15, 129], while [98] contains a collection of relevant papers, starting from the early work of Erdős and Rényi [35].

Networks are collections of binary relations among entities. Binary relations can naturally be transformed to graphs, allowing one to study the properties of the network by the tools of a well-established field of mathematics, namely graph theory. It is thus appropriate to start the introduction with the fundamental definitions and theorems related to graphs.

1.1 Basics of graph theory

There are many equivalent graph definitions in the literature. The one presented here is according to the book of Diestel [30]. A *graph* is a pair $G = (V, E)$ of sets such that $E \subseteq V \times V$; thus, the elements of E are 2-

subsets of V[1]. Elements of V are called the *vertices* of the graph (hence the notation: V stands for the initial of "vertex"). Similarly, elements of E are called *edges*. A graph with no edges ($E = \emptyset$) is called an *empty graph*, denoted by $\emptyset$. Vertex pairs in E can either be ordered or unordered. In the former case, we are talking about *directed* graphs; otherwise the graph is *undirected*. Edges are said to *connect* the vertices. A directed edge $e = (x, y)$ $(x, y \in V)$ is said to *originate* from x and *terminate* in y. x and y are both *endpoints* of the edge. In an undirected graph, we simply say that x and y are *adjacent* to edge e. An edge (x, y) is a *loop* if $x = y$. Graphs without loop edges are called *simple graphs*.

Sometimes it is easier to think about an undirected edge as directed edges in both directions. From now on, all definitions related to directed graphs can also be applied to undirected ones by substituting every undirected edge by two directed ones. For the sake of notational simplicity, we often abbreviate $e = (x, y)$ as $x \to y$ if the graph is directed, and $x \leftrightarrow y$ if it is undirected.

Adjacency is also defined for vertex and edge pairs: two vertices x and y are adjacent if and only if $x \leftrightarrow y$ is an edge of the graph, while two edges e and f are adjacent if they have an endpoint in common. For directed graphs, we can also define *successors* and *predecessors*: vertex y is a successor of x (and x is a predecessor of y) if and only if $x \to y \in E$.

The *degree* of a vertex x is the number of adjacent edges of this vertex. For directed graphs, one can separately define the *out-degree* and the *in-degree* of vertex x as the number of edges originating from and terminating in x. It is straightforward that the degree of x equals the sum of the out- and the in-degree of x. A vertex with degree 0 is *isolated*.

The graph $G'(V', E')$ is a subgraph of graph $G(V, E)$ if $V' \subseteq V$ and $E' \subseteq E$. This can be denoted by $G' \subseteq G$. If $G' \subseteq G$ and E' contains all the edges $x \to y \in E$ where $x, y \in V'$, we say that V' induces G' in G.

A *path* P of a graph is a sequence of edges $\{v_0 \to v_1,\ v_1 \to v_2,\ \ldots,\ v_{n-1} \to v_n\}$. It is said that P goes from v_0 to v_n and its length is n (since it contains n edges). The path is a *cycle* if $v_0 = v_n$. These definitions also apply to undirected graphs, since one can convert every undirected graph to a directed counterpart by replacing $v_i \leftrightarrow v_j$ with $v_i \to v_j$ and $v_j \to v_i$. A path is said to be *shortest* between vertices v_i and v_j if it starts in v_i, terminates in v_j and there is no shorter path between these two vertices. There can be multiple shortest paths between any two vertices, but the length of the shortest path is uniquely defined (and assumed to be infinity if there is no

[1]This definition assumes that we do not allow multiple edges connecting the same pair of vertices. However, this is not a limitation in our case, the networks studied in this dissertation do not possess multiple edges.

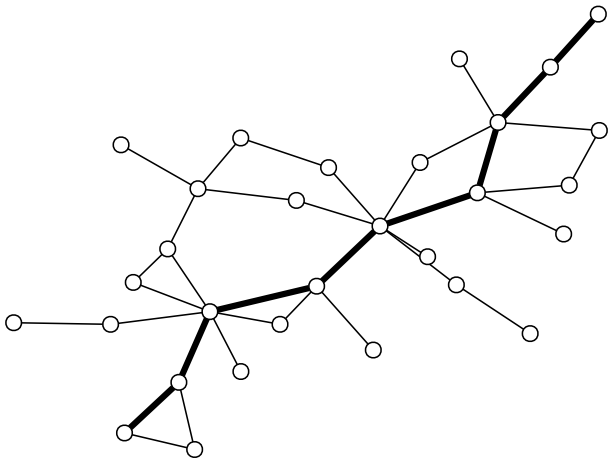


FIGURE 1.1: An example graph and one of its diameters

appropriate path between the vertices). The length of the shortest path is sometimes called the *distance* of the vertices involved. The *diameter* of a graph is the maximum distance over all possible vertex pairs, or a path attaining that distance, depending on context.

A graph is *strongly connected* if there exists a path between any two vertices. Graphs with only a single vertex are strongly connected by definition. Graphs that are not strongly connected can be decomposed to subgraphs that are strongly connected themselves. More precisely: given an arbitrary graph $G(V, E)$, there exists a partition of V into k disjoint subsets $V_1, V_2, \ldots, V_k$ such that $\cup_{i=1}^{k} V_i = V$ and every subgraph of G induced by V_i is strongly connected. (Note that $k = 1$ if the graph itself is strongly connected). The subgraphs induced by V_i's are called the *strongly connected components* of G. The diameter of a disconnected graph is infinite.

A graph is *bipartite* if there exists a partition of its vertex set V into subsets V_1 and V_2 such that $V_1 \cap V_2 = \emptyset$, $V_1 \cup V_2 = V$ and $E \subseteq V_1 \times V_2$. Informally speaking, vertices of a bipartite graph can be coloured by red and blue in a way that all edges go between a red and a blue vertex.

The usual way of visualising a graph is to draw its vertices as dots in the 2D or 3D Euclidean space and connect pairs of vertices with a (not necessarily straight) line if there is an edge connecting them. Arrowheads are drawn on the edges when they are directed. The actual placement of the vertices in space does not matter; however, certain layouts are more pleasing to the eye than others. To illustrate some of the concepts introduced in this section, Fig. 1.1 shows an example graph with one of its diameters highlighted by thick edges.

1.2 Random graph models

We study networks in order to understand the underlying mechanisms that generated these networks, and a possible way to do that is to construct models that are simple enough to work with while simultaneously conveying enough detail to reconstruct the most important features of the network being studied. This section will introduce some basic network models that were either able to reproduce significant aspects of real-world networks or have proven to be useful null models for significance tests of statistical hypotheses regarding real networks.

1.2.1 Erdős-Rényi graphs

Probably the most basic random graph model one can conceive is as follows: let there be n vertices and for every undirected pair of vertices, connect them with probability p, independently of other possible edges or any intrinsic properties of the vertices. This model (denoted by $G_{n,p}$) was first introduced by Solomonoff and Rapoport [117] and studied further in the papers of Erdős and Rényi [35, 36, 37]. A practically equivalent definition of the model (denoted by $G_{n,m}$) is as follows: consider the ensemble of all simple graphs with n vertices and m edges and choose one of them randomly with equal probabilities. From now on, I study the $G_{n,p}$ model, noting that $p = 2m/(n^2 - n)$ results in a random graph whose expected edge count is exactly m.

An algorithm for generating graphs according to the $G_{n,p}$ model is described by pseudo-code in Algorithm 1. The loop in line 2 iterates all possible ordered vertex pairs if the graph is directed or over all possible unordered vertex pairs if it is undirected, excluding loop edges.

Algorithm 1 Generating graphs according to the $G_{n,p}$ model

Require: $n \geq 0$ and $0 \leq p \leq 1$
1: $G :=$ empty graph with n vertices
2: **for all** $\{v_1, v_2\} \in V(G) \times V(G)$ **do**
3: $q :=$ random number between 0 and 1, inclusive
4: **if** $q \leq p$ **then**
5: $E(G) := E(G) \cup \{v_1, v_2\}$
6: **end if**
7: **end for**
8: **return** G

Erdős and Rényi were interested in answering questions about specific properties of these graphs as n approaches infinity while keeping $z = np$

constant. They showed that the degree distribution of the limit graph is a Poisson distribution around z, the mean degree of the network. They also proved that the size of the largest connected component of the $G_{n,p}$ model tends to infinity in the limit of the infinite graph ($n \to \infty$) if $z > 1$, but not if $z < 1$. $z = 1$ is the point where a phase transition occurs and a giant component emerges.

1.2.2 Small world networks

Pál Erdős was one of the most influential mathematicians of the twentieth century with hundreds of mathematical papers and 511 direct collaborators [47]. In honour of his contributions to mathematics, Goffman [47] introduced the notion of Erdős number, a measure for the "collaboration distance" between an arbitrary scientist and Erdős. Erdős himself had an Erdős number 0, and all his collaborators were assigned an Erdős number 1. In order to obtain a finite Erdős number, one has to co-author a paper with someone who has a finite Erdős number. The Erdős number of an author is $k+1$ if the lowest Erdős number of his co-authors is k. It is easy to prove that the Erdős number of an author is the length of the shortest path from himself to Erdős in the co-authorship network of science, where two scientists are connected by an edge if they have collaborated in at least one published scientific paper.

A peculiar property of the distribution of Erdős numbers is that amongst all working mathematicians up to the turn of the millenium having finite Erdős numbers (roughly 268 000 people), the largest one (the diameter of the network) was only 13 and the mean was 4.65 [50]. Even scientists apparently completely unrelated to mathematics occasionally have surprisingly small Erdős numbers. For instance, Steven Brams (a political scientist) has an Erdős number 2. A large fraction of the genetics community also has a low average Erdős number due to a joint paper between the geneticist Eric Lander and the mathematician Daniel Kleitman [101]. These authors can be considered as bridges between distinct scientific fields, shrinking the diameter of the graph effectively. I will return to bridge vertices later in Chapter 3.

This so-called small world phenomenon, characterized by low average path length and small diameter compared to the number of vertices, is abundant in many real-world networks. One may argue that the scientific co-authorship network is an artificial network constructed by mathematicians who had too much spare time, therefore I cite a more realistic example: the Internet. According to recent Internet maps constructed by traceroute-style path probes [18], the diameter of the Internet on its router level was only 43 in June 2006, despite the fact that this network contained 222 935 routers and 279 511 undirected connections. The phenomenon is even more pro-

nounced in the case of the World Wide Web: although there are millions of web pages in the World Wide Web, two randomly chosen web pages are only 19 clicks away from each other on average [4].

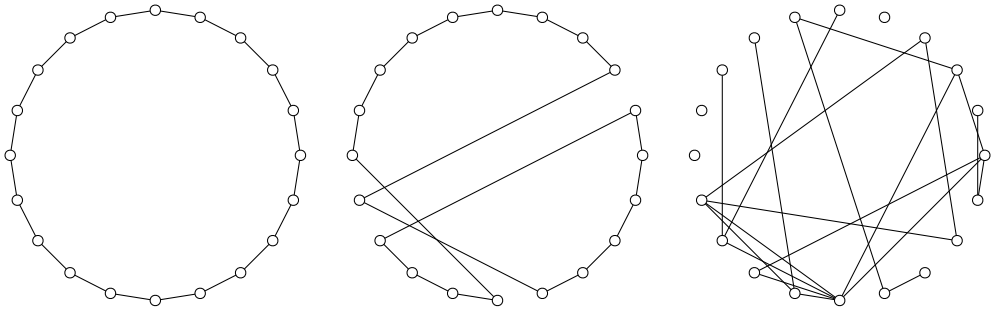
The first written appearance of the small world phenomenon is probably in the 1929 short story of the Hungarian writer Frigyes Karinthy, entitled *Chains* (*Láncszemek* in Hungarian) [60]. Karinthy believed that due to advances in communication and travel in the early 20th century, the density of the acquaintance network among human beings is increasing, making the social distance between people smaller and smaller. He writes:

> A fascinating game grew out of this discussion. One of us suggested performing the following experiment to prove that the population of the Earth is closer together now than they have ever been before. We should select any person from the 1.5 billion inhabitants of the Earth – anyone, anywhere at all. He bet us that, using no more than five individuals, one of whom is a personal acquaintance, he could contact the selected individual using nothing except the network of personal acquaintances.[2]

One of the first scientific descriptions of the small world phenomenon is in the famous paper of Milgram [80]. He experimented with sending a packet from one individual in the United States to another one by forwarding the packet only to close acquaintances step by step. Using the results of these experiments, he estimated the expected number of "handshakes" needed to get from an arbitrary person to another one. This number was six, which is very small compared to the whole population of the United States. On the other hand, it is a well-known fact in sociology that given three individuals A, B and C, the connections $A \leftrightarrow B$ and $B \leftrightarrow C$ are precursors of a connection between A and C. Pure random Erdős-Rényi networks do not show the latter property at all (since edges were placed independently), so a new model had to be introduced which maintains the short average path length of Erdős-Rényi networks while possessing the above mentioned clustering of connections. The model of Watts and Strogatz [130] provided a possible explanatory mechanism as follows:

1. Let the vertices of the network be placed on a regular lattice.
2. For each edge, disconnect one of its endpoints with probability p and connect it to a uniformly chosen vertex, disallowing loop and multiple edges.

[2]Translated by Ádám Makkai and Enikő Jankó.



(a) p=0 (b) p=0.2 (c) p=1

FIGURE 1.2: A comparison of regular lattices (a), Watts-Strogatz small world networks (b) and Erdős-Rényi random graphs (c)

In the context of social networks, the connections of the original, regular lattice represent our social circle (people who we meet regularly), and the random connections model the fact that new social ties are formed randomly with people who are not parts of our social circle. As p increases, the average path length decreases rapidly, and in the limit of an infinite network, the average path length scales only logarithmically with the size of the network. Informally speaking, we can say that p controls the amount of randomness in the model: $p = 0$ results in a highly structured, regular network while $p = 1$ rewires every edge, making it practically equivalent to an Erdős-Rényi random graph. This is illustrated on Fig. 1.2.

1.2.3 Scale-free networks and the principle of preferential attachment

Real networks usually differ in at least one more property from Erdős-Rényi networks: the degree distribution. It is straightforward that the degrees in an Erdős-Rényi network generated by the $G_{n,p}$ model follow a binomial distribution $\mathrm{B}(n, p)$ (or $\mathrm{Pois}(np)$ in the infinite limit), since every vertex has n independent chance to form connections with other vertices, and each trial succeeds with probability p. Informally, this means that most vertices have a degree around np; there might be small deviations, but it is extremely unlikely to find a vertex whose degree is, say, a hundred times as much as the average.

Now, take the citation network of scientific papers as an example. In this network, papers are represented by vertices and a directed edge points from paper A to paper B if A cites B. Everyone who has ever done research work and has written scientific papers knows from experience that most of

the papers are never cited (their in-degree is zero), or maybe only a couple of times, while a small fraction of papers are cited all the time, even decades after their publication [81, 106]. Therefore, the degree distribution of such a network cannot follow a Poisson distribution. A more appropriate model is a power-law or Yule–Simon distribution [115]: the probability of a vertex having degree k is proportional to $k^{-\gamma}$, where γ is usually between 2 and 3 (see Figure 1.3 for the comparison of Poisson and power-law distributions). A possible explanation of the phenomenon was given by Price [107], based on two simple assumptions:

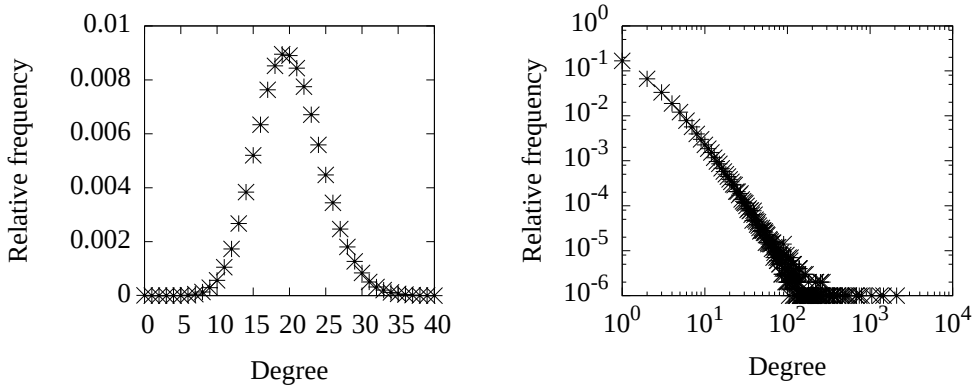
1. The network is *growing*: vertices are added one by one and each vertex connects to m already added vertices right after its birth. No new connections are initiated by the vertex afterwards.

2. The probability of a vertex receiving a new connection from newcomers is proportional to the degree of the vertex. This is the "rich gets richer" principle: the more incoming edges a vertex has, the more likely it is to get another incoming edge. Newly added vertices prefer those vertices that already have a lot of connections.

The principle was popularised years later by Barabási and Albert [7] under the name of *preferential attachment*. It was shown by Dorogovtsev et al. [32] that the model generates networks with power-law degree distributions ($\gamma = 3$). Since then, several other alternative models have been proposed that similarly give rise to power-law degree distributions (e.g., [67, 71, 103]).

Networks having a power-law degree distribution are often called *scale-free networks*, since $f(ck) \propto f(k)$ for any $c > 0$, where $f(k)$ is the probability density function of the degree distribution. In other words, scaling k does not change the shape of the probability density function.

A key difference between Erdős-Rényi and scale-free networks is that vertices of an Erdős-Rényi random graph have a "typical" mean degree and we cannot expect large deviations from that. In contrast, the degree distribution of scale-free networks implies the existence of vertices that have significantly larger degree than the mean degree of the network. These vertices are called the *hubs* of the network, and they are responsible for the low diameter, since any vertex chosen arbitrarily will almost surely have a hub in its proximity. At the same time, hubs are the Achilles heels of the network, since an attacker who is able to systematically delete the hubs can quickly break the network into small, isolated components. This disadvantage is counterbalanced by the fact that random vertex deletions (random failures in the network) hardly affect the overall connectivity of a network, since a randomly selected vertex is likely to have a low degree. That is a possible explanation to why the



(a) Erdős-Rényi networks (b) Preferential attachment

FIGURE 1.3: The comparison of the degree distribution of Erdős-Rényi $G_{n,p}$ graphs ($n = 10^6$, $p = 2 \times 10^{-5}$) and networks generated by the preferential attachment model of Barabási and Albert ($n = 10^7$, $m = 1$) [7]. The plot on panel (b) has a log-log scale.

Internet works well in spite of the fact that at any given moment there are several routers turned off due to failure, and why it is vulnerable to attacks targeted at the Internet backbone or the root DNS servers.

Finally, I note that scale-free degree distributions and small world properties are usually meaningful only in the context of large networks; one can not say that a network with only a few hundred or thousand vertices possesses such properties. The number of vertices in networks studied in this dissertation is not in the order of magnitude where it is possible to meaningfully talk about scale-free or small world properties; however, the overview of network science given in this introductory chapter would not have been complete without mentioning these concepts. Results presented in this dissertation presume neither small world properties nor scale-free degree distributions.

1.2.4 Community structure

The last peculiar feature of real-world networks I will discuss here is the existence of densely connected subgraphs (communities or clusters) in otherwise sparse networks [46]. This is a feature that is reproduced by the model of Leskovec et al. [71], along with the now well-known power-law degree distribution and a small diameter that shrinks as the network grows. A typical network with strong community structure is shown on Figure 1.4.

Community detection was one of the most studied problems in network theory during the last few years. Many algorithms have been proposed based

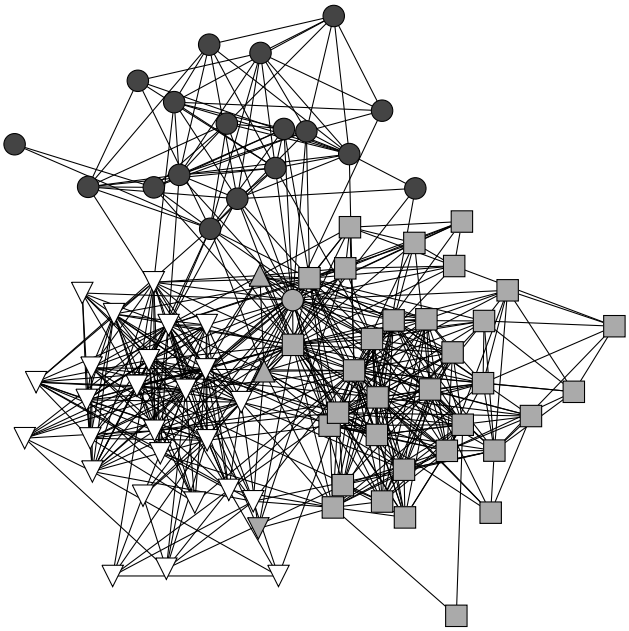


FIGURE 1.4: The social network of the academic staff of a faculty at a UK university [92]. Vertex shapes denote the school affiliations of the staff, vertex colors represent the communities detected by the algorithm of Latapy and Pons [69].

on simple graph-theoretic concepts like edge betweenness [46] and cliques [102], physical analogies [108], eigenvectors of different matrices [17, 96], random walks [69, 124] and the heuristic optimisation of a measure called modularity [27, 94, 127]. The problem of community detection is too complex to be treated properly in such a short overview as this section, so I will only discuss the concept of modularity, which will be needed later in Chapter 3.

Most community detection algorithms calculate a partition of the vertices in the network. Intuitively, a good partition is one that maximises intra-community edge density and minimises the number of edges going between communities. This intuitive definition should be treated carefully, though: a trivial partition that puts all vertices in the same community would attain the smallest possible inter-cluster edge density, but we still do not consider it a good partition. The reason why we think that the trivial partition is not meaningful is that we would be able to attain the same intra-cluster and inter-cluster edge densities with this partition even in the case when the edges are placed completely randomly among the same vertices. A more precise statement would be that a good partition puts *more* edges inside the clusters than what we would expect from the same partition if the edges were randomly rewired while keeping the degree distribution, and likewise, it puts

less edges between the clusters than the expectation under the assumption of complete randomness.

A random graph model (called the *configuration model*) that is able to generate uncorrelated networks conditioned on a predefined degree distribution was depicted by Molloy and Reed [82]. It can be shown that given a degree sequence $s_0, s_1, s_2, \ldots$ (where s_i represents the number of vertices with degree i), the probability of an edge between vertices of degree i and j is $ij/\sum_{k=0}^{\infty} ks_k$ if the network is uncorrelated. The denominator is exactly twice the number of edges m. The modularity score of a given partition is then the sum of difference between the observed (zero or one in the case of simple graphs) and the expected number of edges (as calculated from the configuration model) for all intra-cluster vertex pairs, divided by $2m$ to normalise the measure:

$$Q = \frac{1}{2m} \sum_{i=1}^{n} \sum_{j=1}^{n} \left(A_{ij} - \frac{d_i d_j}{2m} \right) \delta_{ij} \tag{1.1}$$

where A_{ij} is 1 if there is an edge between vertices i and j; zero otherwise. d_i and d_j are the degrees of vertex i and j, respectively. δ_{ij} is 1 if vertices i and j are in the same community according to the partition being assessed; zero otherwise. Several community detection algorithms strive to maximise the modularity of the obtained partition [27, 94, 98, 127]. According to Newman [94], a modularity score larger than 0.3 indicates the presence of a strong community structure in the network.

Note that the modularity score is not the property of the network, it corresponds to a *given partition* of the network, and in general it is hard to compute the exact partition that maximises modularity.[3] E.g., the network on Figure 1.4 with the partition defined by the *colors* of the vertices has modularity 0.4317, while the partition defined by the *shapes* has modularity 0.4259.

A relatively new result is that many complex networks show an overlapping community structure [102]. Therefore, it is presumed that a more accurate description can be obtained by allowing vertices to belong to multiple groups at the same time. The development of algorithms that are able to handle this situation is an open problem in the field of network research at present (see [17, 92, 102, 108, 135] for possible solutions). The problem of overlapping communities will be treated in detail in Chapter 3.

[3] A different modularity measure was introduced in [138] that is independent from a given partition and assesses the network as a whole.

1.3 Further reading

Additional introductory material to network theory is to be found in the popular books of Buchanan [15], Barabási [6] and Watts [129]. The reviews of Albert and Barabási [3], Dorogovtsev and Mendes [31], Newman [93] and Boccaletti et al. [11] dive more into the technical details. The textbook of Bollobás [12] is a comprehensive overview of the theory of random graphs. An excellent collection of related papers is published in [98].

2

Link prediction in complex networks

Depending on the way a network model of a real-world dataset is devised, the network may be complete or incomplete. For instance, a scientific co-authorship network generated automatically from the database of a preprint archive (e.g., Cornell University Library's arXiv service at `http://arxiv.org`) is usually complete (see [95, 97] for analyses on scientific co-authorship networks), since the underlying dataset is clear-cut and it is easy to tell whether two vertices are connected or not. On the other hand, network models arising from the field of biology (such as metabolic networks of organisms [57], protein interaction networks [58] or cortical networks [86]) can be incomplete. For example, the existence of a neural connection between two brain areas in a cortical network can only be decided by checking it experimentally, but such experiments are usually expensive, and some connections are extremely hard to examine due to methodological difficulties. Complications like this arise not only in biological datasets: the map of the Internet reconstructed from traceroute-style path probes [18] is also sub-

Related publications:

Nepusz T., Négyessy L., Tusnády G., Bazsó F.: *Reconstructing cortical networks: case of directed graphs with high level of reciprocity.* In Béla Bollobás, Róbert Kozma, and Dezső Miklós, editors, *Handbook of Large-Scale Random Graph Methods*, volume 18 of *Bolyai Society Mathematical Studies*. Springer, 2008. ISBN 978-3-540-69394-9.

Nepusz T., Bazsó F.: *Maximum likelihood methods for data mining in datasets represented by graphs.* In: IEEE Proceedings of the 5th International Symposium on Intelligent Systems and Informatics, Subotica, Serbia, 24-25 August 2007, pp. 161-165.

Nepusz T., Bazsó F.: *Likelihood-based clustering of directed graphs.* In: IEEE Proceedings of the 3rd International Symposium on Computational Intelligence and Intelligent Informatics, Agadir, Morocco, 28-30 March 2007, pp. 189-194.

ject to missing data. Existing wired links might never appear on the map if the traceroute probe packets are never routed through them, and even complete subnetworks can conceal themselves by (accidentally or deliberately) discarding ICMP echo packets.

In this chapter, I discuss the problem of predicting previously unknown links in complex networks. The motivation of this research direction was a study on cortical networks [91] for which I needed an algorithm that is able to draw inferences about missing links in a network from known link patterns. The results of this specific study will be briefly discussed later in Section 4.1, this chapter deals only with the methodological issues.

From now on, I will assume that we study a network with the following properties:

1. All vertices of the network are known, or if they are not, unknown vertices and their connections are not relevant.

2. The connections of the network can be classified as follows:

   Known existing connections. These connections have been explicitly checked and confirmed.

   Known nonexisting connections. These connections have been explicitly checked and found to be nonexistent.

   Unknown connections. These connections have never been checked, they might exist or they might be missing. Sometimes we have some *a priori* information or assumption about the probability of their existence. For instance, in cortical networks, it is accepted that connections are sparse between the visual and the sensorimotor cortex, therefore even if we know nothing about the specific connections between visual and sensorimotor areas, we can assume that roughly 10% of those connections exist, therefore the probability of existence for any individual connection between these two cortices is approximately 0.1.

My primary aim was to construct a method that is able to give meaningful predictions regarding the unknown connections. The validity of such a method can be tested by checking its predictions on the known connections. A secondary aim was to assign a confidence index to the predictions. I also wanted to keep the method as general and as widely applicable as possible, therefore I refrained from incorporating assumptions into the model if they were likely to hold only for a small subclass of the broad family of complex networks.

This chapter is organised as follows: first, a brief overview of generic prediction approaches is given. After that, a stochastic graph model is introduced and algorithms are given to fit the parameters of the model to the specific network whose unknown edges we are trying to predict. Benchmark results will be presented to illustrate the validity of the fitting algorithm. Finally, I will discuss how can one utilise the model in order to obtain meaningful predictions.

2.1 Overview

Link prediction is a relatively new field of research in network science. Some early experiments in cortical networks date back to the late 90's (e.g., [59]). The work of Liben-Nowell and Kleinberg [74] was one of the first papers that tried to evaluate a set of different similarity measures between vertices of a network in order to find the ones that predict unknown edges with high confidence. These measures can roughly be classified as (1) local similarity indices and (2) complex measures based on path ensembles and random walks. The key idea of link prediction based on these measures is very simple: two vertices tend to connect if they are similar according to a measure or a combination of some measures, and the task is to find which measure(s) give good results.

One can also derive similarity measures based on additional domain-specific knowledge. A good example of this strategy is presented in the paper of da Fontoura Costa et al. [25], where predictions on the cortical network made use of information like the spatial position of the different brain areas.

There is also a third approach one can take: by choosing an appropriate probabilistic network model and appropriately fitting the model parameters to the network instance being studied, one can evaluate the probabilities of the existence of unknown edges according to the model. These basic ideas will be discussed in the upcoming three subsections.

2.1.1 Prediction by local similarity indices

These indices try to capture the essence of vertex similarity by considering some local, easily computable properties of the vertices. The most widely used property is the set of neighbours $\Gamma(x)$ of a given vertex x; in fact, all the indices described in this section make use of this property. In the case of directed networks, one can also use the set of predecessors $\Gamma^+(x)$ and the set of successors $\Gamma^-(x)$. For a given pair of vertices x and y, the most common local similarity indices are as follows:

Number of common neighbors. This is simply defined as $|\Gamma(x) \cap \Gamma(y)|$.

Cocitation index [116]. A variant of the previous measure for directed networks, defined as $|\Gamma^+(x) \cap \Gamma^+(y)|$. Intuitively, if the vertices are books or scientific papers and an edge goes from A to B if A cites B, the cocitation index of x and y is the number of other papers citing both of them.

Bibliographic coupling [63]. The opposite of the cocitation index, defined as $|\Gamma^-(x) \cap \Gamma^-(y)|$. In other words, the bibliographic coupling of two papers is the number of papers they both cite.

Jaccard coefficient [54]. This is defined as $|\Gamma(x) \cap \Gamma(y)|/|\Gamma(x) \cup \Gamma(y)|$, the number of common neighbors divided by the number of vertices that are adjacent to at least one of the vertices considered.

Cosine similarity. A common similarity used in text mining. Cosine similarity works with binary vectors, the i-th element of $\mathbf{x}$ is 1 if $i \in \Gamma(x)$, 0 otherwise. The definition is as follows: $\theta(x, y) = \frac{\mathbf{xy}}{||\mathbf{x}|| \cdot ||\mathbf{y}||}$. Using the standard Euclidean vector norm, this can be reformulated as $\theta(x, y) = |\Gamma(x) \cap \Gamma(y)|/\sqrt{d_x d_y}$ where d_x and d_y are the degrees of the respective vertices. The measure can also be defined for only successors or predecessors in directed networks.

Adamic–Adar similarity measure [1]. Defined as $\sum_{z \in \Gamma(x) \cap \Gamma(y)} 1/\log |\Gamma(z)|$, this measure evaluates the degrees of the common neighbours of x and y and penalises high degree neighbours, as a vertex with high degree has a higher chance to be in the common neighbourhood of other vertices anyway. In other words, the fact that me and someone else both know who Albert Einstein is does not tell an independent observer anything meaningful about the similarity of me and that other person. On the other hand, the fact that we both know the lead guitarist of the local rock band implies a higher degree of similarity between us (e.g., that we both live in the same area and we are both keen on rock music).

These measures are easy to calculate and are still powerful enough to be used in predicting unknown connections. However, one should take into account that these methods treat all unknown connections as nonexistent. This approach is valid in cases when the network is dynamic and one would like to extrapolate to the future of the network based on the past and present (e.g., answering such questions like "which scientists are likely to publish a joint paper in the near future, given their existing publication records"), but may fail in the static case, as it will be illustrated in Section 4.1. A

recent data mining survey on telephone call networks utilising this approach is presented in [68], which also contains a more detailed discussion of local similarity measures.

2.1.2 Prediction by path ensembles and random walks

These measures derive vertex similarity from paths and random walks between two given vertices x and y. They are often computationally expensive, and surprisingly enough, the additional complexity does not always pay off, since similar predictive power can be achieved with simpler methods as well (see [74] for a comparison on scientific co-authorship networks).

Katz similarity score [62]. This score simply takes all possible paths between vertices x and y. Since the number of paths between any two vertices can theoretically be infinite (edge and vertex repetitions are allowed), the weight of a path is exponentially damped by its length. The sum of the path weights consitutes the Katz similarity score: $\sum_{k=1}^{\infty} \beta^k |P_{x,y}^{\langle k \rangle}|$, where $0 < \beta < 1$ is an arbitrary parameter and $P_{x,y}^{\langle k \rangle}$ is the set of all paths of length k between x and y. It can be shown that the equation above can be reformulated as $(\mathbf{I} - \beta \mathbf{A})^{-1} - \mathbf{I}$, where $\mathbf{A}$ is the adjacency matrix of the graph and $\mathbf{I}$ is the identity matrix.

Hitting time, commute time. Consider a random walk starting from x which follows the outgoing edges of the current vertex with equal probability. The hitting time $H_{x,y}$ is then defined as the expected number of steps required for a random walk to reach y from x. Since the hitting time is not symmetric for directed graphs, one can also consider the return time $H_{x,y} + H_{y,x}$, which is the expected number of steps required to start from x, visit y at least once and then return to x. Both the hitting time and the return time are large when the vertices are dissimilar, so one should use their inverse as a similarity measure.

Personalised PageRank [14, 41, 56]. A variant of the PageRank measure used in the Google search engine. The personalised PageRank considers a random walk starting from x which follows one of the outgoing edges with probability λ or returns to x with probability $1 - \lambda$. The personalised PageRank measure of x and y is then the probability of y in the stationary distribution of this random walk.

SimRank [55]. This measure is defined recursively: two vertices are similar if they are referenced by similar predecessors. Formally, the similarity

of two vertices is the average similarity of all possible predecessor pairs, multiplied by a damping factor $\gamma \in [0, 1]$:

$$\mathrm{SimRank}(x, y) = \gamma \frac{\sum_{a \in \Gamma^+(x)} \sum_{b \in \Gamma^+(y)} \mathrm{SimRank}(a, b)}{d_x^+ d_y^+} \tag{2.1}$$

This recursive definition is then solved by a fixed point iteration in order to obtain the SimRank values, under the trivial constraint that $\mathrm{SimRank}(x, x) = 1$. An alternative definition of SimRank is that it is the expected value of γ^L, where L is a random variable giving the time when two random walks started from x and y meet for the first time.

2.1.3 Prediction based on stochastic network models

All the methods described so far treat unknown connections as nonexistent, therefore it is complicated to incorporate our *a priori* beliefs about the probability of a given unknown connection into these measures; e.g., how should we consider a path of length 2 between x and y in the Katz similarity score when one of the edges in this path has a probability of only 0.3? A better approach in this case is to consider an appropriate random stochastic network model and let the probability of an edge between x and y in this model be the similarity of x and y. This approach assumes nothing more and nothing less than the previous methods, the only assumption is that an existence of an edge implies similarity of the endpoints and vice versa. The interesting part is then to find a model that describes the network appropriately and in case of a parameterised model, estimate the parameters in a way that maximises the likelihood of the model, given the network observed.

A typical example of this method is given in the paper of Clauset et al. [22]. Their model builds the network in a bottom-up hierarchical fashion. Initially, every vertex belongs to a single isolated group. In step i, two groups are selected randomly with equal probability and edges *between* groups are generated independently with probability p_i. Note that at this point, the edges *inside* the groups being joined are already generated in earlier steps. The process stops when there are no more groups to join. The model is parametrized by the number of vertices n, the probabilities $p_1, p_2, \ldots, p_{n-1}$ and the order of groups in which they are joined. Markov chain Monte Carlo methods are then utilised to find the best fit of this model to a given network. Finally, edges are classified according to their presence in the original network and their respective probabilities in the fitted model. Edges of primary interest are those that are present in the original network but have low probability in the fitted model (since the presence of these edges cannot be

explained by the model) or the opposite (since these edges are presumed to be existent if the model is a good description of the network).

In the following sections, I will introduce a model similar to that of Clauset et al. [22]. The difference is that the model presented here is not hierarchical; vertices are assigned to groups instead and the connection probability of the vertices are governed by their group affiliations and a predefined preference matrix.

2.2 The model framework

2.2.1 Formal description

I consider a generic non-growing network model where n vertices are initially assigned to one of k possible vertex types, denoted by integers from the interval $[1, k]$ [88, 89]. Let us assume that the vertices are indexed by integers from $[1, n]$. Let u_i denote the type of vertex i. Vertices are connected to each other randomly, but the connectional probability of vertices i and j is a function of u_i and u_j, determined by a preference matrix $\mathbf{P}$ which has k rows and k columns. More precisely, let there be a discrete random variable with probability distribution such that:

$$\mathbf{P}(T = i) = \begin{cases} t_i & i = 1, 2, \ldots, k \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

The t_i values are usually arranged in a column vector $\mathbf{T} = [t_1, t_2, \ldots, t_k]^T$. Moreover, let there be a matrix $\mathbf{P} = [p_{ij}]$ called the *preference matrix*, having k rows and k columns. The matrix must be symmetric if we are generating an undirected graph. A graph according to this preference model can be constructed as follows:

1. Consider an empty graph with n vertices.
2. Assign a vertex type to each vertex drawn from the discrete probability distribution defined by $\mathbf{T}$.
3. For each (directed or undirected) pair of vertices i and j, add an edge between them with probability p_{u_i,u_j}

Graphs generated by this model will be denoted by $G_{\text{Pref}}(n, k, \mathbf{t}, \mathbf{P})$ from now on. See Algorithm 2 for the formal description in pseudo-code.

One may note the similarity of Algorithms 1 and 2; in fact, the two algorithms are completely equivalent for $k = 1$, and the model simply generates

Algorithm 2 Generating graphs according to the preference model

Require: T, $n \geq 0$, $k \geq 1$ and $0 \leq p_{ij} \leq 1$ for all $1 \leq i, j \leq k$

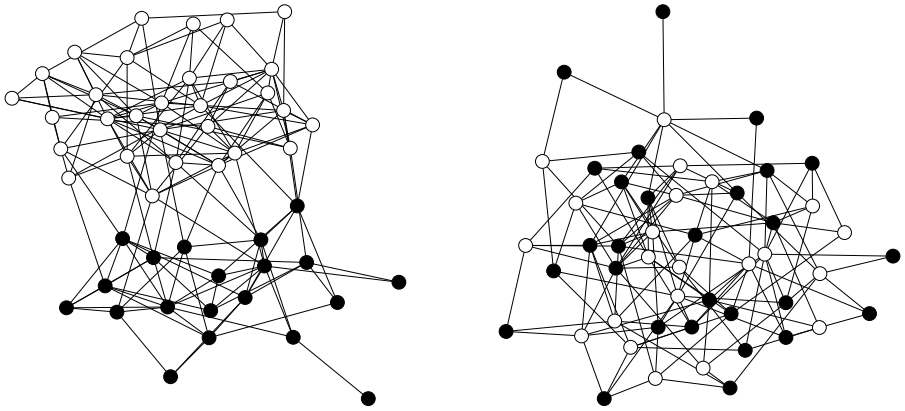
```
G := empty graph with n vertices
for i = 1 to n do
    u_i := random integer drawn from T
end for
for all {v_1, v_2} ∈ V(G) × V(G) do
    q := random number between 0 and 1, inclusive
    if q ≤ p_{u_i,u_j} then
        E(G) := E(G) ∪ {v_i, v_j}
    end if
end for
return G
```

an Erdős-Rényi graph according to the $G_{n,p}$ model in this case. However, things get more complicated when k is larger than 1, and as we will see, a wide variety of networks can be approximated after appropriate parameterisation of this model. Some example networks generated by the model are shown on Figure 2.1.

Relationship with Szemerédi's Regularity Lemma

Readers who are familiar with extremal graph theory may have noted the vague similarity of the preference model with the idea of ε-regular partitions and the famous regularity lemma of Endre Szemerédi [64, 120, 121]. Loosely speaking, the lemma states that the vertex set of every graph which is large enough can be partitioned into a given number of classes such that almost every pair of classes are ε-regular. A pair of classes V_i and V_j are said to be ε-regular if for all subsets $X \subseteq V_i, |X| \geq \varepsilon|V_i|$ and $Y \subseteq V_j, |Y| \geq \varepsilon|V_j|$, it holds that $|d(V_i, V_j) - d(X, Y)| < \varepsilon$, where $d(X, Y)$ is the number of edges connecting X and Y, divided by $|X||Y|$. For a more rigorous description of the lemma, see [120] or [64].

An intuitive explanation of the lemma can be given as follows: a graph that is large enough can be partitioned into subsets of vertices such that every vertex belongs to exactly one of the subsets and the edges between any two subsets can be described adequately with a single number: the edge density between the subsets. This holds due to the ε-regularity of the partition: given any sufficiently large subsets X and Y of vertex sets V_i and V_j, the actual edge density between X and Y does not differ too much from the edge density between V_i and V_j. The analogy with the preference model



(a) $p_{11} = p_{22} = 0.2,\ p_{12} = p_{21} = 0.02$ (b) $p_{11} = p_{22} = 0,\ p_{12} = p_{21} = 0.1$

FIGURE 2.1: A clustered (left) and a bipartite network (right) generated by the preference model. Colors correspond to vertex types, $n = 50$, $k = 2$, $t_1 = t_2 = 0.5$ in both cases.

is then straightforward: vertex types of the preference model correspond to the subsets defined by the partition whose existence is guaranteed by the regularity lemma, while the probability matrix (apart from its diagonal) describes the edge densities. The preference model can be thought of as an approximation of an ε-regular partition in finite and relatively small graphs, although this likeness should be handled with care. In particular, the term "*generalised random graph*" appearing in [64] denotes a graph model that is practically identical to the preference model described in Section 2.2.1.

Unfortunately, the lemma is an asymptotical and existential statement: it holds only for very large graphs, and it does not provide us with an algorithm that finds an ε-regular partition in an arbitrary graph.[1] However, me and my colleagues had the general idea of the regularity lemma (i.e. the probabilistic description of edges between vertex groups) in our minds when we first started considering the preference model as a tool for missing link prediction in complex networks.

2.2.2 Extended preference model

To allow for greater generality among the networks generated by this model, one can consider the following generalization of the preference model which always generates directed graphs and allows vertices to behave differently depending on their role in a hypothesized edge (source or target vertex):

[1] A constructive regularity lemma was given later in [5].

1. Consider an empty graph with n vertices.

2. Assign *outgoing vertex types* (or *out-types* in short) to each vertex drawn from the discrete probability distribution defined by $\mathbf{T}_- = [t_{-,1}, t_{-,2}, \ldots, t_{-,k}]$.

3. Assign *incoming vertex types* (or *in-types* in short) to each vertex drawn from the discrete probability distribution defined by $\mathbf{T}_+ = [t_{+,1}, t_{+,2}, \ldots, t_{+,k}]$.

4. For each directed pair of vertices i and j, draw an edge between them with probability $p_{\text{out-type}(i),\text{in-type}(j)}$.

The rationale behind this extension is that in directed networks, we usually can not neglect the assumption that the directionality of the edges does matter (see for example [70] where an excellent example is given to explain why the directionality of edges may matter in community detection). By assigning different in- and out-types to a vertex, the extended preference model is able to grasp substantial differences between the incoming and outgoing connectional preferences of that vertex. As I show later in Section 4.1, this can improve the overall fit of the model in case of directed networks while also providing important insights regarding the role of specific vertices.

2.3 Basic statistical properties

One of the first questions a network scientist may ask upon encountering a new network model is probably the following: what is the degree distribution of the network in the limit of infinite network size? This dissertation follows the same trail: first, I will analytically derive the limiting degree distribution, then I will present the results of numerical simulation and discuss the corollaries of the limit distribution.

2.3.1 Analytical results

Degree distribution

I start by studying the degree distribution of the extended preference model. First I note that line 7 in Algorithm 2 is a Bernoulli trial with success probability p_{u_i,u_j}. Therefore, the number of outgoing edges from a vertex of type i towards vertices of type j follows a binomial distribution $B(nt_j, p_{ij})$ [49]. Let this random variable be denoted by D_{ij}. The total number of edges

originating from a vertex of type i (that is, the out-degree, denoted by $D_{-,i}$) can then simply be expressed as follows:

$$D_{-,i} \sim \sum_{j=1}^{k} D_{ij} = \sum_{j=1}^{k} \mathrm{B}(nt_j, p_{ij}) \tag{2.3}$$

where k is the number of vertex types. Therefore, the out-degree of a vertex with type i is a random variable distributed according to a sum of binomially distributed random variables. The in-degree $D_{+,i}$ is completely identical, except for the order of i and j in the indices:

$$D_{+,i} \sim \sum_{j=1}^{k} D_{ji} = \sum_{j=1}^{k} \mathrm{B}(nt_i, p_{ji}) \tag{2.4}$$

According to the law of total probability, $d_{-,i}$ (the probability of the event that a random vertex has out-degree i) can be calculated as follows:

$$\begin{aligned} d_{-,i} &= \mathbf{P}(\text{a randomly chosen vertex has out-degree } i) \\ &= \sum_{j=1}^{k} \mathbf{P}(\text{a random vertex of type } j \text{ has out-degree } i) t_j \\ &= \sum_{j=1}^{k} \mathbf{P}(D_{-,j} = i) t_j \end{aligned} \tag{2.5}$$

Let $n \to \infty$ while keeping every $np_{ij} = z_{ij}$ constant. In the infinite limit, $B(nt_j, p_{ij})$ becomes a Poisson distribution with mean $nt_j p_{ij}$. Substituting it back to (2.3) yields:

$$D_{-,i} \sim \sum_{j=1}^{k} \mathrm{Pois}(nt_j p_{ij}) = \mathrm{Pois}\left(\sum_{j=1}^{k} t_j z_{ij} \right) = \mathrm{Pois}(\lambda_{-,i}) \tag{2.6}$$

since D_{ij}'s were independent. Therefore (2.5) can be rewritten as:

$$d_{-,i} = \sum_{j=1}^{k} \mathbf{P}(\mathrm{Pois}(\lambda_{-,j}) = i) t_j = \sum_{j=1}^{k} \frac{(\lambda_{-,j})^i}{i!} e^{-\lambda_{-,j}} t_j \tag{2.7}$$

Following a similar train of thought, (2.4) becomes:

$$d_{+,i} = \sum_{j=1}^{k} \mathbf{P}(\mathrm{Pois}(\lambda_{+,j}) = i) t_j = \sum_{j=1}^{k} \frac{(\lambda_{+,j})^i}{i!} e^{-\lambda_{+,j}} t_j \tag{2.8}$$

where $\lambda_{+,i} = \sum_{j=1}^{k} t_j z_{ji}$.

Note that this approximation is valid only in the infinite limit when the sum of the binomials tend to a Poisson distribution. In practice, the Poisson distribution can safely be used in place of a binomial when the probability of an individual success in the binomial is less than 0.05. Since I assumed that $np_{ij} = z_{ij}$ is constant, it follows that as n tends to infinity, p_{ij} tends to zero, therefore it will always attain the point at some n where the approximation becomes valid.

The degree distribution of the undirected case is derived exactly as I did above for the directed case, the only difference is that I do not have to consider incoming and outgoing edges separately. The degree distribution in the infinite limit is the following:

$$d_i = \sum_{j=1}^{k} \mathbf{P}(\mathrm{Pois}(\lambda_j) = i) t_j = \sum_{j=1}^{k} \frac{\lambda_j^i}{i!} e^{-\lambda_j} t_j \tag{2.9}$$

where $\lambda_i = \sum_{j=1}^{k} z_j p_{ij}$.

This leads to the following theorem:

Theorem 2.1. *Let $G_0, G_1, \ldots$ be an infinite sequence of directed graphs where G_i is generated according to $G_{\mathrm{Pref}}(n_i, k, \mathbf{t}, \mathbf{P}_i)$, assuming that $n_i \to \infty$ while $n_i \mathbf{P}_i = \mathbf{Z}_i$ is constant. The out- and in-degree distributions of G_∞ are then described by Eqs.* (2.7) *and* (2.8)*, respectively. If the graph is undirected, the degree distribution is according to Eq.* (2.9)*.*

To simplify the formulae, let $\lambda = [\lambda_1, \lambda_2, \ldots, \lambda_k]^T$, $\lambda_- = [\lambda_{-,1}, \lambda_{-,2}, \ldots, \lambda_{-,k}]^T$ and $\lambda_+ = [\lambda_{+,1}, \lambda_{+,2}, \ldots, \lambda_{+,k}]^T$. This enables us to express the λ values in a simple and concise vector equation, since $\lambda_+ = \mathbf{Z}^T\mathbf{t}$ and $\lambda_- = \mathbf{Z}\mathbf{t}$. λ can be expressed both ways, for $\mathbf{Z}$ is symmetric for undirected graphs.

The next step is to calculate the average number of edges adjacent to a randomly chosen vertex in G_∞.

Theorem 2.2. *The average number of edges adjacent to a randomly chosen vertex in G_∞ is $\mathbf{t}^T\mathbf{Z}\mathbf{t}$. In the directed case, this is the average number of both the incoming and outgoing edges.*

Proof. The average number of adjacent edges is simply the expected value of D, the random variable denoting the degree of an individual vertex:

$$\begin{aligned} \mathbf{E}(D) = \sum_{i=0}^{\infty} i d_i = \sum_{i=0}^{\infty} i \left(\sum_{j=1}^{k} \frac{\lambda_j^i}{i!} e^{-\lambda_j} t_j \right) = \sum_{j=1}^{k} t_j \sum_{i=0}^{\infty} i \frac{\lambda_j^i}{i!} e^{-\lambda_j} = \\ \sum_{j=1}^{k} t_j \mathbf{E}(\mathrm{Pois}(\lambda_j)) = \sum_{j=1}^{k} t_j \lambda_j = \mathbf{t}^T\lambda = \mathbf{t}^T\mathbf{Z}\mathbf{t} \end{aligned} \tag{2.10}$$

The procedure is the same for the expected value of D_+ or D_- and it immediately follows that $\mathbf{E}(D_+) = \mathbf{E}(D_-)$: the expected number of incoming and outgoing edges of a vertex is equal, even though their distribution is different. □

Naturally, all the results presented here apply only for the average behaviour of large networks. The reason is twofold: first, it is not really meaningful to talk about the degree distribution of a network with only a few thousand vertices (especially if we only have a single instance of the network), second, the approximation of binomial distributions with Poisson distributions hold only if $n \to \infty$.

A simple corollary: the giant component

In this subsection, I only examine the undirected preference model.

This model, like many others, exhibits an interesting phenomenon as we gradually increase the connection probabilities in the preference matrix. In an Erdős-Rényi random graph, the average component size in the case of $np < 1$ is finite, while a so-called *giant component* emerges when $np > 1$. The size of this component is in the order of the number of vertices in the whole graph, meaning that as the number of vertices in the network approach infinity, so does the size of the giant component (but not the others). $np = 1$ is called the *critical point* where this phase transition occurs and the initially disconnected small components join to form the giant component. The corollary is that this type of phase transition will also occur in the preference model, since the subgraph spanned by vertices of type i is always an Erdős-Rényi graph with nt_i vertices and connection probability p_{ii}. I will show that a similar threshold exists for subgraphs consisting of edges between vertices of type i and j as well and derive a sufficient (but by no means necessary) condition for the existence of a giant component.

Theorem 2.3. *G_∞ contains a giant component if* $\max_{i,j} np_{ij}\sqrt{t_i t_j} > 1$.

Proof. I distinguish two cases:

1. There exists i such that $nt_i p_{ii} > 1$. Consider the subgraph consisting of the vertices of type i and the edges that originate from and terminate in those vertices. This subgraph is an Erdős-Rényi graph with nt_i vertices. $nt_i p_{ii} > 1$ implies that the subgraph contains a giant component, therefore G_∞ also contains a giant component.

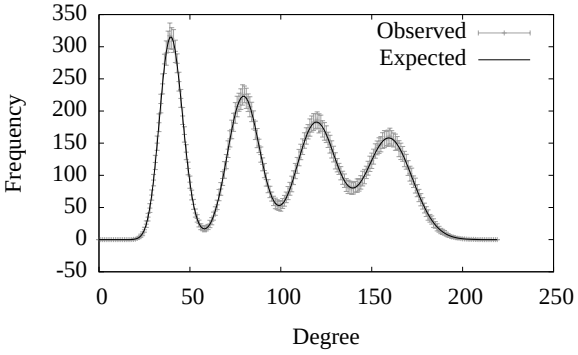
2. There exists i and j such that $i \neq j$ and $n\sqrt{t_i t_j} p_{ij} > 1$. Consider the subgraph that consists of vertices of type i and j and the edges that

go between these vertices and their endpoints correspond to different vertex types. This is a random bipartite graph where pairs of vertices are connected with probability p_{ij}. Let us introduce a branching process on the vertices of the subgraph, starting from a randomly chosen vertex (generation 0). The first generation of the branching process contains the first-order neighbours of the root vertex, the second generation contains the second-order neighbours and so on. Assume that the root vertex is of type i and let m_x denote the number of vertices up to and including generation x. Note that all children of a vertex of type i will be of type j and vice versa. The expected number of children of a vertex in generation x follows a binomial distribution $B(nt_j - m_x, p_{ij})$ if x is odd and $B(nt_i - m_x, p_{ij})$ if x is even, since we are not interested in edges that link back to earlier generations. When n is large compared to m_x, these distributions can be approximated with $B(nt_j, p_{ij})$ and $B(nt_i, p_{ij})$, respectively. The expected number of children is therefore $nt_j p_{ij}$ in odd generations and $nt_i p_{ij}$ in even generations. To obtain a branching process whose branching distribution is equal for all generations, generation $2k$ can be merged with generation $2k + 1$ for all $k \geq 0$. The expected number of children in each generation in the merged process is $n^2 p_{ij}^2 t_i t_j$. The branching process terminates in ultimate extinction with probability 1 when $n^2 p_{ij}^2 t_i t_j < 1$, resulting in a finite component that does not scale as n while n tends to infinity. On the other hand, $np_{ij}\sqrt{t_i t_j} > 1$ implies the existence of a giant component in this specific subgraph, therefore G_∞ also contains a giant component. □

It can not be stressed enough that the conditions presented above are sufficient but not necessary for the existence of a giant component. As an example, one can consider a preference matrix where the elements in the main diagonal are only slightly below the critical threshold and all other elements of the preference matrix are zeros. This graph does not contain a giant component. However, only a slight increase in the off-diagonal entries is usually sufficient to connect the otherwise disconnected subgraphs and form a component whose size is comparable to the size of the whole network.

2.3.2 Numerical simulations

To test the validity of the theoretical degree distribution presented above, I conducted numerical simulations using the `igraph` library [24]. Instances of the preference model were generated with $n = 2 \times 10^4$ vertices. k was drawn from $U(2, 20)$, the uniform distribution. Vertex type probabilities were cho-



(a) Degree distribution

$$\mathbf{P} = \begin{bmatrix} 2 & 2 & 2 & 2 \\ 2 & 2 & 6 & 6 \\ 2 & 6 & 4 & 12 \\ 2 & 6 & 12 & 12 \end{bmatrix} \times 10^{-3}$$

$$n = 2 \times 10^4, k = 4$$

$$t_i = 0.25$$

$$\lambda = [40, 80, 120, 160]$$

(b) Parameters

FIGURE 2.2: Expected and observed degree distribution of a random graph according to the undirected preference model

sen from a k-dimensional Dirichlet distribution with parameters $(1, 1, \ldots, 1)$, Elements of the preference matrix were drawn from $U(0, m)$, $m \leq 1$. m was chosen appropriately to obtain a graph that fits entirely in RAM (typically, m had to be lowered for $n \gg 10^4$). The in- and out-degree distributions of the generated graphs were averaged over 100 instantiations using the same parameter set and then compared to the expected distributions as derived in Section 2.3.1 using the χ^2 goodness of fit statistic.[2] P-values were larger than 0.98 in all cases.

Results of a single test run are shown on Figure 2.2. For this specific example, the value of the χ^2 statistic was 7.9148 with 170 degrees of freedom, resulting in $p = 1.000$. Although the χ^2 test in general is only used to rule out a specific null hypothesis, the almost perfect match between the observed and expected distributions (see Figure 2.2) and the high p-value together confirm the analytical calculations.

2.4 Fitting the model to data

We invent models not only for their own beauty. Good models are simplified descriptions of real-world phenomena; simple enough to be studied, yet complex enough to help us gain new insights into the driving forces behind a specific phenomenon. The first step in this trail is to fit the model to some dataset we obtained while studying the phenomenon. Since I examine a graph model, I assume that this data can readily be converted into a

[2]When applying the χ^2 test, neighboring bins with less than 10 observations were joined together to justify the χ^2-approximation used in this test.

directed or undirected graph.

In this section, the real-world dataset is denoted by $G_0(V_0, E_0)$, while $G(V, E)$ denotes the graph of the fitted extended preference model. The dataset can be imprecise in the sense that the existence of some edges may be unknown (but the number of vertices is always known in advance). The elements of the *belief matrix* $\mathbf{B} = [b_{ij}]$ contain our degree of belief in the existence of the edges: $b_{ij} = 0$ if we are absolutely certain that the edge from vertex i to j is nonexistent and $b_{ij} = 1$ if we know for sure that the edge exists. Intermittent b_{ij} values describe uncertainties, so from a probabilistic point of view, b_{ij} values are the *a priori* probabilities of the edges. It is also possible that there are some unknown edges for which we cannot come up with a reasonable *a priori* probability due to lack of information. In this case it is safe to set b_{ij} to $1/2$, for it will have the same effect on the goal function (see Section 2.4.1) as the complete exclusion of that edge, apart from a constant offset in the value of the goal function. Note that a binary $\mathbf{B}$ matrix consisting entirely of zeros and ones, which occurs when there are no uncertain parts in the data, coincides with the adjacency matrix of the graph we are studying.

Parameters of the fitted model will be denoted by n (the number of vertices), k (the number of in- and out-groups) and $\mathbf{P}$ (the preference matrix). To allow for reasoning about the properties of the individual vertices, we will be interested not only in the type distribution of the model, but in the actual type assignments of the vertices as well. Therefore, the type assignments u_i and v_i (or simply $\mathbf{u}$ and $\mathbf{v}$ in vector form) will be parameters in the fitting process instead of the type distribution $\mathbf{T}_-$ and $\mathbf{T}_+$, respectively. n does not have to be estimated, for the fitted model must contain exactly the same number of vertices as the dataset. The set of parameters to be estimated will be denoted by $\theta = (k, \mathbf{u}, \mathbf{v}, \mathbf{P})$.

The fitting methods described here work perfectly with the simple preference model as well with some minor modifications. The methods are iterative, and the value of the parameters or the graph itself in the i-th iteration is denoted by an upper index $^{(i)}$ (e.g., $G^{(i)}$, $\mathbf{P}^{(i)}$ and so on).

2.4.1 The goal function for model fitting

No matter what specific algorithm we choose for fitting the preference model, we need a measure that quantifies the fitness of a particular model parameterisation θ with respect to the data being fitted. The measure I will use from now on as a goal function is the likelihood of the model parameterisation, given the known data. Informally, the likelihood of the parameterisation is the probability of the event that the parameterised model generates the

dataset. In case of the extended preference model, the likelihood is given by:

$$\mathcal{L}(\theta|G_0) = \prod_{i=1}^{n} \prod_{\substack{j=1 \\ j\neq i}}^{n} \left(b_{ij} p_{u_i,v_j} + (1 - b_{ij})(1 - p_{u_i,v_j}) \right) \tag{2.11}$$

It is straightforward that $b_{ij} = 1/2$ implies that the particular term in the likelihood corresponding to b_{ij} will remain constant, irrespective of u_i and v_j, since $p_{u_i,v_j}/2 + (1 - p_{u_i,v_j})/2 = 1/2$.

There exists an equivalent but less intuitive form of the goal function. Let B denote the set of all b_{ij}'s used in the current network: $B = \cup_{i=1,j=1}^{n,n} b_{ij}$. Let $E_{u,v,b}$ be the number of vertex pairs (i, j) where vertex i belongs to out-group u, vertex j belongs to in-group v and $b_{ij} = b$. The likelihood is then as follows:

$$\mathcal{L}(\theta|G_0) = \prod_{u=1}^{k} \prod_{v=1}^{k} \prod_{b\in B} \left(b p_{uv} + (1 - b)(1 - p_{uv}) \right)^{|E_{u,v,b}|} \tag{2.12}$$

The equivalence of Eqs. (2.11) and (2.12) follows from the proper rearrangement of terms in Eq. (2.11): since the number of different p_{u_i,v_j}'s is only k^2 (which is significantly less than n^2), the product can be calculated by iterating over the possible b_{ij} and p_{u_i,v_j} combinations instead of iterating over all vertex pairs. This form will be of use to us in the efficient implementation of the fitting algorithms. For theoretical calculations, I will use the first form, as it is easier to understand and to work with.

In practice, the logarithm of the likelihood function is used instead of the likelihood. Maximising the likelihood is the same as maximising the log-likelihood (since the logarithm function is strictly increasing), but numeric calculations involving the log-likelihood do not suffer from accumulated floating point errors as much as direct likelihood calculations. The log-likelihood is given by:

$$\log \mathcal{L}(\theta|G_0) = \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j\neq i}}^{n} \log \left(b_{ij} p_{u_i,v_j} + (1 - b_{ij})(1 - p_{u_i,v_j}) \right) \tag{2.13}$$

or equivalently in the alternative form:

$$\log \mathcal{L}(\theta|G_0) = \sum_{u=1}^{k} \sum_{v=1}^{k} \sum_{b\in B} |E_{u,v,b}| \log \left(b_{ij} p_{u_i,v_j} + (1 - b_{ij})(1 - p_{u_i,v_j}) \right) \tag{2.14}$$

The idea of the likelihood function as a tool for statistical reasoning dates back to the paper of Fisher [40], where the method of maximum likelihood is

also devised. I also strive for maximising the likelihood, assuming that the parameterisation that produces the maximum likelihood is a good description of the known dataset in terms of the extended preference model. However, the process is more complicated here: the number of parameters in θ depends on k, and the maximisation of the likelihood can only be done properly by choosing k first. I will briefly assume that k is known and describe two methods that can be used to maximise the likelihood. Finally, I return to the problem of choosing the proper k in Section 2.4.5.

2.4.2 Fitting by expectation-maximisation

The algorithm presented here is a variant of the expectation-maximisation (EM) algorithm scheme [28]. EM algorithms alternate between two steps (the expectation (E) and the maximisation (M) step) in order to maximise the likelihood function. However, there is no guarantee of converging to the maximum likelihood estimator. In some cases, the EM algorithm gets stuck in a local maximum, depending on the starting values. Local maxima can be ruled out by either restarting the algorithm from different starting points or randomly mutating the parameter set in the local maximum in order to proceed further.

Starting the algorithm

The algorithm starts in a state where n and k are given in advance, $\mathbf{u}$ and $\mathbf{v}$ are chosen randomly (every vertex is assigned to one of k possible in- and out-types) and $\mathbf{P}$ is unspecified yet. The iteration counter t is set to zero. Theoretically it is possible to assign the vertex types based on some *a priori* analysis (e.g., according to the community structure of the network as detected by some fast heuristic algorithm [21, 69]), but in practice, the convergence speed of the algorithm is not influenced by the starting state significantly, so the time spent on calculating a suitable starting state may as well be spent on the actual optimisation of the likelihood function.

The E step

The E step involves calculating the conditional expectation of the log-likelihood function by considering $\mathbf{u}$ and $\mathbf{v}$ fixed, given the observed graph:

$$\begin{aligned}\mathbf{E}(\log \mathcal{L}(\theta^{(t)}|G_0)) &= \sum_{i=1}^{n}\sum_{\substack{j=1\\i\neq j}}^{n} \mathbf{E}\left(\log\left(b_{ij}p^{(t)}_{u_i,v_j} + (1-b_{ij})\left(1-p^{(t)}_{u_i,v_j}\right)\right)|G_0\right) \\ &= \sum_{i=1}^{n}\sum_{\substack{j=1\\i\neq j}}^{n} \log\left(b_{ij}\mathbf{E}(p^{(t)}_{u_i,v_j}|G_0) + (1-b_{ij})\left(1-\mathbf{E}(p^{(t)}_{u_i,v_j}|G_0)\right)\right)\end{aligned} \tag{2.15}$$

$\mathbf{E}(p_{kl}|G_0)$ above is the expected probability of the presence of an edge between vertices of out-type k and vertices of in-type l, given the observed graph. Since the number of edges between any two groups is binomially distributed (the existence of each edge is decided by an independent Bernoulli trial with probability p_{kl}), the expected value of p_{kl} is simply the number of actual edges going from out-type k to in-type l, divided by the number of all possible edges. Uncertain edges are taken into account according to their degrees of belief, resulting in the following estimation of p_{kl}:

$$p^{(t)}_{kl} = \mathbf{E}(p_{kl}|G_0, \mathbf{u}^{(t)}_{+}, \mathbf{v}^{(t)}_{-}) = \frac{\sum_{i=1,j=1}^{n,n}(1-\delta(i,j))\delta(u^{(t)}_i,k)\delta(v^{(t)}_j,l)b_{ij}}{\sum_{i=1,j=1}^{n,n}(1-\delta(i,j))\delta(u^{(t)}_i,k)\delta(v^{(t)}_j,l)} \tag{2.16}$$

where $\delta(i,j)$ is the Kronecker delta function: $\delta(i,j) = 1$ if and only if $i = j$, 0 otherwise). After the E step, the iteration counter t is increased by one.

The M step

The M step involves the maximisation of the log-likelihood function using the preference matrix $\mathbf{P}^{(t-1)}$ obtained in the previous E step. The new $\mathbf{u}^{(t)}$ and $\mathbf{v}^{(t)}$ vectors should be chosen from the space of all possible type assignments in a way that maximises the likelihood. The space contains k^{2n} possible assignments, which is exponential in n, rendering the exhaustive search of the state space practically impossible when n is large. However, Neal and Hinton [85] showed that it is not necessary to maximise the log-likelihood in the M step, even *some* improvement over the previous log-likelihood is sufficient for the algorithm to converge.

The improvement of the log-likelihood in this step is achieved by a distributed decision mechanism performed by each vertex. Every vertex observes the current type assignments of other vertices and chooses its own type assignment in a way that maximises its own contribution to the total log-likelihood under the assumption that all other vertices retain their type assignments. This assumption does not hold of course, since all vertices decide about their own type assignment simultaneously. The new type assignment is stored in $\mathbf{u}^{(t)}$ and $\mathbf{v}^{(t)}$. To decide whether the log-likelihood really improved, an E step has to be performed again. The algorithm stops when it is not possible to improve the log-likelihood any more, which can be detected by the fact that $\mathbf{u}_-^{(t)} = \mathbf{u}_-^{(t-1)}$ and $\mathbf{v}_+^{(t)} = \mathbf{v}_+^{(t-1)}$.

Termination and evaluation

After several iterations of the E and M steps, the algorithm reaches a consensus situation. This consensus is characterised by the fact that it is not possible to improve the log-likelihood by changing the type assignment of any single vertex. (Remember: all vertices base their decision in the M step on the assumption that no other vertex will change its type assignment!). Theoretically, it is possible to improve the value of the goal function by changing several type assignments simultaneously, but evaluating all such possibilities is computationally expensive. It is therefore suggested to mutate some small portion of the type assignments and restart the algorithm from the E step a couple of times in order to climb out from possible local maxima while keeping track of the best parameterisation encountered so far. The algorithm can safely be terminated when it ends up at the same or worse configurations even after m mutations (m is given in advance). See Algorithm 3 for a possible pseudo-code description of the procedure.

Finally I note that since the actual type indices assigned to the vertices have no specific meaning, they can be rearranged as one wishes by swapping appropriate rows or columns of the preference matrix and reassigning the out- and in-types of the vertices appropriately. For example, one can swap row 2 and row 4 in the preference matrix if all vertices of out-type 2 are reassigned to out-type 4 and vertices of out-type 4 are reassigned to out-type 2 simultaneously. The rearrangement does not contribute anything to the result in a strict mathematical sense, but some configurations may be easier to interpret for humans than others. I found two conventions particularly useful:

C1. Let $\phi(i)$ denote the smallest vertex index among vertices having out-type i. It is desirable to reassign the out-types appropriately to ensure that $1 = \phi(1) < \phi(2) < \cdots < \phi(k)$.

Algorithm 3 Fitting the preference model by expectation-maximisation

Require: $G(V,E), |V| = n, k > 0, m_0 > 0$
$t := 0$, $best :=$ null
Choose the elements of $\mathbf{u}^{(0)}$ and $\mathbf{v}^{(0)}$ randomly from the range $(1;k)$
$m := m_0$
while $m > 0$ **do**
  while $t = 0 \vee \mathbf{u}^{(t)} \neq \mathbf{u}^{(t-1)} \vee \mathbf{v}^{(t)} \neq \mathbf{v}^{(t-1)}$ **do**
    Calculate $\mathbf{P}^{(t)}$ according to the E step, Eq. (2.16)
    if $best =$ null $\vee \log \mathcal{L}(\theta^{(t)}) > \log \mathcal{L}(\theta^{(best)})$ **then**
      $best := t$, $m := m_0$
    end if
    $t := t + 1$
    Calculate $\mathbf{u}^{(t)}$ and $\mathbf{v}^{(t)}$ according to the M step
  end while
  $m := m - 1$
  Mutate $\mathbf{u}^{(t)}$ and $\mathbf{v}^{(t)}$ randomly
end while
return $\theta^{(best)}$

C2. It is helpful to rearrange the matrix in a way that maximises the number of vertices that have the same in- and out-type indices. This is especially useful when visualising the results, since vertices can be coloured according to their in- and out-types, and this way usually only a small fraction of the vertices has to be coloured by two different colours. These multi-coloured vertices are usually of particular interest due to their unusual connection patterns.

A type assignment is said to be canonical when it satisfies these conventions: $\phi(i)$ is in ascending order and the number of vertices with the same type indices is maximal. The rearrangement can be done according to Algorithm 4.

2.4.3 Fitting by Markov chain Monte Carlo methods

The EM method described in Section 2.4.2 has some drawbacks. The most important disadvantage is that the solution obtained is a local optimum which might be far from the global one at times. Another flaw is the problem of overfitting. Consider a situation where the log-likelihood is almost the same for two given parameterisations θ_1 and θ_2 that differ only in the type assignment of a single vertex. Without loss of generality, let us assume

Algorithm 4 Canonical rearrangement of vertex type assignments

Let $f :=$ zero vector of length k
for $i := 1$ to n **do** {Calculating $\phi(i)$}
  if $f[u_i] = 0$ **then**
    $f[u_i] := i$
  end if
end for
Rearrange out-types according to f (out-type i becomes $f[i]$)
Permute the rows of $\mathbf{P}$ according to f {Now C1. is satisfied}
Let $M :=$ zero matrix of size $k \times k$
for $i := 1$ to n **do** {Count the number of different type configurations}
  $M[u_i, v_i] := M[u_i, v_i] + 1$
end for
while $\max M > 0$ **do**
  $i, j := \text{argmax}_{ij} M$
  Swap in-types i and j
  Swap column i and j in $\mathbf{P}$ and $\mathbf{M}$
  Fill row i and column i with -1 in $\mathbf{M}$
end while{Now C2. is also satisfied}
return $\mathbf{u}, \mathbf{v}, \mathbf{P}$

that $\log \mathcal{L}(\theta_1|G_0) = \log \mathcal{L}(\theta_2|G_0) + \varepsilon$, where ε is small. In this case, the EM algorithm will return θ_1 if that is the optimal parameterisation, without ever mentioning that θ_2 is almost as successful in describing the data as θ_1, suggesting that the type assignment of that single vertex is not relevant. The Markov chain Monte Carlo (MCMC) method described in this section aims to overcome these disadvantages at the expense of increased computational complexity.

Generally, MCMC methods are a class of algorithms to draw samples from a probability distribution that is hard to be sampled from directly. These methods generate a Markov chain whose equilibrium distribution is equivalent to the distribution we are trying to sample from. In the case of the extended preference model, the samples are parameterisations of the model, and the distribution we are sampling from is the following:

$$\mathbf{P}(\Theta = \theta_0) = \frac{\mathcal{L}(\theta_0|G_0)}{\displaystyle\int_{\mathbf{S}_{n,k}} \mathcal{L}(\theta|G_0)\,\mathbf{d}\theta} \tag{2.17}$$

where $\mathbf{S}_{n,k}$ is the space of all possible parameterisations of the probability model for given n and k. Informally, the probability of drawing θ as a sample should be proportional to its likelihood of generating G_0. For instance, if θ_1 generates the studied network with a probability of 0.5 and θ_2 generates it with a probability of 0.25, θ_1 should be drawn twice as frequently as θ_2.

The generic framework of the MCMC method I use is laid down in the Metropolis–Hastings algorithm [52]. The only requirement of the algorithm is that a function proportional to the density function (that is, $\mathbf{P}(\Theta = \theta_0)$ in (2.17)) can be calculated. Note that $\mathbf{P}(\Theta = \theta_0) \propto \mathcal{L}(\theta_0|G_0)$, since the denominator in (2.17) is constant, therefore the likelihood function satisfies the criteria of the Metropolis–Hastings algorithm. Starting from an arbitrary random parameterisation $\theta^{(0)}$, MCMC methods propose a new parameterisation θ' based on the previous parameterisation $\theta^{(t)}$ using some proposal density function $Q(\theta'|\theta^{(t)})$. If the proposal density function is symmetric ($Q(\theta'|\theta^{(t)}) = Q(\theta^{(t)}|\theta')$), the probability of accepting the proposed parameterisation is given by:

$$\mathbf{P}(\theta^{(t+1)} = \theta'|\theta^{(t)}) = \min\left(1, \frac{\mathcal{L}(\theta'|G_0)}{\mathcal{L}(\theta^{(t)}|G_0)}\right) = \min\left(1, e^{\log \mathcal{L}(\theta'|G_0) - \log \mathcal{L}(\theta^{(t)}|G_0)}\right) \tag{2.18}$$

When the proposal is accepted, it becomes the next state in the Markov chain ($\theta^{(t+1)} = \theta'$); otherwise the current state is retained ($\theta^{(t+1)} = \theta^{(t)}$).

MCMC sampling can only approximate the target distribution, since there is a residual effect depending on the starting position of the Markov

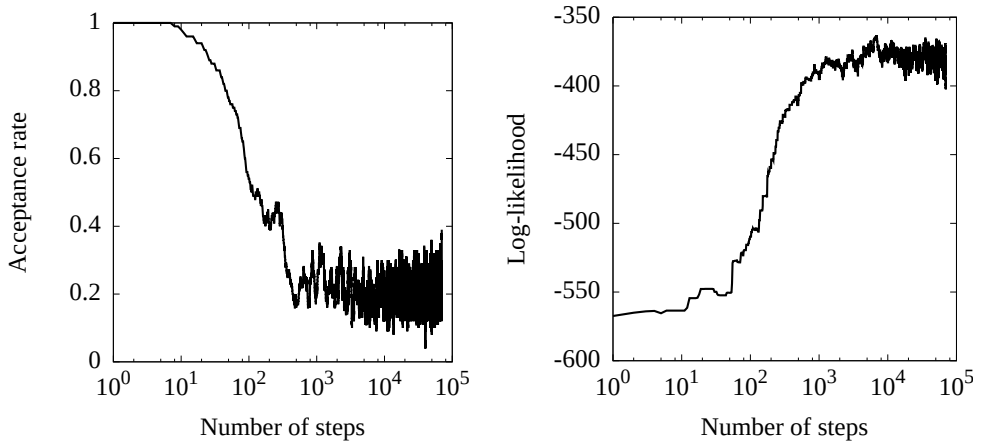


FIGURE 2.3: Acceptance rates (left) and log-likelihoods (right) during a typical run of the MCMC algorithm in the function of time.

chain. Therefore, the sampling consists of two phases. In the first phase (called *burn-in*), the algorithm is run for many iterations until the residual effect diminishes. The second phase is the actual sampling. The burn-in phase must be run long enough so that the residual effects of the starting position become negligible. A particular MCMC sampling implementation is said to mix rapidly if its required burn-in phase is short and it converges quickly to the equilibrium distribution. This will be discussed in detail later.

It is hard to decide when to finish the burn-in phase and start the actual sampling. The most common approach involves calculating the acceptance rate α, which is the fraction of proposals accepted during the last m steps in the burn-in phase. Sampling is started when the acceptance rate drops below a given threshold $\hat{\alpha}$. Local maxima are avoided by accepting parameterisation proposals with a certain probability even when they have a lower likelihood than the last one, but being biased at the same time towards parameterisations with high likelihood. In the case of multiple local maxima with approximately same likelihoods, MCMC sampling tends to oscillate between those local maxima. These concepts are illustrated on Figure 2.3. By taking a large sample from the equilibrium distribution, one can approximate the probability of vertex i being in out-group k and in-group l and extract the common features of all local maxima (vertices that tend to stay in the same groups despite randomly walking around in the parameter space).

The only thing left up to us before employing MCMC sampling on fitting the preference model is to define an appropriate symmetric proposal density function. I note that the number of groups k is constant and the preference matrix $\mathbf{P}$ can be approximated by the edge densities for a given out- and

in-group assignment, leaving us with only $2n$ parameters that have to be determined by the proposal density function: practically, the proposal density function should propose new group assignment vectors $\mathbf{u}$ and $\mathbf{v}$ based on the actual ones.

The simplest proposal density function one can conceive is a uniform distribution over all possible group assignments. This function yields weak performance, since it completely ignores the current group assignment in the Markov chain, which might already contain subsets of vertices that are assigned consistently to each other. Taking this into account, it is straightforward that the newly proposed group assignment should not differ too much from the current group assignment in order to keep the satisfactory parts of the group assignment vectors with high probability. Therefore, a simple but powerful proposal density function should change the group affiliation of only a few vertices. Simulations showed that even a single point mutation in every step is usually satisfactory in practical applications.

A sophisticated proposal density function that can be useful in several cases is based on Gibbs sampling [45], which depends on the conditional distributions of the parameters (that is, the components of the group membership vectors). Here I take advantage of the fact that the conditional distribution of each parameter (assuming the others are known) can be calculated exactly as follows:

$$\begin{aligned} \mathbf{P}(u_i = j) &= \frac{\mathcal{L}(\theta'_{u_i=j}|G)}{\sum_{l=1}^{k} \mathcal{L}(\theta'_{u_i=l}|G)} \\ \mathbf{P}(v_i = j) &= \frac{\mathcal{L}(\theta'_{v_i=j}|G)}{\sum_{l=1}^{k} \mathcal{L}(\theta'_{v_i=l}|G)} \end{aligned} \tag{2.19}$$

where $\theta'_{u_i=j}$ is a parameterisation that equals the current parameterisation $\theta^{(t)}$ with the exception of u_i which equals j. Similarly, $\theta'_{v_i=j}$ is the parameterisation derived from $\theta^{(t)}$ by letting $v_i = j$. In practice, the fractions can be simplified by the terms that do not involve vertex i at all (since they are present both in the numerator and the denominator), therefore it is enough to calculate the local contribution of the vertex to the log-likelihood for all in-types and out-types in order to determine the marginal distributions.

Since the marginal distribution of each parameter is known, Gibbs sampling [45] can be used. Gibbs sampling alters a single variable of the parameter vector in each step according to its marginal distribution, given all other parameters. It can be shown that the proposal distribution defined this way is symmetric if the variable being modified is picked randomly according to a uniform distribution [44]. In practice, it is sufficient to cycle through the

variables in a predefined order as long as the Markov chain can access all states under this ordering.

The pseudo-code description of the MCMC algorithm used for fitting the preference model is shown in Algorithm 5. m is the window size used to calculate the acceptance rate, $\hat{\alpha}$ is the acceptance rate threshold under which I start the actual sampling, N is the number of samples to be taken. This particular implementation simply returns the best sample from the Markov chain, but of course all the samples could have been stored and used later in more rigorous analyses.

Algorithm 5 Fitting the preference model by the Metropolis–Hastings algorithm

Require: $G(V, E), |V| = n, k > 0, N > 0, m \geq 1, 0 < \hat{\alpha} \leq 1$

$t := 0$, $best :=$ null, $sampling :=$ false
$accepts := [1, 1, \ldots, 1]$, indexed from zero
Choose the elements of $\mathbf{u}^{(0)}$ and $\mathbf{v}^{(0)}$ randomly from the range $(1; k)$
while $N > 0$ **do**
  Calculate $P^{(t)}$ according to Eq. (2.16)
  Draw θ' from the proposal density function $Q(\theta'|\theta^{(t)})$
  $p :=$ random number drawn from $[0; 1)$
  if $p < \exp\left(\log \mathcal{L}(\theta'|G) - \log \mathcal{L}(\theta^{(t)}|G)\right)$ **then**
    $\theta^{(t+1)} = \theta'$
    $accepts\,[t \mod m] := 1$
  else
    $\theta^{(t+1)} = \theta^{(t)}$
    $accepts\,[t \mod m] := 0$
  end if
  $t := t + 1$
  if *sampling* **then**
    $N := N - 1$
    if $\log \mathcal{L}(\theta^{(t)}|G) > \log \mathcal{L}(best|G)$ **then**
      $best := t$
    end if
  else if $\sum accepts < \hat{\alpha} m$ **then**
    $sampling :=$ true
  end if
end while
return $\theta^{(best)}$

Does the chain mix rapidly?

A desirable property of a Markov chain in a MCMC method is *rapid mixing*. A Markov chain is said to mix rapidly if its mixing time grows at most polynomially fast in the logarithm of the number of possible states in the chain. Mixing time refers to a given formalisation of the following idea: how many steps do we have to take in the Markov chain to be sure that the distribution of states after these steps is close enough to the stationary distribution of the chain? Given a guaranteed short mixing time, one can safely decide to stop the burn-in phase and start the actual sampling after the number of steps taken exceeded the mixing time of the chain. The number of possible states in the Markov chain of the preference model equals k^{2n}, the number of different group configurations. If the chain possesses the rapid mixing property, the mixing time must therefore be polynomial in $2n \log k$.

Several definitions exist for the mixing time of a Markov chain (for an overview, see [83]). To illustrate the concept, I refer to a particular variant called *total variation distance mixing time*, which is defined as follows:

Definition 2.1 (Total variation distance mixing time)**.** *Let S denote the set of states of a Markov chain $\mathcal{M}$, let $A \subseteq S$ be an arbitrary nonempty subset of the state set, let $\pi(A)$ be the probability of A in the stationary distribution of $\mathcal{M}$, and $\pi_t(A)$ be the probability of A in the distribution observed after step t. The total variation distance mixing time of $\mathcal{M}$ is the smallest t such that $|\pi_t(A) - \pi(A)| \leq 1/4$ for all $A \subseteq S$ and all initial states.*

However, many practical problems have resisted rigorous theoretical analysis. This applies also to the method presented here, mostly due to the fact that the state transition matrix of the Markov chain (and therefore its stationary distribution) is a complicated function of the adjacency matrix of the network and the number of vertex groups, and no closed form description exists for either. Therefore, the rapid mixing property of the Markov chain will be examined by simulations later in Section 2.6.3.

2.4.4 Combining EM and MCMC methods

So far I presented two algorithms for fitting the preference model to a given network. Both of them are able to give a solution, but they have drawbacks as well. The EM method is prone to overfitting (see the introduction in Section 2.4.3) and to get stuck in local minima. The MCMC method requires a relatively long burn-in process before one can actually start sampling. It is natural to ask in such a situation whether the two methods can be combined to complement each other. In the present case, the answer is yes: one

can apply the EM method described in Section 2.4.2 to speed up the burn-in process and revert to the Metropolis–Hastings algorithm when the EM algorithm reached the first local maximum. According to tests on random graphs generated by the preference model, the likelihood observed when the MCMC algorithm starts sampling with an acceptance rate $\alpha = 0.15$ is in the same order as the first local maximum obtained by the EM algorithm; in many cases, the likelihood obtained by the EM algorithm is better. I averaged the likelihoods over 100 instances of a randomly generated network with 256 vertices, and 4 vertex types. Vertices were connected to each other with probability 0.3 if they belonged to the same group and with probability 0.1 otherwise. The average of the observed log-likelihoods at the first minimum of the EM algorithm was -26003.46 ± 250.58, while the first sample of the MCMC algorithm had log-likelihood -27181.07 ± 168.9. The two log-likelihoods are approximately of the same magnitude, but the EM algorithm achieves this much faster than the MCMC sampling.

2.4.5 Choosing the number of vertex types

The algorithms discussed so far rely on the fact that k (the number of types) is given in advance. Of course this is not true in most practical applications, the number of types is unknown and has to be determined in order to achieve meaningful results.

A peculiar feature of the preference model is that one can always achieve a perfect fit of the model by letting the number of vertex types equal to the number of vertices. In this case, the obtained preference matrix is identical to the original belief matrix $\mathbf{B}$ and we learned nothing new about the underlying structure of the graph: the type assignments have no specific meaning, and moreover, we can not predict any missing edges in the network in case of initial uncertainty in the belief matrix. Therefore, a delicate balance has to be maintained between the accuracy of reconstruction and the predictive power of the fitted model. The optimal number of groups should be high enough to allow accurate reconstruction but still low enough to prevent overfitting. In this section, I outline some basic ideas that can be useful for tackling the problem, namely the spectrum of the Laplacian matrix of the graph, the singular value decomposition (SVD) [48] of the adjacency matrix and the Akaike information criterion [2].

Estimation from the Laplacian matrix

Given an undirected graph $G(V, E)$ without loops and multiple edges, its Laplacian matrix is defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{A}$ is the adjacency matrix

and $\mathbf{D}$ is a diagonal matrix composed of the degrees of the vertices. A peculiar property of the Laplacian matrix is that its smallest eigenvalue is always zero, and its multiplicity equals the number of connected components of the graph. The number of eigenvalues close to zero are frequently used for determining the number of dense subgraphs (communities, clusters) in the graph [20, 114], and based on similar reasoning, this could be a good estimate of the number of groups that has to be used in the preference model. In fact, I show later in Section 2.6.2 that the applicability of the Laplacian is severely limited if the graph does not have a strong clustered structure.

We cannot use $\mathbf{D} - \mathbf{A}$ directly, since this form of the Laplacian applies only for undirected graphs. An extension of the Laplacian to directed graphs was introduced in [19]. This involves calculating the *Perron vector* ϕ of the transition probability matrix $\mathbf{P}$ of the graph. The transition probability matrix $\mathbf{P}$ is derived from the adjacency matrix by normalizing row sums to be 1. The Perron vector ϕ is a unique (up to scaling) left eigenvector of $\mathbf{P}$ corresponding to the largest eigenvalue and satisfying $\phi\mathbf{P} = \phi$. The existence of this vector is guaranteed by the Perron-Frobenius theorem [53]. There is no closed-form solution for ϕ, but it is easy to calculate in polynomial time numerically. The directed Laplacian is then defined as:

$$\mathbf{L} = \mathbf{I} - \frac{\mathbf{\Phi}^{1/2}\mathbf{P}\mathbf{\Phi}^{-1/2} + \mathbf{\Phi}^{-1/2}\mathbf{P}^{*}\mathbf{\Phi}^{1/2}}{2} \tag{2.20}$$

where $\mathbf{P}^{*}$ is the conjugate transpose of $\mathbf{P}$ and $\mathbf{\Phi}$ is a diagonal matrix composed of the elements of ϕ, assuming that $\sum_{i=1}^{n} \phi_i = 1$. The properties emphasized above for the undirected Laplacian hold for the directed Laplacian as well.

Besides its limited applicability to graphs without a strong clustered structure, there is another drawback: the method requires some human intervention, since one must either specify a threshold manually under which we consider an eigenvalue close enough to zero or come up with a more sophisticated algorithm that decides the threshold automatically.

Estimation from the SVD of the adjacency matrix

The singular value decomposition (SVD) of an $m \times n$ matrix $\mathbf{M}$ is a factorisation process that produces an $m \times m$ and an $n \times n$ unitary matrix ($\mathbf{U}$ and $\mathbf{V}$, respectively) and an $m \times n$ matrix $\mathbf{\Sigma}$ with non-negative numbers on the diagonal and zeros off the diagonal such that $\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{*}$ (where $\mathbf{V}^{*}$ denotes the conjugate transpose of $\mathbf{V}$). The diagonal of $\mathbf{\Sigma}$ contains the singular values, while the columns of $\mathbf{U}$ and $\mathbf{V}$ are the left and right singular vectors, respectively. It is a common convention to order the values in the

(a) Original image

(b) Reconstruction using the 16 largest singular values

(c) Reconstruction using the 4 largest singular values

FIGURE 2.4: Illustration of the SVD-based approximation of a matrix on a grayscale image

diagonal of $\mathbf{\Sigma}$ in non-increasing fashion. Plotting the sorted singular values on a scree plot (from large to small) is a good visual cue to determining the number of groups in the model: the number of groups can simply be assigned according to the number of large singular values. This can be explained as follows: one can approximate the original matrix $\mathbf{M}$ by setting all singular values but the l largest to zero and disregarding the appropriate rows of $\mathbf{U}$ and $\mathbf{V}$ that correspond to the zeroed singular values. Formally:

$$m_{ij} = \sum_{k=1}^{\min(n,m)} u_{il}\sigma_l v_{jl} \approx \sum_{k=1}^{l} u_{il}\sigma_l v_{jl} \tag{2.21}$$

This approximation is illustrated on Figure 2.4. A grayscale image[3] consisting of 440×440 pixels (shown on the left side) was converted to a 440×440 matrix whose elements varied between 0 and 1, with 0 corresponding to black and 1 corresponding to white. The matrix was decomposed into its singular vectors and values, and all but the 16 largest singular values were set to zero. The image was then reconstructed as $\mathbf{M}_{16} = \mathbf{U}\mathbf{\Sigma}_{16}\mathbf{V}^*$. An even coarser reconsruction can be achieved by keeping only the 4 largest singular values ($\mathbf{M}_4 = \mathbf{U}\mathbf{\Sigma}_4\mathbf{V}^*$).

The precision of the approximation depends on the number and magnitude of the omitted singular values. The kept parts of $\mathbf{U}$ and $\mathbf{V}$ can serve as an input for a k-means clustering algorithm in an l-dimensional space, and the results of the clustering yield a good candidate of an initial position

[3]Image courtesy of `flickr.com` user Pikaluk, licensed under the Creative Commons Attribution–Noncommercial 2.0 Generic license (`http://creativecommons.org/licenses/by-nc/2.0/`)

of the expectation-maximisation based algorithm for fitting the preference matrix.

Drineas et al [33] suggested a randomised SVD approximation algorithm that can also be used in this context. In practice, however, performing an SVD is less efficient than optimizing from a random initial position due to the additional complexity of the SVD algorithm itself, and moreover, the threshold value over which we consider a singular value large still has to be chosen. A possible approach for choosing the threshold automatically will be described in Section 2.6.2.

Estimation from Akaike's information criterion

The third approach I studied is Akaike's information criterion (AIC), proposed by Akaike in [2]. AIC is a measure of the goodness of fit of a statistical model (the preference model in our case). It is an unbiased estimator of the Kullback-Leibler divergence [66] of the model from reality, and it is an operational way of determining the appropriate trade-off between the complexity of a model and its predictive power. AIC is calculated as $2(K - \log \mathcal{L})$, where K is the number of parameters in the model and $\log \mathcal{L}$ is the log-likelihood. In the preference model, $K = k^2 + 2n + 1$, since there are $2n$ parameters for the group affiliations of the vertices, k^2 parameters represent the elements of the preference matrix and the last parameter is k itself. The suggested number of groups can be determined by fitting the model with various numbers of groups and choosing the one that minimises the Akaike information criterion. Since it is not the actual value of the AIC that matters but only the difference between AIC scores of various models for the same network, $2n + 1$ can be omitted in practical calculations.

Fitting the model to an unstructured network

Suppose we are given a large network to be fitted by the preference model. The network has no structure in it, the edges of the network are placed randomly as in the Erdős–Rényi model. A key question is then whether these algorithms are able to detect that the network is best characterised by a single vertex type due to the absence of structure or not.

Given a fixed $k > 2$, the EM and MCMC-based algorithms are likely to overfit the model, especially on relatively small networks (see Section 2.6.1 for a benchmark). This is a disadvantage of several other data-driven network analysis methods as well. For instance, the fast greedy community detection algorithm of Clauset et al. [21] finds 46.39 ± 4.7 communities in an Erdős–Rényi random network with 10^4 vertices and 3×10^4 edges (results averaged

over 100 instances). Although there are no communities in the network, even pure randomness may create structures that look like communities, resulting in a surprisingly high modularity (0.378 ± 0.001).

When there is no real structure in the network, almost all type configurations yield similar likelihood values; in other words, the landscape defined by the likelihood function is almost completely flat. The EM algorithm will get stuck very early in a local maximum, and it will converge to completely unrelated local maxima when restarted several times. The behaviour of the MCMC algorithm will be similar to a random walk over the landscape due to high acceptance probabilities and it may take many steps before the acceptance rate drops low enough to start the actual sampling.

To avoid fitting the model to networks without any real structure, I propose the following approach: before trying to fit the model with any number of groups, let us calculate the log-likelihood of the network assuming a standard Erdős–Rényi random model. Given n vertices and m directed edges, the edge density can be used to estimate the parameter p of the Erdős–Rényi model (simply let $p = \frac{m}{n(n-1)}$), resulting in a log-likelihood of $m \log p + (n^2 - n - m) \log(1-p)$. Since the Erdős–Rényi model has a single free parameter, the corresponding baseline AIC is $2 - 2m \log p - 2(n^2 - n - m) \log(1-p)$. If the AIC corresponding to 2 groups is larger than the baseline AIC, the network can be considered an Erdős–Rényi network devoid of any structure that can be described by the preference model. Out of 100 Erdős–Rényi networks with 1000 vertices and 2000-8000 edges (selected uniformly from this interval), this simple test was able to detect the absence of structure 97 times, while the three remaining cases were fitted with 2 groups.

2.5 Running time considerations

This section deals with the implementation details of both the generation and the fitting process of the preference model. I will discuss ideas for the efficient implementation of these algorithms and estimate their running times.

2.5.1 Network generation

The basic outline of the generation process is described in Algorithm 2 on page 20. The algorithm runs in at least $O(n^2)$ at first sight, since generating vertex types is done in $O(n \log k)$, and one has to loop over all possible vertex pairs once, which means approximately n^2 loop iterations. This can be reduced to $O(n \log k + m)$ where m is the number of edges in the final

network. When the network is sparse (m is $O(n)$), the optimised algorithm yields a huge performance gain over the naïve implementation.

First I show why generating the types of the vertices is $O(n \log k)$. Every vertex has to be considered once, so the iteration in line 3 of Algorithm 2 is performed n times. Drawing a random element from a discrete uniform distribution can be done in $O(\log k)$ time if we proceed as follows:

1. Let p_i be the probability of the ith element. Calculate the cumulative probability distribution by letting $q_i = \sum_{j=1}^{i} p_i$ for $i \in [1, k]$. By definition, q_{k+1} equals 1 if there are k distinct possibilities (i.e., k groups) and q_0 is zero.

2. Generate a random number x from the interval $[0, 1]$.

3. Find i such that $q_{i-1} \leq x < q_i$.

The first step is $O(k)$, but it has to be done only once if the probability distribution does not change. The second step can be considered $O(1)$. Finally, the third step can be done by a binary search, since $\mathbf{q}$ is sorted, therefore it is $O(\log k)$ [109]. This leads to a total running time of $O(k + n \log k)$ when n elements are drawn, which is essentially $O(n \log k)$ if $n \gg k$.

Second, I show why it is possible to generate the actual edges of the network in $O(m)$. In Algorithm 2, independent Bernoulli trials are performed in line 7 with various probabilities n^2 times. An edge has to be added when a trial succeeds. An equivalent loop would be to loop over the possible *group* combinations and generate edges between out-group i and in-group j with probability p_{ij}. The two descriptions are completely the same, only the order of vertex pairs being considered is different. However, we are conducting Bernoulli trials with the *same* probabilities in the latter case (apart from the time instances when we move to a different group pair). Since the geometric distribution with parameter p is the number of Bernoulli trials with probability p needed to get one success, we can proceed as follows:

1. Consider the set of vertices in out-group i and in-group j. Sort the possible vertex pairs in lexicographic order.

2. Draw a number x from a geometric distribution with probability parameter p_{ij}.

3. Drop the first $x - 1$ elements from the sorted vertex pair list.

4. If there are no more elements in the list, move on to the next group pair and go back to step 1.

5. Otherwise, add the first element of the vertex pair list as an edge to the network and go back to step 2.

The sorted list does not have to be generated at all. One can simply use an integer index in the range $\left[0, n_i^- n_j^+ - 1\right]$ where n_i^- is the number of vertices in out-group i and n_j^+ is the number of vertices in in-group j. This integer index will be advanced by x in step 3 and the resulting index will be stored in a list. The list indices can then be translated to vertex index pairs by an appropriate mapping. This way we will generate m indices during the algorithm, each of them in $O(1)$ time when assuming that drawing a number from a geometric distribution is $O(1)$. This holds as long as we can generate a uniformly distributed random number in constant time by making use of the fact that given $x \in U(0,1)$, the following expression is geometrically distributed with probability p [29]:

$$\left\lceil \frac{\log x}{\log(1-p)} \right\rceil \tag{2.22}$$

Therefore, I can finally conclude that generating a network according to the preference model requires $O(n \log k + m)$ time which is essentially linear in m.

2.5.2 Model fitting

Calculating the order of the running time of the fitting process is a complicated problem, since the methods described above are stochastic, and the number of steps needed is not known in advance (and heavily depends on the starting position anyway). Therefore, I will derive equations on the asymptotical behaviour of a single step in the EM process and in the MCMC method instead.

For both methods, two time complexity estimations will be given. One of them belongs to the case when one calculates the log-likelihood function directly. The other one belongs to the alternative form of the log-likelihood function (see Eq. (2.14)) which is feasible only if there are just a few distinct b_{ij} values in the network and we can afford the extra storage space needed to store $E_{u,v,b}$ in a three-dimensional array (which should not present an obstacle with its magnitude of $O(k^2|B|)$ if $|B|$ is small).

Asymptotic behaviour of the EM process

The E step involves the estimation of the preference matrix $\mathbf{P}$. p_{ij} is estimated by counting the number of edges going from out-group i to in-group

j (uncertain edges are taken into account by the amount described by their belief values) and dividing it by the number of possible edges between these two groups. The actual number of edges can be calculated in one single run by iterating over the edges of the network and adding their belief values in the respective cells of an appropriately sized array (the number of rows and columns equal the number of groups). This requires $O(m)$ time. Let V_i^- denote the size of out-group i and let V_j^+ denote the size of in-group j. In this case, the number of possible edges between out-group i and in-group j is $|V_i^-||V_j^+| - |V_i^- \cap V_j^+|$ due to the exclusion of loop edges. This can be calculated in $O(n)$ for a single $i - j$ pair, since the size of each set is in the order of n, and taking the intersection of two sorted sets is linear. Therefore, the whole E step requires $O(m + nk^2)$ time, but calculation of the group sizes and the size of intersections between groups can be done in the M step simultaneously with the log-likelihood calculation, and we can assume them to be readily usable in the E step if there was at least a single M step before, leaving us with a time complexity of $O(m)$ for the E step.

The M step requires the calculation of the so-called local log-likelihoods: the contribution of a single vertex to the total log-likelihood function. Each vertex tries to maximise its own local log-likelihood based on the assumption that all other vertices retain their group membership. In the initial phase of the EM process, every vertex changes its group membership frequently until the algorithm begins to converge. After a few steps, most of the group structure freezes and only a few remaining vertices change their group membership until consensus is reached. Therefore the local log-likelihoods are changing rapidly from step to step, excluding the final phase, so there is no use of introducing local update rules to the log-likelihoods due to the overhead they would cause. One has to calculate the local log-likelihood for every vertex and for every possible group affiliation of that vertex. Calculating a single local log-likelihood is $O(n)$ (one has to consider the relationship of the vertex to all $n-1$ other vertices), yielding a time complexity of $O(n^2k^2)$. We must also maintain the group sizes and the group intersection sizes which will be used by the next E step, but this does not increase the complexity.

At this point, we can make use of the alternative form of the likelihood function introduced in Eq. (2.12). The alternative form can be calculated in $O(k^2|B|)$, but it requires that the number of vertex pairs with different in-group, out-group and belief combinations be counted in advance. (Remember, B is the set of all possible belief values occurring in the specific network). This would still yield a time complexity of $O(n^2k^2|B|)$ if we counted the number of vertex pairs falling in each category in every step. As a matter of fact, such counting should only be done once after choosing the initial state in $O(n^2)$ time, and the edge counts should be updated after every single al-

teration to the group affiliations. Before moving vertex i from some group affiliation $u_{i,old}, v_{i,old}$ to a new one $u_{i,new}, v_{i,new}$, we must iterate over all belief values b_{*i} and b_{i*} concerning vertex i, decrease the appropriate edge counts $|E_{u_{i,old},v_{i,old},b_{*i}}|$ and $|E_{u_{i,old},v_{i,old},b_{i*}}|$ by one and increase $|E_{u_{i,new},v_{i,new},b_{*i}}|$ and $|E_{u_{i,new},v_{i,new},b_{i*}}|$ by one. This is an $O(n)$ operation. With this trick, the time complexity of the M step is successfully reduced to $O(nk^2|B|)$ at the expense of the extra space needed to store the $|E_{...}|$ values. $|B|$ can be omitted, since it is assumed to be constant and independent of n and k. The total time complexity of an EM iteration can then be reduced to $O(nk^2 + m)$ as well, which is practically $O(nk^2)$ when the network is sparse.

Asymptotic behaviour of the MCMC process

The key parts of the MCMC process are the calculation of (1) the log-likelihood of a given state and (2) the likelihood ratio of two given states which differ only in the group affiliation of a single vertex (since the difference between the current and the proposed state involves only a single vertex). I have shown in the previous analysis of the M step that calculating the log-likelihood is practically $O(n^2)$ when one uses the original form of the log-likelihood or $O(k^2)$ when using the alternate form. Calculating the ratio of two likelihoods is constant if both are calculated in advance, but since one of them corresponds to the proposed state, this has to be calculated again in $O(n^2)$ or $O(k^2)$ time, not forgetting that the alternative form requires extra bookkeeping of $|E_{...}|$ that means an $O(n^2)$ initialisation step and $O(n)$ overhead with every change in the group affiliations of the vertices.

An MCMC step proceeds as follows: first, a proposed state is drawn from the proposal density function. This can be done in $O(1)$ time when we use the simple proposal density function (that is, a single vertex is selected uniformly and every group affiliation is proposed uniformly for this single vertex). Next, the likelihood of the proposed state is compared to the likelihood of the current state, which involves the calculation of the new likelihood in $O(n^2)$ or $O(nk^2)$ (due to the fact that the $|E_{...}|$ values also have to be updated). The proposed state is either accepted or rejected, which is an $O(1)$ operation. The time complexity of the whole MCMC iteration is therefore $O(n^2)$ or $O(nk^2)$, depending on the form of the likelihood function we use.

2.6 Performance measurements

The performance measurements conducted on the fitting process of the preference model are primarily aimed at demonstrating the validity of the fitting

algorithm. This section is organised as follows: first, I generated graphs according to the preference model, ran the fitting algorithm on the graphs by supplying the appropriate number of groups beforehand and then compared the known and the estimated parameters of the model. These benchmarks were performed in order to test the validity of the fitting algorithm and to assess the quality of the results obtained. Next, I ran the fitting algorithms without specifying the number of groups to prove that the Akaike information criterion is suitable for determining the right value of k. Finally, I checked the rapid mixing property of the Markov chain used in the MCMC method.

2.6.1 Fitting the model with given number of groups

This benchmark proceeded as follows: graphs with 128 vertices were generated according to the preference model using 4 in- and out-types. The type distribution was uniform, so there were 32 vertices of each type on average. The preference matrix was chosen as follows: each element p_{ij} was set to one of two predefined values p_1 and p_2 with probability 0.5. p_1 and p_2 was varied between 0 and 1 with a step size 0.05. For each (p_1, p_2) combination, I generated 50 instances of the preference model. Values of the quality functions (described below) were averaged over these instances and the results were plotted as a function of p_1 and p_2. The reason why I used only two probabilities is that this way the results can be visualised on a heat map or a 2.5D plot.

To assess the fitness of the fitted model, I had to define some quality functions that compare the fitted parameters to the original (expected) ones. First I note that the number of groups and the probability matrix do not have to be compared, since the former is fixed and the latter one is calculated from the group assignments, so errors in the elements of the probability matrices are simply due to errors in the group assigments. Therefore, only the group assignments matter. The following quality functions were defined:

Success ratio. This simple measure counts the number of agreements in the expected and fitted group assigments. Let $\mathbf{u}$ and $\mathbf{v}$ denote the expected assignments and $\tilde{\mathbf{u}}$ and $\tilde{\mathbf{v}}$ denote the ones calculated by the fitting process. The success ratio is then defined as follows:

$$h(\mathbf{u}, \mathbf{v}, \tilde{\mathbf{u}}, \tilde{\mathbf{v}}) = \frac{\sum_{i=1}^{n} \delta(u_i, \tilde{u}_i) + \delta(v_i, \tilde{v}_i)}{2n} \tag{2.23}$$

where $\delta(i, j)$ is the Kronecker-delta again.

Note that the success ratio is highly dependent on the actual indices: even a perfect fit does not guarantee that the groups are indexed in

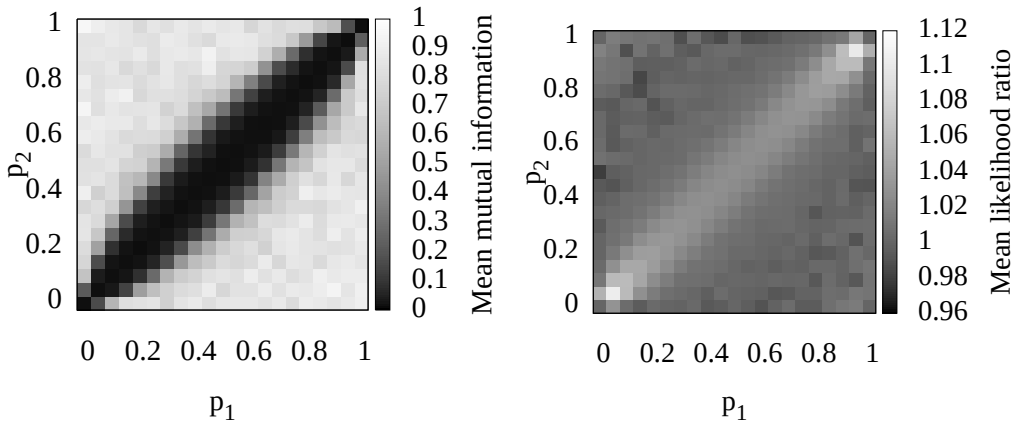
exactly the same order, e.g., it may happen that group 1 of the original model is group 2 in the fitted model. In the extreme case, we may observe zero success ratio even in case of a perfect fit if the group indices do not match each other. Therefore, the group assignments have to be brought into a canonical form by running Algorithm 4. It may still happen that there exists a rearrangement of groups that achieves larger success ratio than the canonical rearrangement, but it is easier to calculate and still yields success ratios close to the theoretical maximum (provided the fitted model is close to the original).

Normalised mutual information of the confusion matrix. This measure was suggested by Fred and Jain [42] and later applied to community detection in graphs by Danon et al. [26]. Its advantage is that it does not require appropriate rearrangement of the groups prior to calculation. One must calculate the *confusion matrix* $\mathbf{C} = [c_{ij}]$ of the expected and observed group assignments. c_{ij} is the number of vertices that are in group i in the original and group j in the fitted model. The confusion matrix can be calculated separately for in- and out-groups, but they can safely be added together to obtain a single confusion matrix and then a single quality measure, which is the normalised mutual information of the confusion matrix:

$$I(\mathbf{C}) = -2\frac{\sum_{i=1}^{k}\sum_{j=1}^{k} c_{ij}\log\frac{c_{ij}c_{**}}{c_{i*}c_{*j}}}{\sum_{i=1}^{k}\left(c_{i*}\log\frac{c_{i*}}{c_{**}} + c_{*i}\log\frac{c_{*i}}{c_{**}}\right)} \tag{2.24}$$

where c_{i*} is the sum of the ith row, c_{*j} is the sum of the jth column of the confusion matrix. c_{**} is the sum of c_{ij} for all i, j. It is assumed that $0\log 0 = 0$. When the fitted group assignment is completely identical to the expected one (apart from rearrangement of group indices), $I(\mathbf{C})$ attains its maximum at 1. $I(\mathbf{C}) = 0$ if the two group assignments are independent. Danon et al. [26] argue that this measure is in general stricter than most other quality measures proposed so far. For instance, a completely random assignment of groups still has an expected success ratio of 0.25 for 4 groups (since each pair is consistent with probability 1/4). In this case, the normalised mutual information is close to zero, which is a more intuitive description of what happened than a success ratio of 0.25. See [26] for a list of other measures they considered.

Likelihood ratio. This measure is simply the ratio of the likelihoods of the original and the fitted parametrizations, given the generated graph. Since the actual implementation is working with log-likelihoods, an easier way to calculate it is to take the difference of the log-likelihoods.



(a) Mutual information index

(b) Likelihood ratio

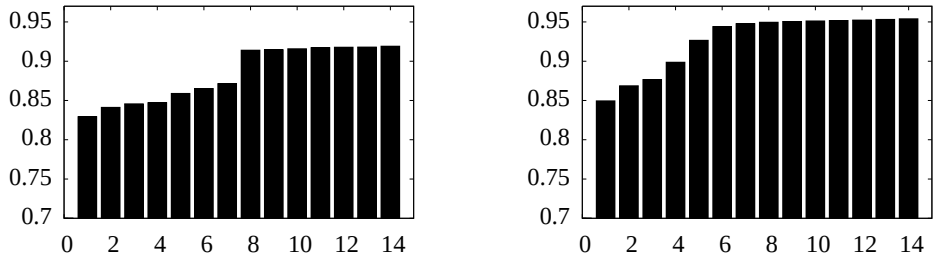
FIGURE 2.5: Mean likelihood ratios of the fitted parameterisations to the expected ones (left) and mean normalised mutual information conveyed by the confusion matrices (right) as a function of p_1 and p_2.

The likelihood ratios and the mutual information indices are plotted on Figure 2.5. Success ratios were omitted due to their high correlation with mutual information indices. As expected, the mutual information index is low when $p_1 \approx p_2$. This is no surprise, since $p_1 \approx p_2$ implies that the actual difference between different vertex types diminish: they all behave similarly, and the random fluctuations at this network size render them practically indistinguishable. The overall performance of the algorithm is satisfactory in the case of $p_1 \ll p_2$ and $p_1 \gg p_2$, with success ratios and mutual information indices larger than 0.9 in all cases. In cases when $p_1 \approx p_2$, the likelihood ratio is greater than 1, which indicates that the fitted model parameterisation is more likely than the original one. This phenomenon is an exemplar of overfitting: apparent structure is detected by the algorithm where no structure exists at all if we use too many groups (see also Section 2.4.5).

2.6.2 Choosing the number of groups

In Section 2.4.5, I described three different methods for estimating the number of groups one should use for a given network when fitting the preference model. Two of these methods required some human intervention, since one had to choose a threshold manually for the eigenvalues of the Laplacian matrix or for the singular values of the adjacency matrix.

I investigated the eigenvalues of the directed Laplacian matrix first. After some experiments on graphs generated according to the preference model, it became obvious that the number of eigenvalues of the Laplacian close to



(a) With clustered structure

(b) Without clustered structure

FIGURE 2.6: The 15 smallest *nonzero* eigenvalues of the Laplacian matrix for graphs generated by the preference model with 8 groups, either with or without a strong clustered structure (left and right panel, respectively)

zero correlate to the number of groups only if the vertex groups coincide with densely connected subgraphs. In other words, p_{ii} must be large and p_{ij} for $i \neq j$ must be small. This is illustrated on Figure 2.6. The left panel shows the case when $p_{ij} = 0.2 + 0.6\delta(i, j)$ (the graph is clustered) and the right panel shows the case when p_{ij} is 0.2 or 0.8 with 1/2 probability. There is indeed a relatively large jump after the eighth eigenvalue for the former case, but the transition is smooth for the latter. Therefore, the eigenvalues of the directed Laplacian matrix were excluded from further investigations.

In the case of SVD analysis, one has to count the large singular values. "Large" is definitely a subjective term, therefore a scree plot of the singular values is often used as a visual aid. The scree plot is simply a bar graph of the singular values sorted in their decreasing order. The plot usually looks like the side of a mountain with some debris at the bottom: the singular values decrease rapidly at first, but there is an elbow where the steepness of the slope decreases abruptly, and the plot is almost linear from there on (see Figure 2.7 for an illustration). The number of singular values to the left of the elbow is the number of groups we will choose. To allow for automated testing, I implemented a simple method to decide on the place of the elbow. The approach I used is practically equivalent to the method of Zhu and Ghodsi [137]. It is based on the assumption that the values to the left and right of the elbow behave as independent samples drawn from a distribution family with different parameters. The algorithm first chooses a distribution family (this will be the Gaussian distribution in our case), then considers all possible elbow positions and calculates the maximum likelihood estimation of the distribution parameters based on the samples to the left and right side of the elbow. Finally, the algorithm chooses the position where the likelihood

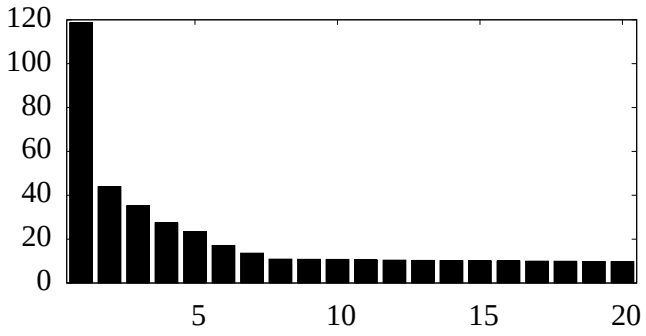


FIGURE 2.7: The largest 20 singular values of the adjacency matrix of a graph generated by the preference model with 8 groups

was maximal. Assuming Gaussian distributions on both sides, the estimates of the mean and variance are as follows:

$$\begin{gathered} \tilde{\mu}_1 = \frac{\sum_{i=1}^{q} x_i}{q} \qquad \tilde{\mu}_2 = \frac{\sum_{i=q+1}^{n} x_i}{n-q} \\ \tilde{\sigma}^2 = \frac{\sum_{i=1}^{q} (x_i - \mu_1)^2 + \sum_{i=q+1}^{n} (x_i - \mu_2)^2}{n-2} \end{gathered} \tag{2.25}$$

where x_i is the ith element in the scree plot (sorted in decreasing order), n is the number of elements (which coincides with the number of vertices) and q is the number of elements standing to the left of the elbow. Note that the means of the Gaussian distributions are estimated separately, but the variance is common. Zhu and Ghodsi [137] argue that allowing different variances makes the model too flexible. The common variance is calculated by taking into account that the first q elements are compared to μ_1 and the remaining ones are compared to μ_2. See [137] for a more detailed description of the method.

In this benchmark, 100 networks were generated with 128 vertices each. Elements of the preference matrix were chosen to be p_1 or p_2 with equal probability, as in Section 2.6.1 before, but the case of $p_1 \approx p_2$ was avoided by constraining p_1 to be above 0.6 and p_2 to be below 0.4. The number of groups was varied between 2 and 8 according to a uniform distribution. The number of groups in the fitted model was estimated by the SVD and the AIC methods, the best AIC was chosen by trying all possible group counts between 2 and 10. The AIC method proved to be superior to the SVD method: the estimation was perfect in 79% of the cases. The number of groups was underestimated by 1 group in 14, 2 groups in 3 and 3 groups in 2 cases. There were 2 overestimations by 1 group as well, resulting in a

mean squared error of 0.46 groups. On the other hand, the SVD method made severe mistakes at times; in fact, only 7% of its estimations matched the prior expectations, all other cases were overestimations, sometimes by 7 or 8 groups. This is due to the unsupervised choice of the elbow in the scree plot. It is assumed that better results can be achieved by making the choice manually, therefore the conclusion is that the SVD-based estimation should be handled with care and the AIC method is preferred when one would like to choose the number of groups automatically.

2.6.3 Rapid mixing of the Markov chain

In this benchmark, I studied whether it holds that the Markov chain possesses the rapid mixing property. (As discussed in Section 2.4.3, an exact theoretical analysis was infeasible). I generated networks with $n = 50(i+1)$ vertices, with i increasing from 1 to 20. For each i, 100 different networks were generated with 4 groups. In order to restrict benchmarking for sparse networks (which is a common property of all networks arising in practical situations), the elements of the preference matrix were chosen from $p_1 = 64/n$ and $p_2 = 16/n$ with equal probability. In each case, I counted the number of steps needed for the acceptance rate to drop below 10%. It was expected that the number of steps is polynomial in the number of vertices (or equivalently, it is log-polynomial in the number of possible states in the Markov chain). Although there was a significant variance in the number of steps, averaging over 100 instances for each i led to the plot on Figure 2.8, clearly indicating the polynomial trend, which serves as an empirical evidence of the rapid mixing property.

2.7 Using the preference model for predicting unknown links

In this final section, I demonstrate how the preference model can be used for predicting connections in a network. Recall that the fitting process produces a probability matrix $\mathbf{P}$ and two group assignment vectors $\mathbf{u}$ and $\mathbf{v}$. Consider p_{u_i,v_j} as the predicted probability of edge $i \to j$: if the graph was generated by the preference model, this would be the actual probability of that edge. To obtain a binary prediction for each edge, a threshold level τ has to be determined that reconstructs the *known* part of the network most accurately, and the same τ can be used for thresholding the predicted probabilities of the unknown connections. Determining the accuracy of reconstruction on the known parts is then practically the assessment of a binary classification.

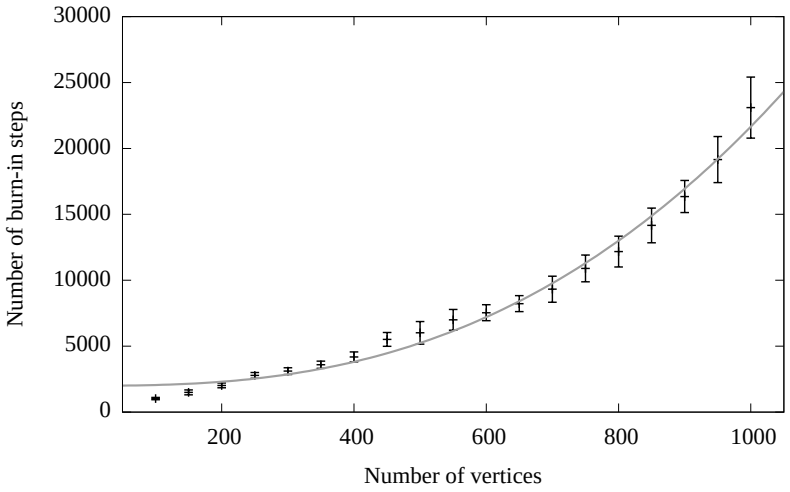


FIGURE 2.8: Number of steps needed for the Markov chain of the preference model to achieve an acceptance rate smaller than 10%. The fitted curve is polynomial in n (the number of vertices): $f(n) = 3.07 \times 10^{-4} \times n^{2.602} + 2005.91$, RMS of residuals = 727.443

Let t_p, t_n, f_p, f_n denote the number of true positive, true negative, false positive and false negative outcomes (e.g., t_n is the number of edges that were predicted as missing while actually being nonexistent, f_p is the number of missing edges predicted as existent and so on). Let $r_p = t_p/(t_p + f_p)$ and $r_n = t_n/(t_n + f_n)$, the ratio of correctly predicted positive and negative edges. Of course these measures are highly dependent on the threshold level τ of the classifier, and we wish to select a τ that maximises a quality measure derived from t_p, t_n, f_p and f_n. r_n alone is not suitable: given a high ratio of confirmed missing edges, it is easy to achieve high r_n by predicting all edges as nonexistent. Similar reasoning shows that r_p alone should also be avoided. The geometric mean of r_p and r_n is a good candidate, since it ensures that r_p and r_n is high at the same time. An even more sophisticated approach is the Matthews correlation coefficient [78], defined as follows:

$$\text{MCC} = \frac{t_p t_n - f_p f_n}{\sqrt{(t_p + f_p)(t_p + f_n)(t_n + f_p)(t_n + f_n)}} \tag{2.26}$$

In case the denominator is zero, let MCC be zero by definition (to avoid the division by zero). A Matthews coefficient of +1 represents a perfect prediction and -1 the worst possible prediction. The steps to take are then as follows:

1. Prepare the dataset. This means the construction of the belief matrix $\mathbf{B}$: $b_{ij} = 1$ if edge $i \to j$ exists, $b_{ij} = 0$ if $i \to j$ is confirmed to be

missing, otherwise b_{ij} is our *a priori* belief about the probability of the existence of an edge from i to j. In cases where no reasonable prior assumption can be made, b_{ij} should be set to 0.5, thus practically making the term belonging to $i \to j$ in the goal function constant. The maximum of the goal function will then not be influenced by $i \to j$.

2. Fit the parameters of the preference model to the network being studied using one of the methods described in this chapter.

3. Determine an appropriate probability threshold τ based on $\sqrt{r_p r_n}$ or the Matthews correlation coefficient.

4. Unknown edges can be considered as existent when their predicted probability is higher than τ.

Of course it is not necessary to use a strict binary threshold. For example, Section 4.1 will show a case study where there were a large set of unknown edges going between two distinct parts of a network. Based on domain-specific assumptions, their *a priori* belief values were set to 0.1, since it was asserted that roughly 10% of them exists in reality. None of their predicted probability exceeded the threshold applied globally to the whole network, but there were a few edges whose predicted probability was significantly higher than 0.1: they can be considered as possibly existent due to the sharp difference between their prior and predicted probabilities.

Another tool to assess the overall quality of the prediction method (irrespective of the threshold level) is the ROC curve. For every possible threshold level, one must calculate the *true positive rate* (TPR, alternatively called as *sensitivity*) and the *false positive rate* (FPR) and plot TPR versus FPR in a Descartes coordinate system[4]. They are defined as follows:

$$\mathrm{TPR} = \frac{t_p}{t_p + f_n} \qquad \mathrm{FPR} = \frac{f_p}{f_p + t_n} \tag{2.27}$$

The area under the ROC curve (AUC) is often used as a summary statistic for the overall quality of the prediction. Section 4.1.3 will demonstrate the usage of ROC curves in the comparison of several prediction methods applied to the same dataset.

[4]An alternative definition for the ROC curve is to plot TPR against 1-SPC (the specificity of the test). Since $\mathrm{SPC} = \frac{t_n}{f_p + t_n}$, the two definitions are equivalent.

2.8 Conclusion

In this section, a generic stochastic network model was presented, which can be thought of as a generalisation of the Erdős-Rényi model by introducing vertex types and type-dependent connection probabilities. The structure of the generated network is highly dependent on the types of the individual vertices. I presented algorithms to fit the preference model to a given network in a way that maximises the likelihood of the obtained parameterisation and I discussed how the model can be used to predict previously unknown links in a network. Comparisons with existing prediction methods and a case study will be presented in Section 4.1.

3

Fuzzy community structure in complex networks

RECENT STUDIES REVEALED that graph models of many real world phenomena exhibit an overlapping community structure, which is hard to be described with the classical graph clustering methods where every vertex of the graph belongs to exactly one community [102]. This is especially true for social networks, where it is not uncommon that individuals in the network belong to more than one community at the same time. Individuals who connect groups in the network function as "bridges", hence the concept of bridge is defined as a vertex that cross structural holes between discrete groups of people [16]. It is therefore important to define a quantity that measures the commitment of a node to several communities in order to obtain a more realistic view of these networks.

The intuitive meaning of a bridge vertex may differ in different kinds of networks that exist beyond sociometrics. In protein interaction networks [58], bridges can be proteins with multiple roles. In cortical networks containing brain areas responsible for different modalities (for instance, visual and tactile input processing [86]), the bridges are presumably the areas that take part in the integration and higher level processing of sensory signals [92]. In word association networks [87, 119], words with multiple meanings are likely to be bridges [102]. The state-of-the-art overlapping community

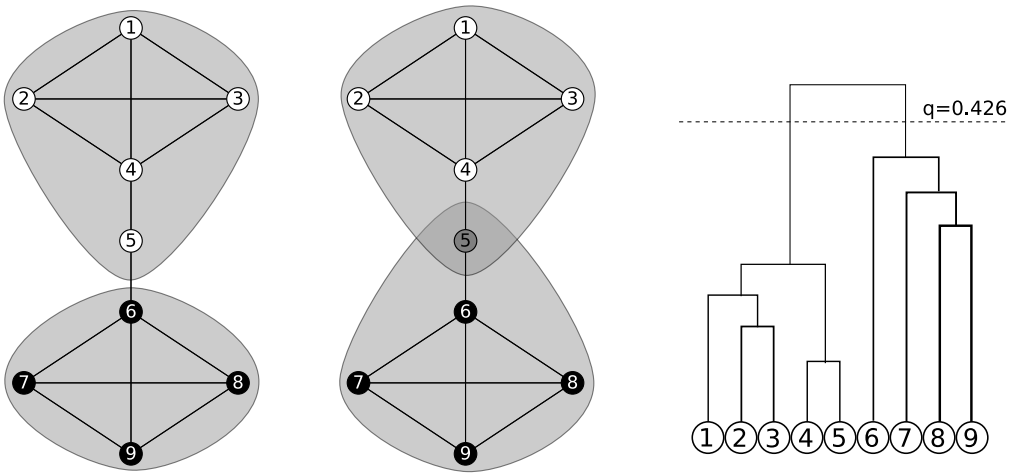
Related publications:
Nepusz T., Petróczi A., Négyessy L., Bazsó F.: *Fuzzy communities and the concept of bridgeness in complex networks.* Phys Rev E **77**(1):016107, 2008.
Nepusz T., Bazsó F., Strausz Gy.: Algorithmic identification of bridge vertices in complex networks. In: Proceedings of the 15^{th} PhD Mini-symposium, Budapest University of Technology and Economics, pp. 78–81, 2008.

detection algorithms [17, 102, 108, 135] are not able to quantify the notion of bridgeness, while other attempts at quantifying it (e.g., the participation index [51]) are only applicable to nonoverlapping communities. Note that bridges described in this thesis are not to be confused with the concept of cut edges which are sometimes also referred as bridges in classical graph theory. Articulation points (vertices whose removal disconnects the remaining subgraph) bear more similarity to the concept of bridges described in this thesis, but not all bridge vertices are articulation points.

To illustrate the concept of bridge vertices and overlapping communities, I created a simple graph shown on Fig. 3.1. A visual inspection of this graph most likely suggests two communities (see panel (*b*)), with vertex 5 standing somewhere in between, belonging to both of them at the same time. One may argue that vertex 5 itself forms a separate community, but a community with only a single node is usually not meaningful (and we can simply add more edges connecting the two communities to vertex 5 to emphasize its sharedness). This property of vertex 5 is not revealed by any classical community detection algorithm without accounting for overlaps or outliers.

One of the most commonly used method (the Clauset-Newman-Moore algorithm [21]) was applied on this graph to show how such an algorithm behaves in this case. This algorithm iteratively joins already found communities in order to form larger ones while striving to maximise a measure called *modularity* (see Section 1.2.4 for the definition). At this point, it is enough to understand that modularity is a measure assigned to a given partition of vertices, and higher modularity values represent partitions that put more edges inside clusters and leave only a few outside. The Clauset-Newman-Moore algorithm greedily joins those communities in every step that would yield a maximal increase in the modularity locally. (Several heuristics have been proposed that perform better than greedy maximisation [27, 127], but the results are the same for this example graph). The output of the process is a dendrogram (panel (*c*) of Fig. 3.1), which should be cut at the level where the modularity was maximal. However, there is always a point where the algorithm must decide whether it joins vertex 5 with one or the other large community. In fact, this decision is made surprisingly early during the process, right in the first step, as illustrated by the dendrogram.

A better solution can be achieved by applying the clique percolation method (CPM) of Palla et al. [102], which is also able to discover overlapping communities (see Section 3.1 for a short discussion of the method). In the case on Fig. 3.1, vertex 5 was classified as an outlier (a vertex not belonging to any community). This result stands closer to our visual inspection and clearly underlines the fact that in many cases, we should not assume that



(a) (b) (c)

FIGURE 3.1: Panel (a): a simple graph that can not be partitioned into two communities in a meaningful way without allowing overlaps or outliers. Vertices are colored according to the best hard partition obtained by greedy modularity optimisation [21]. Panel (b): The intuitive fuzzy partition of the graph into two groups. Panel (c): the dendrogram of the graph as calculated by greedy modularity optimiation algorithm. The dashed line denotes the level where the dendrogram should be cut in order to obtain the maximal modularity (denoted by q).

a vertex belongs to one and only one community in the graph. However, vertex 5 is not an outlier in the sense that removing it from the network would result in two disconnected components. Vertex 5 is an integral part of the network, serving as the only connection between two densely connected subgroups.

In this chapter, I will describe a method that is able to discover meaningful fuzzy communities in moderate size undirected networks [89, 92]. The chapter is organised as follows: after a generic overview of recent methods trying to tackle the problem of overlapping communities, the basic concepts behind the fuzzy clustering approach are discussed. It is followed by the description of a gradient-based method that allows us to find fuzzy communities in a given network. Several measures are introduced that classify the vertices of the network as bridges and regular vertices. Finally, benchmarks are presented to assess the performance of the method. Real-world applications will be shown later in Sections 4.2 and 4.3.

3.1 Overview

Since the groundbreaking work of Dunn [34] and Bezdek [9] on the fuzzy c-means clustering algorithm, many methods have been developed to search for fuzzy clusters in multidimensional datasets. For an overview of these methods, see Bezdek and Pal [10]. However, these methods usually require a distance function defined in the space the data belong to, therefore it is impossible to apply them to graph partitioning directly, except in cases where the vertices of the graph are embedded in an n-dimensional space. A recent paper of Zhang *et al* [135] discusses a possible embedding of the vertices of an arbitrary graph into an n-dimensional space using spectral mapping in order to utilize the fuzzy c-means algorithm on graphs. The first few eigenvectors of the Lagrangian matrix were appropriately mapped into an n-dimensional space, and fuzzy clusters were sought by the classical fuzzy c-means algorithm. They were able to identify meaningful fuzzy communities in several well-known test graphs (e.g., the Zachary karate club network [134] and the network of American college football teams [46]), but the eigenvector calculations involved in the algorithm render it computationally expensive to use on large networks.

Eigenvector calculations were also involved in the method of Capocci et al. [17]. They define a function $z(\mathbf{x})$ based on a vector $\mathbf{x}$ containing values assigned to the individual vertices under the constraint that $\sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j = 1$ as follows:

$$z(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(x_i - x_j)^2 w_{ij} \tag{3.1}$$

where w_{ij} is the positive weight of edge $i \leftrightarrow j$ (or zero if such an edge does not exist). Directed networks were symmetrized by taking $\mathbf{W}\mathbf{W}^T$ instead of $\mathbf{W}$. The stationary points of z over all $\mathbf{x}$ are then obtained by solving

$$(\mathbf{D} - \mathbf{W})\mathbf{x} = \lambda\mathbf{x} \tag{3.2}$$

where $\mathbf{D}$ is a diagonal matrix with $d_{ii} = \sum_{j=1}^{n} w_{ij}$ (the sum of the weights of all adjacent edges of vertex i) and λ is a Lagrange multiplier accounting for the constraints imposed on $\mathbf{x}$. Therefore, solving this eigenproblem is equivalent to the constrained minimisation of z. The first eigenvector is the trivial solution $\mathbf{x} = \mathbf{1}$, the remaining eigenvectors represent less and less appropriate partitions of the vertices, with eigenvector components close to each other for vertices belonging to the same community. A similarity measure is then introduced between vertices by correlating the corresponding components of the first k eigenvectors. They successfully tested their method on a word association network of 10616 nodes [119], but the nature of the algorithm

implies that not the actual communities are returned but similarity measures between vertices. This may be either an advantage or a disadvantage, depending on context.

The clique percolation method (CPM) of Palla et al. [102] is based on a simple yet powerful assumption. The method considers cliques of size k (where k is a free parameter) as basic building blocks for communities. Two cliques, C_i and C_j stand in a relation $C_i \sim C_j$ if they share at least $k-1$ vertices. The equivalence classes of the transitive closure of $\sim$ define the communities: each community consists of the set of vertices that occur in a particular equivalence class. Note that the classes are defined over the set of cliques, so no clique is shared between two classes, but vertices can be shared, forming overlaps between the communities. It can also happen that a vertex is not a part of any k-clique and is therefore left out from all communities.

Finally, I would like to mention the information bottleneck method of Tishby et al. [122], grounded in information theory and applied to graph clustering by Ziv et al. [138]. According to Tishby et al. [122], there are three random variables involved in the information bottleneck method: the input data X, the relevance variable Y and the cluster assignment Z. The task is the extraction of a compressed description Z of the input data X that accurately represents the relevance variable Y (the part of information stored in X that we are interested in). Ziv et al. [138] illustrate this on an example of protein sequences. In this case, the individual protein sequences are represented by X, while Y describes the fold of the proteins[1]. A useful Z is a random variable that maximises the mutual information between Y and Z (preserving relevant information) while minimising the mutual information between X and Z (omitting irrelevant information and compressing the dataset). Formally, we are looking for

$$\min_{p(z|x)} I(X,Z) - \beta I(Y,Z) \tag{3.3}$$

where β represents the desired tradeoff between accuracy and simplicity. Tishby et al. [122] devised an iterative process for obtaining Z, given the joint probability of X and Y.

Since there is no Y in the problem of community detection, Ziv et al. [138] had to find a method to derive it from the structure of the graph in order to employ the information bottleneck method. Their relevance variable Y is defined by the node at which a random walker stands at a given time

[1]Protein folding is a physical process by which a polypeptide folds into a three-dimensional structure. Proteins have a characteristic fold that relates to the function of the protein, and some diseases are believed to relate to the accumulation of incorrectly folded proteins in the patient.

instance t, while X is the node where the random walker started at time 0. X is either a uniform distribution or the stationary distribution of the Markov chain assigned to the graph (where every outgoing edge of the current vertex is followed with equal probability). They only studied the case when $p(z|x) \in \{0, 1\}$, but this constraint could theoretically be relaxed to arrive at fuzzy partitions.

3.2 Basic concepts

From now on, I will describe the fuzzy community detection method I suggested in [92].

3.2.1 Fuzzy partition matrices

The objective of classical community detection in networks is to partition the vertex set of the graph $G(V, E)$ into c distinct subsets in a way that puts densely connected groups of vertices in the same community. c can either be given in advance or determined by the community detection algorithm itself. For the time being, let us assume that c is known. In this case, a convenient representation of a given partition is the *partition matrix* $\mathbf{U} = [u_{ik}]$ [9]. $\mathbf{U}$ has $N = |V|$ columns and c rows, and $u_{ik} = 1$ if and only if vertex k belongs to the ith subset in the partition; otherwise it is zero. From the definition of the partition, it clearly follows that $\sum_{i=1}^{c} u_{ik} = 1$ for all $1 \leq k \leq N$. The size of community i can then be calculated as $\sum_{k=1}^{N} u_{ik}$, and for any meaningful partition, we can assume that $0 < \sum_{k=1}^{N} u_{ik} < N$. These partitions are traditionally called *hard* or *crisp* partitions, because a vertex can belong to one and only one of the detected communities [9]. For instance, the hard partition of the example graph on Fig. 3.1 is described by the following partition matrix:

$$\mathbf{U} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{3.4}$$

The generalization of the hard partition follows by allowing u_{ik} to attain any real value from the interval $[0, 1]$. The constraints imposed on the partition

matrix remain the same [111]:

$$u_{ik} \in [0,1] \text{ for all } 1 \leq i \leq c, 1 \leq k \leq N \tag{3.5a}$$

$$\sum_{i=1}^{c} u_{ik} = 1 \text{ for all } 1 \leq k \leq N \tag{3.5b}$$

$$0 < \sum_{k=1}^{N} u_{ik} < N \text{ for all } 1 \leq i \leq c. \tag{3.5c}$$

Eq. (3.5b) simply states that the total membership degree for each vertex must be equal to 1. Informally, this means that vertices have a total membership degree of 1, which will be distributed among the communities. Eq. (3.5c) is the formal description of a simple requirement: we are not interested in empty communities (to which no vertex belongs to any extent), and we do not want all vertices to be grouped into a single community. Partitions of this type are called *fuzzy partitions*. The fuzzy membership degrees for a given vertex can be thought about as a trait vector that describes some (possibly nonobservable) properties of the entity which the vertex represents in a compact manner. A possible fuzzy partition of the graph that is shown on Fig. 3.1 corresponds to the following partition matrix:

$$\mathbf{U} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{3.6}$$

In the upcoming sections, I will sometimes refer to the ith column of $\mathbf{U}$ as $\mathbf{u}_i$, the *membership vector* of vertex i.

3.2.2 Similarity and the goal function

A meaningful partition (let it be hard or fuzzy) should group vertices that are somehow similar to each other in the same community. It is reasonable to assume that an edge between vertex v_1 and v_2 implies the similarity of v_1 and v_2, and likewise, the absence of an edge implies dissimilarity. Let us assume that we have a function $s(\mathbf{U}, i, j)$ that satisfies the following criteria:

1. $s(\mathbf{U}, i, j) \in [0, 1]$
2. $s(\mathbf{U}, i, j)$ is continuous and differentiable for all u_{ij}.
3. $s(\mathbf{U}, i, j) = 1$ if the membership values of v_i and v_j suggest that they are as similar as possible.

4. $s(\mathbf{U}, i, j) = 0$ if the membership values of v_i and v_j suggest that they are completely dissimilar (there is no chance that they belong to the same community).

Let us call such $s(\mathbf{U}, i, j)$ a *similarity function*, and for the sake of simplicity, it will be denoted by s_{ij} from now on (not emphasizing its dependence on $\mathbf{U}$). Now suppose we have a prior assumption about the actual similarity of the vertices, denoted by $\tilde{s}_{ij}$ for v_i and v_j. This leads to the following equation, which measures the fitness of a given partition $\mathbf{U}$ of graph $G(V, E)$ by quantifying how precisely it approximates the prescribed similarity values with s_{ij}:

$$D_G(\mathbf{U}) = \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} \left(\tilde{s}_{ij} - s_{ij}\right)^2, \tag{3.7}$$

where w_{ij}'s are optional weights and $n = |V|$ is the number of vertices in the graph. Better partitions yield smaller $D_G(\mathbf{U})$ values. For the sake of notational simplicity, the matrices $\mathbf{W} = [w_{ij}]$, $\mathbf{S}(\mathbf{U}) = [s_{ij}]$ and $\tilde{\mathbf{S}} = [\tilde{s}_{ij}]$ are also introduced. From now on, I assume that $\tilde{\mathbf{S}} = \mathbf{A} + \mathbf{I}$, the adjacency matrix of the graph with ones along the main diagonal, in concordance with the assumption that the similarity of connected vertex pairs should be close to 1 and the similarity of disconnected vertex pairs should be close to zero. The ones in the main diagonal account for the fact that a vertex should be as self-similar as possible. The only thing left is to precisely define a similarity function s_{ij} that satisfies the conditions prescribed above. The definition used in [92] was the following:

$$s_{ij} = \sum_{k=1}^{c} u_{ki} u_{kj} = \mathbf{u}_i^T \mathbf{u}_j \tag{3.8}$$

Eq. (3.8) simply states that the similarity of vertices i and j is the dot product of the respective membership vectors, which makes it particularly convenient to calculate the similarity matrix, since $\mathbf{S}(\mathbf{U}) = [s_{ij}] = \mathbf{U}^T\mathbf{U}$. Note that any other similarity function is conceivable as long as it satisfies the criteria described above.

The matrix form of this problem with equal and positive weights bears some similarity with the Cholesky decomposition. In this case, $D_G(\mathbf{U})$ is zero if and only if $\tilde{\mathbf{S}} = \mathbf{U}^T\mathbf{U}$. This would be easy to solve if $\mathbf{U}$ was an $n \times n$ matrix (meaning that the number of communities c is equal to the number of vertices n), and $\tilde{\mathbf{S}}$ was symmetric and positive-definite. Since none of these conditions hold, all that one can do is to minimize the difference between $\tilde{\mathbf{S}}$ and $\mathbf{U}^T\mathbf{U}$ by finding an appropriate $\mathbf{U}$.

3.3 Finding fuzzy communities in undirected networks

3.3.1 Outline of the algorithm

Given an expected similarity matrix $\tilde{\mathbf{S}}$ (which coincides with the adjacency matrix in this case except the main diagonal, which contains only ones), and the number of communities c, the goal is to find a partition matrix $\mathbf{U}$ that minimises Eq. (3.7), or in other words, to perform a weighted least squares fit of the similarity matrix $\mathbf{S} = \mathbf{U}^T\mathbf{U}$ to $\tilde{\mathbf{S}}$. The Levenberg-Marquardt algorithm [72, 76] would be an efficient solution for the problem had there not been the constraints imposed on $\mathbf{U}$ (see Eq. (3.5): all column sums in $\mathbf{U}$ must be 1 and all u_{ij} must be between zero and one). This poses a challenge to most numeric optimisation algorithms, since they usually cannot take constraints into account, or even if they can (like the L-BFGS-B method [136]), the constraints they can enforce are not satisfactory for this problem. Therefore, we must take another approach.

There exist a set of necessary conditions that restrict the set of possible $\mathbf{U}$'s worth evaluating [61, 65]. The Karush–Kuhn–Tucker conditions are a generalisation of the method of Lagrange multipliers, as they can take both equality and inequality constraints into account. The formal necessary conditions are as follows:

Theorem 3.1 (Karush–Kuhn–Tucker conditions)**.** *Let the objective function $f : \mathbb{R}^n \to \mathbb{R}$ be convex and let there be constraint functions $g_i : \mathbb{R}^n \to \mathbb{R}$ and $h_j : \mathbb{R}^n \to \mathbb{R}$. Suppose they are continuously differentiable at a point x^*. If x^* is a local minimum, then there exist constants μ_i and ν_j such that*

$$\begin{gathered} \nabla f(x^*) + \sum_i \mu_i \nabla g_i(x^*) + \sum_j \nu_j \nabla h_j(x^*) = 0 \\ \mu_i \geq 0 \qquad g_i(x^*) \leq 0 \qquad h_j(x^*) = 0 \qquad \mu_i g_i(x^*) = 0 \end{gathered} \tag{3.9}$$

However, due to the computational complexity of minimising such a function for networks larger than only a few vertices, this solution is only of theoretial interest. The solution I proposed in [92] is based on a simple steepest descent method with adaptive step size, for which I used Lagrange multipliers to incorporate the equality constraints into the goal function. The remaining inequality constraints were enforced by adjusting the step size appropriately during the search in order not to step out from the allowed range.

Let there be $\lambda = [\lambda_1, \lambda_2, \ldots, \lambda_n]$ Lagrange multipliers. λ_i will be responsible for the equality constraint $\sum_{k=1}^{c} u_{ki} = 1$. The modified goal function is

defined as follows:

$$\tilde{D}_G(\mathbf{U}, \lambda) = \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} \left(\tilde{s}_{ij} - s_{ij}\right)^2 + \sum_{i=1}^{n} \lambda_i \left(\sum_{k=1}^{c} u_{ki} - 1 \right) \tag{3.10}$$

The modified goal function compactly encodes the original goal function and the equality constraints, since $\frac{\partial}{\partial u_{ij}} \tilde{D}_G(\mathbf{U}, \lambda) = 0$ (for all $1 \leq i \leq c$ and $1 \leq j \leq n$) ensures that we are at a stationary point of the goal function, and $\frac{\partial}{\partial \lambda} \tilde{D}_G(\mathbf{U}, \lambda) = 0$ ensures that the conditions of Eq. (3.5b) are satisfied. Therefore, stationary points of Eq. (3.10) will also be stationary points of Eq. (3.7) and they do not violate Eq. (3.5b).

To employ a gradient-based iterative optimization method, the derivatives of the goal function with respect to u_{kl} are needed. First note that

$$\frac{\partial s_{ij}}{\partial u_{kl}} = \frac{\partial}{\partial u_{kl}} \left(u_{ki} u_{kj}\right) \tag{3.11}$$

which is zero, except when $i = l$ or $j = l$:

$$\frac{\partial s_{ij}}{\partial u_{kl}} = \begin{cases} 2u_{kl} & \text{if } i = l \wedge j = l \\ u_{ki} & \text{if } i \neq l \wedge j = l \\ u_{kj} & \text{if } i = l \wedge j \neq l \\ 0 & \text{if } i \neq l \wedge j \neq l \end{cases} \tag{3.12}$$

The partial derivative of $\tilde{D}_G(\mathbf{U}, \lambda)$ with respect to u_{kl} is therefore

$$\begin{aligned} \frac{\partial \tilde{D}_G}{\partial u_{kl}} &= -\sum_{i=1}^{n} \sum_{j=1}^{n} 2 w_{ij} (\tilde{s}_{ij} - s_{ij}) \frac{\partial s_{ij}}{\partial u_{kl}} + \lambda_l \\ &= -2 \sum_{i=1}^{n} w_{il} (\tilde{s}_{il} - s_{il}) u_{ki} - 2 \sum_{j=1}^{n} w_{lj} (\tilde{s}_{lj} - s_{lj}) u_{kj} + \lambda_l \end{aligned} \tag{3.13}$$

Let $e_{ij} = w_{ij} \left(\tilde{s}_{ij} - s_{ij}\right)$ denote the residual corresponding to the vertex pair (i, j) to shorten the formulae. Summing the partial derivatives for $k = 1, 2, \ldots c$, making them equal to zero and substituting Eq. (3.5b) back where appropriate leaves us with:

$$\begin{aligned} c\lambda_l &= 2 \sum_{k=1}^{c} \sum_{i=1}^{n} (e_{il} + e_{li}) u_{ki} \\ \lambda_l &= \frac{2}{c} \sum_{i=1}^{n} \left[(e_{il} + e_{li}) \sum_{k=1}^{c} u_{ki} \right] \\ \lambda_l &= \frac{2}{c} \sum_{i=1}^{n} (e_{il} + e_{li}) \end{aligned} \tag{3.14}$$

The substitution of Eq. (3.14) into Eq. (3.13) yields one component of the modified goal function's gradient vector:

$$\frac{\partial \tilde{D}_G}{\partial u_{kl}} = 2 \sum_{i=1}^{N} (e_{il} + e_{li}) \left(\frac{1}{c} - u_{ki} \right) \tag{3.15}$$

Simple steepest descent optimisation

The simplest gradient-based algorithm for finding a local minimum of $\tilde{D}_G$ is the following:

1. Start from an arbitrary random partition $\mathbf{U}^{(0)}$ and let $t = 0$.
2. Calculate the gradient vector of $\tilde{D}_G$ according to Eq. (3.15) and the current $\mathbf{U}^{(t)}$.
3. If $\max_{k,l} |\frac{\partial \tilde{D}_G}{\partial u_{kl}}| < \varepsilon$, stop the iteration and declare $\mathbf{U}^{(t)}$ a solution.
4. Otherwise, calculate the next partition in the iteration with the following equation:
$$u_{ij}^{(t+1)} = u_{ij}^{(t)} + \alpha^{(t)} \frac{\partial \tilde{D}_G}{\partial u_{ij}} \tag{3.16}$$
where $\alpha^{(t)}$ is a small step size constant chosen appropriately.
5. Increase t and continue from step 2.

$\alpha^{(t)}$ can be determined by a line search towards the direction defined by the gradient vector, it can be adjusted iteratively according to some simulated annealing schedule (see [99] for a comparison of strategies), or it can be made adaptive from iteration to iteration by checking the difference of the values of the goal function in the last few steps: the step size can be increased if the value of the goal function decreased, and it must be decreased if the value of the goal function increased. We must also make sure that the procedure does not end up accidentally in a saddle point or a local maximum of $D_G(\mathbf{U})$. Local maxima are easy to avoid by choosing an $\alpha^{(t)}$ that always decreases the value of the goal function in the next step. Saddle points and not too deep local minima can be avoided by randomly mutating the acquired solution and see if the iteration converges back to the original, unmutated solution.

According to my simulations, the quality of the result is not affected by the initial membership degrees, but the speed of convergence is. In the extreme case, if one chooses all u_{ij} to be equal to $1/c$, all the gradients will be zero (see Eq. 3.15), therefore it is suggested to use a randomised initial

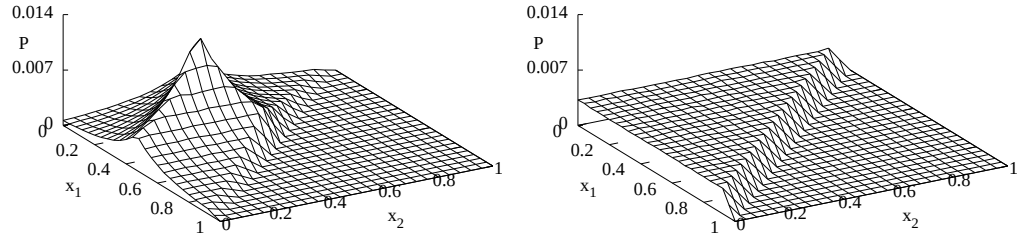


FIGURE 3.2: Left: joint probability density function of x_1 and x_2 when components of $\mathbf{x} = [x_1, x_2, x_3]$ are drawn from a uniform distribution and then their sum is normalised to 1. Right: the same joint distribution when the components are drawn from a gamma distribution with shape and scale parameters equal to 1, resulting in a joint Dirichlet distribution with $\alpha = [1, 1, 1]$. Note that the latter one is uniform in the allowed range, while the former one is not.

partition matrix. I propose choosing the initial membership degrees from a uniform distribution while still satisfying the sum constraints. Uniformity with respect to the constraints is not straightforward to achieve. The naïve approach is to choose a random number from the interval $[0, 1]$ for every u_{ij} and divide them with their respective column sums to satisfy Eq. (3.5b). However, this method is biased towards membership vectors describing vertices equally participating in every community. The proper way to sample from all possible membership vectors is to draw every vector from a Dirichlet distribution with order c and $\alpha = [1, 1, \ldots, 1]$ where α has c coordinates, since this ensures that all possible valid membership vectors occur with equal probability. Such a distribution can be generated by drawing c independent random samples from gamma distributions each with shape and scale parameters equal to 1, and dividing each variable with the sum of all of them [29]. Figure 3.2 illustrates the difference between the two sampling methods.

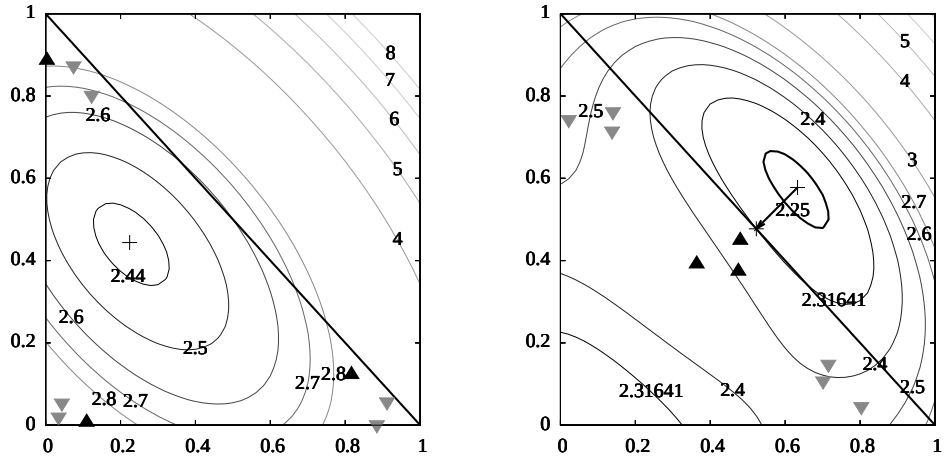
For the sake of completeness, I show that $\mathbf{U}^{(t+1)}$ remains a partition matrix if $\mathbf{U}^{(t)}$ was a partition matrix. Recall that a partition matrix satisfies Eq. (3.5a) and Eq. (3.5b). In the first step, I choose $\mathbf{U}^{(0)}$ that satisfies Eq. (3.5c). The persistence of Eq. (3.5a) and Eq. (3.5c) is straightforward if I always keep $\alpha^{(t)}$ low enough, so I only have to prove the persistence of Eq. (3.5b):

$$
\begin{aligned}
\sum_{i=1}^{c} u_{ik}^{(t+1)} &= \sum_{i=1}^{c} u_{ik}^{(t)} + \sum_{i=1}^{c} \alpha^{(t)} \frac{\partial \tilde{D}_G}{\partial u_{ik}^{(t)}} \\
&= 1 + 2\alpha^{(t)} \sum_{i=1}^{c} \sum_{j=1}^{n} (e_{jk} + e_{kj}) \left(\frac{1}{c} - u_{ij}^{(t)} \right) \\
&= 1 + 2\alpha^{(t)} \sum_{j=1}^{n} (e_{jk} + e_{kj}) \underbrace{\left(1 - \sum_{i=1}^{c} u_{ij}^{(t)} \right)}_{0} \\
&= 1
\end{aligned}
\tag{3.17}
$$

Despite all the drawbacks usually associated with simple gradient descent methods (e.g., the slow convergence in case of pathological goal functions such as the Rosenbrock function [110]), the method described above gives satisfactory results on real-world and artificial networks as well (see [92] or Section 3.5). I used the following strategy to set the step size: initially, the step size was set at 0.5. After every three consecutive successful step (when the goal function decreased), I multiplied the step size by 1.5. After every unsuccessful step (when the goal function decreased), the step size was halved and the step retried with the smaller step size. The search was stopped when the step size became smaller than a predefined threshold (e.g., 10^{-3}).

Optimisation with the BFGS algorithm

Instead of the simple method presented above, one can also make use of the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm [113], a quasi-Newton method that makes use of the fact that we are looking for roots of the derivative, therefore Newton's method for root finding can be applied to the derivative. Since Newton's method requires the derivative of the function whose root is being sought, we would need the Hessian matrix of the modified goal function. Given n vertices and c communities, the Hessian matrix is of size $nc \times nc$ (since there are nc variables) and even though most elements in the Hessian are zeros, it is still computationally expensive to keep track of. The BFGS method overcomes this difficulty by approximating the Hessian at every step based on previous successive gradient vectors. The details are given in [113]. Due to the shape of the goal function, it is advised to decompose it to parts corresponding to individual vertices and optimise the parts one by one. The local modified goal function of vertex i is as follows (derived from



(a) Optimum inside the valid region (b) Optimum outside the valid region

FIGURE 3.3: Local goal function of vertex i in the fuzzy community detection algorithm with $n = 10$ vertices and $k = 3$ without the Lagrangian term. $u_{1,i}$ is plotted on the X axis, $u_{2,i}$ on the Y axis. The thin black diagonal line shows the boundary of the valid region for $\mathbf{u}_i$. Black upright triangles represent the current membership vectors of vertices connected to vertex i, gray triangles pointing down represent vertices not connected to vertex i. The optimum is shown by a crosshair. If it falls outside the valid region (right), it is replaced by the closest point of the valid region (denoted by an arrow).

Eq. (3.10) by omitting terms not related to vertex i):

$$\tilde{D}_{G,i}(\mathbf{U}, \lambda) = \sum_{j=1}^{n} w_{ij} \left(\tilde{s}_{ij} - s_{ij}\right)^2 + \lambda_i \left(\sum_{k=1}^{c} u_{ki} - 1 \right) \tag{3.18}$$

The partial derivatives are:

$$\frac{\partial \tilde{D}_{G,i}}{\partial u_{ki}} = 2 \sum_{l=1}^{N} (e_{il} + e_{li}) \left(\frac{1}{c} - u_{kl} \right) \tag{3.19}$$

See Figure 3.3 for the typical shape of the local goal function in a simple example.

Let $\mathbf{u}_i^{(t)}$ denote the membership vector of vertex i in time instance t. Starting from an initial random partition matrix $\mathbf{U}^{(0)}$, only one vertex will be modified in every time step. Every time I modify the membership vector of a vertex, I store the distance it travelled in r_i ($r_i = \infty$ if the membership vector has not been modified so far). The BFGS method is then utilised according to the following scheme:

1. If $\max r_i < \varepsilon$, terminate the algorithm. Otherwise, if there exists vertices with $r_i = \infty$, choose one from them randomly. If all r_i's are finite, then choose a vertex randomly according to a distribution where the probability of choosing vertex i is proportional to r_i^γ (γ and ε are parameters of the algorithm, I achieved good results with $\gamma = 2$ and $\varepsilon = 10^{-3}$).

2. Increase t by one.

3. Optimise the local goal function of the chosen vertex i according to the BFGS method with all other membership vectors held constant at their values in time step $t - 1$. The partial derivatives in Eq. (3.19) ensure that $\sum_{k=1}^{c} u_{ki}^{(t)} = 1$ in the new optimum. Should there be any $u_{ki}^{(t)}$ in the new optimum that is less than zero, set it to zero and renormalise $\mathbf{u}_i^{(t)}$ to adhere to Eq. (3.5b). This correction seems rather arbitrary, but due to the typical shape of the local goal function, the corrected goal function value is in practice sufficiently close to the local minimum inside the allowed range (see the right panel of Figure 3.3 for an illustration). Also note that we do not have to explicitly take care of $\mathbf{u}_{ki}^{(t)}$'s larger than one; the existence of such a membership value implies that there are negative membership values as well in the same membership vector, since the sum of membership values in a membership vector is always one.

4. Calculate the Euclidean distance between $\mathbf{u}_i^{(t)}$ and $\mathbf{u}_i^{(t-1)}$ and store it in r_i. Formally, let $r_i = \sum_{k=1}^{c} \left(u_{ki}^{(t)} - u_{ki}^{(t-1)} \right)^2$.

5. Go back to step 1.

The implementation of this method can take advantage of the many open-source BFGS minimisers available, e.g., the one in the GNU Scientific Library [43]. On the other hand, the algorithm suffers from floating point rounding errors in some cases (which is usually reported by the solver stating that it was unable to achieve the desired precision), and the algorithm may spend too much time on finding the exact minimum of a *local* goal function, which is not necessary in the early stage of the minimisation process, since the place of the local minimum is heavily influenced by the current membership vector of other vertices that also change rapidly at the stage. Random mutations of found local minima can be used similarly as in the simple steepest descent case in order find a better local minima.

3.3.2 Connection weights

Up to now, I did not say anything about the weighing terms w_{ij} in Eq. (3.7). The easiest approach is to consider all vertex pair weights as equal, leading to an algorithm that works best in small networks. The reason is that the algorithm implicitly assumes that the absence of a connection between vertices i and j implies the dissimilarity of those two vertices. This does not necessarily hold in large networks: maybe the vast majority of vertex pairs did not have a chance to interact and form a connection due to other constraints (e.g., the spatial distribution of the vertices) not represented in the network structure. Consider a real-world example from the field of social networks. The assumption of the algorithm applies for the social network of the employees of a small enterprise, where everyone had a chance to meet all the others and form a connection; however, the same assumption may not hold for a multi-national company with many divisions all over the world. In the latter case, we must turn to a more precise approach that takes this phenomena into account. This knowledge will be represented in the $\mathbf{W}$ weight matrix.

There is always a possibility that a connection in the network is a byproduct of some random process and does not imply the similarity of those vertices. To cite an example from the field of social networks again: a connection between two people whose social activity is well over the average (they are both connected to a lot of other people) is less important or "surprising" than a connection between two people who are otherwise well-separated from other parts of the network. Similarly, given two hubs in the network, the absence of a connection between them is a stronger precursor of dissimilarity than the absence of a connection between two nodes situated on the opposite peripheries of the network. More formally, the connection between two vertices with high degree is less interesting than a connection between vertices with low degree, since the former one is formed frequently even in a random network that follows the same degree distribution. A sophisticated weighing should associate more weight to connections that are not likely to happen in random networks [90].

Now we can turn to the *configuration model* of Molloy and Reed [82]: a random network conditioned on the degree sequence $s_0, s_1, s_2, \ldots$, where s_i is the number of vertices with degree i. The probability of an edge in such a network depends solely on the degrees of the endpoints of the edge: $P_{ij} = d_i d_j / 2m$, where d_i and d_j are the degrees of the endpoints, respectively, and m is the number of edges. In other words, P_{ij} is the *expected* number of edges between vertex i and j. Since we only have a single instance of the network being analysed, there is no better choice than to let the *observed*

number of edges be equal to A_{ij}. More weight is assigned to vertex pairs where the expected and the observed number of edges differ significantly:

$$w_{ij} = \left(A_{ij} - \frac{d_i d_j}{2m} \right)^2 \tag{3.20}$$

In practice, the vast majority of vertex pairs will have negligible w_{ij} values. By considering these weights equal to zero, some elements of the similarity matrix will not have to be calculated, since they will not influence the goal function or its derivatives due to the corresponding zero weight.

The difference between the unweighted and weighted approaches will be illustrated on real-world examples in Section 4.3. I also note that the Molloy–Reed configuration model is not the only model that can be used for deriving connection weights in the network, and the investigation of other possible weighing schemes is one of the possible future research directions.

3.3.3 Choosing the number of communities

The most important parameter of the fuzzy community detection method is c, defining the number of communities the algorithm tries to discover in the network. This parameter is the keystone of most community detection algorithms, and determining c in a self-consistent way without human intervention is definitely a complicated problem. Spectral methods rely on the largest eigenvalues of the adjacency matrix $\mathbf{A}$ or the smallest eigenvalues of the Laplacian matrix $\mathbf{L} = \mathbf{D} - \mathbf{A}$ (where $\mathbf{D}$ is a diagonal matrix with diagonal elements d_i, the degrees of the vertices) to define the number of communities, but this is usually done by visual inspection, and since the eigenspectrum of most networks found in real applications resemble a straight line instead of a step function, choosing c is not free of subjective elements. For instance, the number of eigenvalues of the Laplacian matrix of a graph that are close to zero are often used as the value of c, but this only replaces the value of c with another parameter: a threshold level that decides which eigenvalues are considered to be close to zero. The threshold is then chosen manually.

In order to get rid of the human intervention needed to choose c based on the eigenvalues, I proposed a different, divisive approach in [92] which also spares some computation in the early stage of the algorithm. Initially, a fuzzy bisection of the graph is computed by setting $c = 2$. After that, whenever the optimisation gets stuck in a local minimum, another degree of freedom is added to the system by increasing c and continue with the optimization from the last local minimum until it converges again. This process is repeated until the newly introduced community does not improve the overall community

structure of the network. The community structure is assessed by a fuzzified variant of the modularity function, originally introduced by Newman [94] (see Section 1.2.4 for a short explanation of the measure). Here I only recall that the "crisp" modularity of a network with vertex i belonging to community $c(i)$ is then defined as:

$$Q = \frac{1}{2m} \sum_{i,j} \left(A_{ij} - \frac{d_i d_j}{2m} \right) \delta_{c(i),c(j)} \tag{3.21}$$

where $\delta_{c(i),c(j)}$ is the Kronecker-delta. Since the community structure in fuzzy applications is not clear-cut, the predicate that "vertices i and j belong to the same community" also has a fuzzy truth value between 0 and 1. When the membership degree u_{ki} is considered the probability of the event that vertex i is in community k, the probability of the event that vertex i belongs to the same community as vertex j becomes the dot product of their membership vectors, resulting in the already introduced similarity measure s_{ij}, which can be used in place of $\delta_{c(i),c(j)}$ to obtain a fuzzified variant of the modularity:

$$Q_f = \frac{1}{2m} \sum_{i,j} \left(A_{ij} - \frac{k_i k_j}{2m} \right) s_{ij} \tag{3.22}$$

Note that in the case of crisp communities (there exists only one k for every vertex i such that $u_{ik} = 1$), the fuzzified modularity Q_f is exactly the same as the crisp modularity Q. In order to determine the optimal number of fuzzy communities, the number of communities c should be increased iteratively and the one resulting in the highest fuzzified modularity Q_f should be chosen.

3.4 Identifying bridge vertices

One of the advantages of fuzzy community detection is that it enables us to analyse to what extent a given vertex is shared among different communities. In this section, a couple of measures will be introduced to quantify the sharedness of a vertex.

3.4.1 Bridgeness

Intuitively, the goal is to define a measure that is zero when the vertex belongs to only one of the communities, and increases gradually as it becomes shared between more and more communities. By limiting this measure from above by 1, we achieve the measure called *bridgeness*, which is the distance of a membership vector $\mathbf{u}_i = [u_{1,i}, u_{2,i}, \ldots, u_{ci}]$ from a reference

vector $\left[\frac{1}{c}, \frac{1}{c}, \ldots, \frac{1}{c}\right]$ in the Euclidean vector norm, inverted and normalised to the interval $[0, 1]$ as follows:

$$b_i = 1 - \sqrt{\frac{c}{c-1} \sum_{j=1}^{c} \left(u_{ji} - \frac{1}{c}\right)^2} \tag{3.23}$$

Other vector norms are also conceivable with different normalization factors to make the result span over the interval $[0, 1]$. The Euclidean norm yields the sample variance of $\mathbf{u_i}$ under the square root, multiplied by c, hence the bridgeness measure with Euclidean norm can be rewritten as:

$$b_i = 1 - \sqrt{c \operatorname{Var}(u_i)} \tag{3.24}$$

3.4.2 Centrality-weighted bridgeness

Note that b_i attains its theoretical maximum when v_i belongs to all of the communities exactly with the same membership degree, therefore it is possible that in this case, v_i is more likely to be an outlier in the graph (a vertex belonging to none of the communities) rather than a bridge. To distinguish outliers and real bridges, one should also look at the centrality measures of the vertex: high centrality supports the assumption that the vertex is effectively a bridge, because despite its central role in the network, the algorithm was not able to assign it to a single community. Low centrality may mean that the algorithm strived to make the vertex dissimilar from almost all other vertices, therefore it made it belong to all the communities. The simplest measure that incorporates centrality and bridgeness score into a single number is simply defined as the product of the degree and the bridgeness of the node, and will be called *degree-weighted bridgeness* from now on. Other centrality measures (e.g. betweenness centrality, closeness centrality or eigenvector centrality [128]) can also be used. More sophisticated centrality measures take into account that several networks contain vertices that have a crucial role but a relatively low degree (e.g. metabolic networks, as shown in [51]). It is also suggested to plot a chosen centrality measure versus the bridgeness score for each vertex to visually aid the selection of bridge vertices and outliers. An example of this kind of plot will be shown in Section 4.2 on Fig. 4.6.

Note that the direct dependence of the bridgeness measure on the number of communities implies a somewhat strange behaviour. Take the graph on Figure 3.1 as an example. The fuzzy partition has two communities, and the bridgeness of vertex 5 in the middle is 1, the maximal possible bridgeness that can be achieved. Now consider a graph that contains k exact copies of

the original graph in k separate components! This graph would have $2k$ fuzzy communities, and the bridgeness of vertex 5 would immediately drop to $1/k$, since *theoretically* there could be vertices that reside in all $2k$ communities to some extent; but due to the topology of the graph, it is of course not really meaningful. Therefore, care should be taken when evaluating bridgeness measures: bridge-like vertices will have to be detected based on the fact that they have a bridgeness value larger than the "typical" bridgeness, e.g., more than one standard deviation higher than the mean bridgeness in the network. The next subsection will present an alternative approach to bridgeness that does not suffer from this drawback.

3.4.3 Exponentiated entropy

Another approach to a bridgeness-like measure can be derived by quantifying how many "significant communities" a given vertex has. Consider $\mathbf{u}_i$ as a probability distribution function defined on a finite support set with c elements: u_{ki} is the probability that vertex i belongs to community k. Take a large sample from this distribution. The average information conveyed by elements of this sequence is given by the Shannon entropy of the p.d.f.: $H(\mathbf{u}) = -\sum_{k=1}^{c} u_{ki} \log_2 u_{ki}$. The entropy will be high if the vertex belongs to many communities significantly, while it will be low if the vertex is practically a member of a single community. $2^{H(\mathbf{u})}$ is therefore a good measure of the number of significant communities χ_i of a single vertex i [123]:

$$\chi_i = 2^{-\sum_{k=1}^{c} u_{ki} \log_2 u_{ki}} = \prod_{k=1}^{c} u_{ki}^{-u_{ki}} \tag{3.25}$$

χ_i equals 1 if there is only a single u_{ki} that is not zero, i.e. the vertex belongs only to one community. The maximal value of $\chi_i = c$ is attained when all u_{ki}'s are equal to $1/c$. In general, $\chi_i \approx 2$ and $\chi_i \geq 2$ is a property of bridge-like vertices.

The exponentiated entropy can also be used for post-processing the obtained fuzzy partitions. The post-processing phase would remove irrelevant community memberships as described in Algorithm 6. Informally, the post-processing phase keeps only the $\lceil \chi_i \rceil$ most significant community memberships of every vertex i, the rest of them is set to zero and $\mathbf{u_i}$ is renormalised after pruning.

Algorithm 6 Post-processing fuzzy community detection results based on the exponentiated entropy

Require: $\mathbf{U} = [u_{ki}]$, $n > 0$, $k \geq 2$

for $i = 1$ to n **do**
  Calculate χ_i according to Eq. (3.25)
  Let $\mathbf{x} = [x_i]$ a vector such that $u_{x_1,i}, u_{x_2,i}, \ldots, u_{x_k,i}$ is in descending order
  $s := 0$
  for $j = \lceil \chi_i \rceil + 1$ to k **do**
    $s := s + u_{x_j,i}$
    $u_{x_j,i} := 0$
  end for
  for $j = 1$ to $\lceil \chi_i \rceil$ **do**
    $u_{x_j,i} := u_{x_j,i}/(1 - s)$
  end for
end for

3.5 Benchmark results

This section presents the assessment of the method on computer-generated graphs. I will use uniform edge weights in all cases. Comparison with other overlapping community detection approaches will be given in Section 4.2.2 on a real dataset.

The graphs on which I tested the method can be grouped as follows:

Graphs with nonoverlapping community structure. These graphs consisted of 1024 vertices grouped into four communities, each containing 256 vertices. Each vertex had an average of $k_{in} = 24$ links to other vertices in its own community and an additional $k_{out} = 8$ links to vertices in different communities. The generated graphs contained 16,384 edges and had a density of 0.031. k_{in} and k_{out} was then later varied to study the robustness of the algorithm as the distinction between intra- and inter-community edges diminish. This method was already used earlier in the literature to assess the performance of community detection algorithms (see [94, 108]).

Graphs with overlapping community structure. These graphs consisted of 768 regular vertices and 256 bridge candidates. (I will shortly explain why are these vertices only "candidates" and not always real bridges). Vertices were grouped into two communities so that each community

contained 384 regular vertices and 128 bridge candidates. Regular vertices had the same connectional patterns as in the nonoverlapping case: 24 links on average to other vertices in their community and 8 links to the other community. Bridge candidates had 6 links to other vertices in their community, 12 links to other bridge candidates in their community, 6 links to bridge candidates of the other community and 8 links to regular vertices of the other community. The edge count and the density was equal to the nonoverlapping case. Some of the bridge candidates thus became bridges between the two communities, but due to the randomised nature of the algorithm, one cannot tell in advance which bridge candidates will actually become bridges.

3.5.1 Nonoverlapping community structure

In order to compare a fuzzy partition with an expected hard partition, I introduced the notion of *dominant community*. The dominant community of a vertex is the community to which it belongs to the greatest extent. Formally, community i is the dominant community of vertex j if $u_{ij} \geq \max_k u_{kj}$ for $1 \leq k \leq c$. If multiple dominant communities exist according to this definition, the one with the lowest index is chosen. Out of 1000 graphs with nonoverlapping community structures, the algorithm classified all vertices correctly in 97.4% of the test cases after converting the achieved fuzzy partition to its hard counterpart using the dominant communities. It was also able to infer the actual number of communities automatically in all cases using the fuzzified modularity. To further study the distribution of intra-community and inter-community edges, I varied the number of inter-community edges (k_{out}) from 0 to 24 while keeping $k_{in} + k_{out}$ constant. When k_{out} reaches 24, the graph practically becomes an Erdős-Rényi random graph devoid of any community structure, since the connection probability between any two of the pre-defined communities is then equal. Figure 3.4(a) shows the results of the benchmark. The quality of the calculated community structure was assessed by the normalised mutual information of the confusion matrix (described earlier in Section 2.6.1). The performance of the algorithm degrades suddenly when the number of inter-community links exceeds 16. This is the point where on average there are more links between the communities than inside them (which is, of course, not really reminiscent of real communities).

3.5.2 Overlapping community structure

Generated graphs with overlapping community structure were used to test the sensitivity of the algorithm to vertices standing between communities.

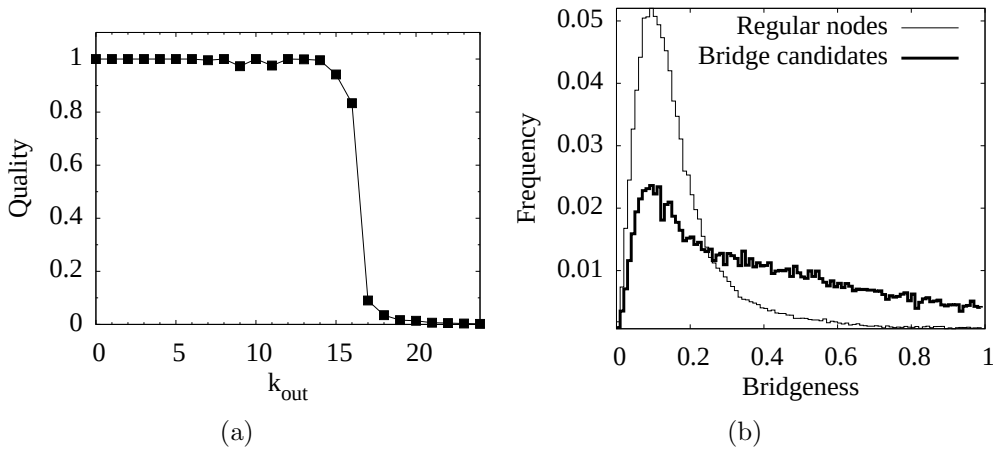


FIGURE 3.4: Panel (a) shows the performance of the algorithm for a graph with nonoverlapping community structure. Inter-cluster link count (k_{out}) was varied while keeping the average degree ($k_{in} + k_{out}$) constant. The quality of the obtained result was measured using the normalised mutual information of the confusion matrix [26]. Panel (b) shows the frequencies of bridgeness values in a graph with overlapping community structure. Thin line shows frequencies for regular nodes, thick line shows frequencies for bridge candidates. The bin width of the histogram was set to 0.01 (100 bins).

As described above, the graph generation model declared 128 vertices out of 512 in both communities as bridge candidates, and clearly distinguished them by their different connectional patterns: bridge candidates tended to connect to each other with a higher probability than to the regular vertices in their communities, even if they originally belonged to different communities, creating an overlap between the two communities. Because of the randomised nature of this model, not all bridge candidates became real bridges between the communities, but they had a significantly higher chance of becoming one. I used the bridgeness value introduced in Section 3.4 to assess the quality of the results. I expected that bridge candidate vertices exhibit a different bridgeness score distribution than the regular vertices in the same graph. I also required that vertices identified as bridges by the algorithm should be among those that have been declared bridge candidates before test graph generation. I generated 1000 random graphs using this graph model and plotted the distribution of the bridgeness scores on Figure 3.4(b). The different nature of the two distributions was supported by a Kolmogorov-Smirnov test (p-value less than 2.2×10^{-16}). Regular vertices usually had lower bridgeness scores than the bridge candidates, and I found that 92.8% of the identified bridges (based on their standardised bridgeness scores) were

among bridge candidates, confirming that the algorithm is sensitive to the existence of overlaps between communities. Similar results were achieved with the number of significant communities χ_i in place of the bridgeness measure.

Real-world applications of the fuzzy community detection method will be shown in Sections 4.2 and 4.3.

3.5.3 Running time

In this section, I will estimate the time complexity of the simple steepest descent based community detection. This method consists of an initialisation step (where the initial partition matrix is generated from random vectors drawn from a Dirichlet distribution) and an iterative optimisation. The time complexity of the method is dominated by the latter one. With n vertices and c communities, the time complexity of calculating the initial partition matrix is $O(nc)$, assuming that drawing a random number from a gamma distribution takes $O(1)$. Calculation of the similarity matrix is $O(n^2c)$ if it is carried out naïvely without weighing the vertex pairs according to Section 3.3.2, but this can be reduced significantly if the network is sparse and vertex pairs with insignificant weight are left out from the calculation. Calculating the gradient vectors in each step takes $O(n^2c)$ time, since there are nc gradient components and each of them requires a summation of n terms. Choosing the maximum gradient component for each vertex is $O(nc)$ and calculating the next partition matrix is also $O(nc)$, assuming that the step size can be chosen in $O(1)$ (which is true for simulated annealing strategies or adaptive step sizes based on the decline of the goal function between subsequent steps). This results in an overall time complexity of $O(n^2ch)$, where h is the number of steps necessary for the algorithm to terminate. Therefore, I expected that the calculation time scales approximately quadratically with the number of vertices if $n \gg c$. This is confirmed by measurements (see Figure 3.5). When calculating the time complexity, I did not take into account the possible dependence of h (the number of steps the algorithm takes before it concludes that it converged) on n, the number of vertices. This is due to the fact that h depends on many factors: the initial configuration, the required precision of the solution and the inner structure of the network. The measurements presented on Figure 3.5 suggest that h does not scale with n: the number of steps taken by the algorithm is mostly independent of the number of vertices involved. The phenomena can be explained by noting that the networks analysed are sparse, and the optimal membership vector of a vertex is mostly dependent only on the membership vectors of its close neighbors and the hubs of the network when appropriate weighing is used; in

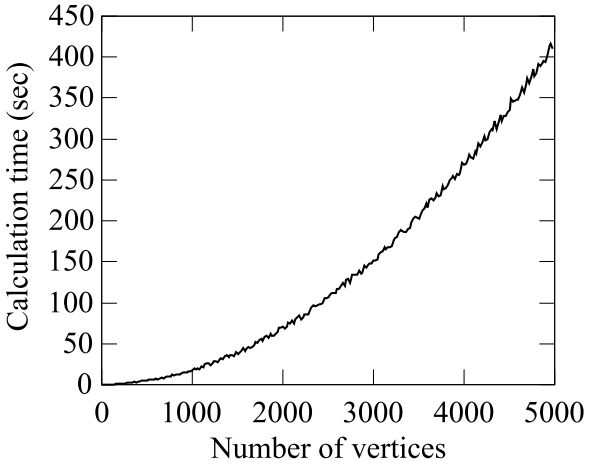


FIGURE 3.5: Running time of the algorithm as a function of the number of vertices in a graph with 4 communities. Fitting $f(x) = ax^b$ resulted in the parameters $a = 2.3 \times 10^{-5} \pm 1 \times 10^{-6}$ and $b = 1.968 \pm 4.24 \times 10^{-3}$ (standard deviation from the fitted curve = 2.583), confirming the reasoning on the quadratic running time of the algorithm when implemented with sparse matrices.

other words, vertex-vertex interactions are mostly local. The running time of the algorithm on dense networks is approximately $O(n^3)$, suggesting that h scales with n in this case (and weighing vertex pairs does not help, since most vertex pairs will have significant weight that can not be neglected). This case is not of much interest to us, since the existence of communities (dense subnetworks inside a *sparse* network) is usually asked in the context of sparse networks.

4

Applications

THE PURPOSE OF THIS CHAPTER is to show some practical applications of the methods presented in Chapters 2 and 3. The studies will be devoted to cortical and social networks. The review of Sporns et al. [118] is a good introductory material to recent advances regarding cortical networks, while the book of Wasserman and Faust [128] gives an in-depth overview of methods and techniques pertaining to social networks. A short description of these types of networks will be given in the respective sections.

4.1 Predicting missing neural connections in cortical networks

The cerebral cortex is probably the most prominent example of natural information processing networks. At the lowest level, the cortical network is composed by physically (i.e., via chemical and electrical synapses) connected nerve cells. The cortex, in general, consists of approximately 10^{10} nerve cells, each receiving numerous connections in the order of 10^3 (up to about 10^4) [13]. However, the study of cortical networks in such detail is yet impossible: there is no accurate (or at least partly accurate) map of the connections of individual nerve cells in such complex organisms as humans, although it is noteworthy that detailed mappings exist for the nervous system of a roundworm species, namely *Caenorhabditis elegans* [131] (but it has only 302 neurons).

At a higher organizational level, the cortex is composed of about a hundred structurally and functionally specialised regions or areas with highly variable shape and size [125]. This level of organisation is of great interest, because the available anatomical and imaging (fMRI, PET, EEG, MEG)

techniques made the investigation of the network of cortical areas possible. Most of our knowledge about this large-scale cortical network comes from studies charting the neuronal, i.e. synaptic connections between cortical areas. Since the usage of sensitive and powerful tract tracing techniques is not feasible in humans, the neuronal connections between the areas have been being studied intensely in non-human primates, especially in the macaque, which serves as a model of the human cortex [125]. The network of areas is usually represented in binary form considering only knowledge of the existence of a connection between the areas. However, this representation is not accurate: many connections treated as missing in the cortical network may exist in reality if their existence has never been checked experimentally due to methodological difficulties. A method that is able to give meaningful predictions about where to look for additional connections would therefore be of primary importance.

A practical way of approaching this problem is to check how exactly can the network be reconstructed based on a given index or network measure [25, 59], and then use this measure to predict the existence of unknown connections. The two studies published up to now present data on such predictions of yet unknown connections in the cortex [25, 59]. The results of these studies, especially those by da Fontoura Costa et al. [25], who investigated a set of measures, suggest that connectional similarity of the areas is a good predictor in reconstructing the original cortical network. However, they also report a relatively large number of violations, where known existing connections were predicted as non-existent and known non-existent connections to be existing in the reconstructed graphs. This suggests that using other approaches could result in better reconstruction of the cortical network. The approach I used was discussed in Chapter 2 and in [91].

4.1.1 The dataset

The dataset I analyse in this section is the graph model of the visuo-tactile cortex of the macaque monkey brain[1], published in Négyessy et al. [86]. The whole network contains 45 vertices and 463 directed links among them. The existence of connections included in the network were confirmed experimentally, while connections missing from the network were either explicitly checked for and found to be nonexistent, or never checked experimentally. To illustrate the uncertainty in the dataset being analyzed, I emphasize that 1157 out of the 1980 possible connections were uncertain and only 360 were known to be absent. Such uncertainty poses a challenge to traditional local

[1]The dataset is available at `http://www.mit.bme.hu/~nepusz`

Table 4.1: Basic properties of the cortical networks

	Visual	Sensorimotor	Visuo-tactile
Vertices	30	15	45
Known connections (edges)	335	85	463
Known nonexistent connections	310	0	360
Unknown connections	225	125	1157
Density	0.385	0.404	0.233
Density (excluding unknowns)	0.519	1.000	0.548
Diameter	3	3	5
Average path length	1.6632	1.767	2.149
Reciprocity	0.850	0.888	0.815

similarity indices and path ensemble methods (see Section 4.1.3).

The network consists of two dense subnetworks corresponding to the visual and the sensorimotor cortex (30 and 15 vertices, respectively). The visual cortex can also be subdivided into the so-called dorsal and ventral parts using a community detection algorithm based on random walks [69]. Most of the uncertain connection candidates are heteromodal (originating in the visual and terminating in the sensorimotor cluster, or the opposite), and it is assumed that the vast majority of possible heteromodal connections are indeed nonexistent. The basic properties of these networks are shown in Table 4.1, while the adjacency matrix of the visuo-tactile network is depicted on Figure 4.1. Note that since the visual and sensorimotor cortices are subnetworks of the visuo-tactile network, their adjacency matrices are the upper-left 30×30 and lower-right 15×15 submatrices of the adjacency matrix. In order to compare my results with previous reconstruction attempts that were only concerned with the visual cortex [25, 59], I present results based on the visual subnetwork as well as the whole visuo-tactile cortex.

4.1.2 Results

Visual cortex

Since the visual cortex is part of the visuo-tactile cortex, I utilised the fitting method on the visual cortex simply by taking the spanning subgraph consisting of the visual areas (see Figure 4.1 where the first 30 areas are visual). It is noteworthy that most of the unknown connections are adjacent to the areas VOT and V4t, and the subgraph consisting of the vertices PITd, PITv, CITd, CITv, AITd, AITv, STPp and STPa (all belonging to the ventral stream of

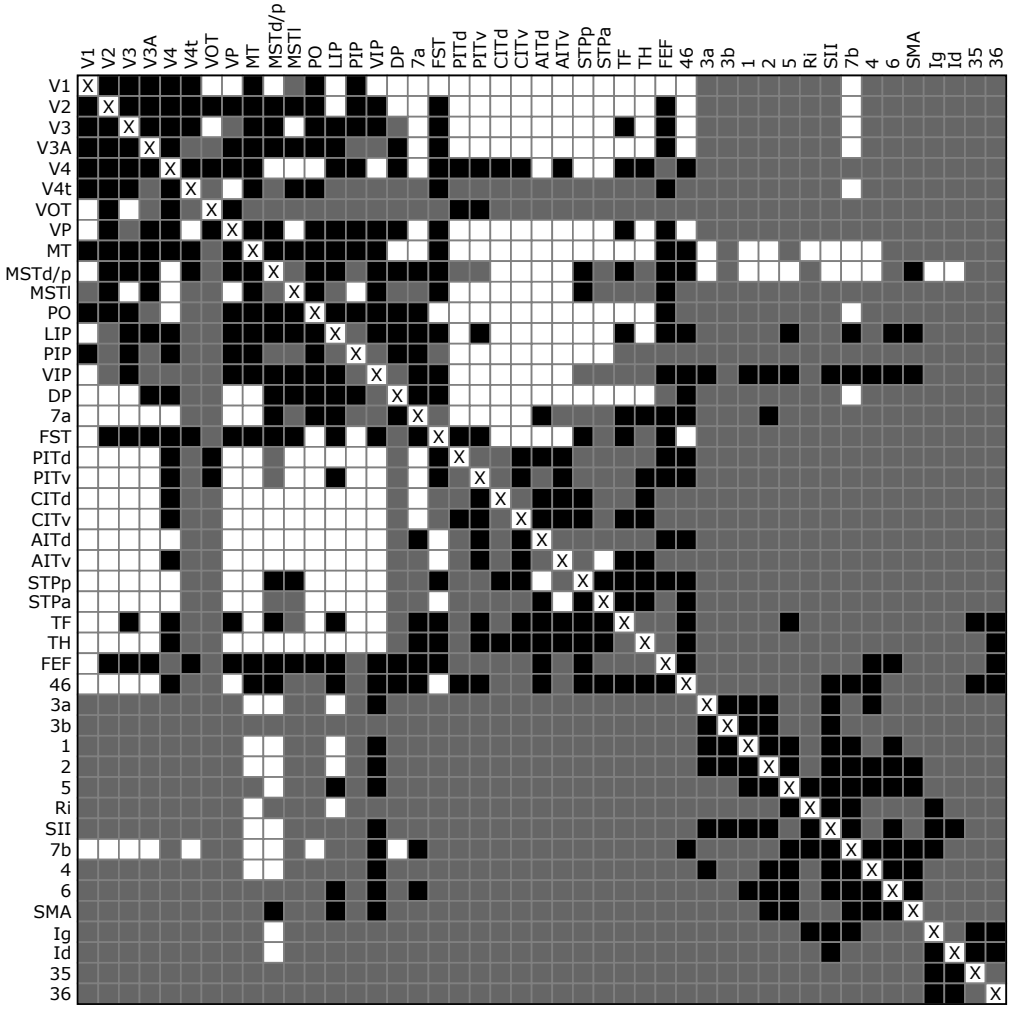


FIGURE 4.1: Adjacency matrix of the visuo-tactile cortex dataset. Black cells denote known existing connections, white cells denote known nonexistent connections. Grey cells are connections not confirmed or confuted experimentally. The upper left 30×30 submatrix is the adjacency matrix of the visual cortex, the lower right 15×15 submatrix describes the sensorimotor cortex.

k	Log-likelihood	AIC	τ	$\sqrt{r_p r_n}$	MCC
2	-481.608	1091.216	0.50	0.808	0.617
3	-440.280	1018.560	0.50	0.812	0.661
4	-413.027	978.055	0.50	0.835	0.683
5	-394.664	959.328	0.50	0.852	0.705
6	-378.271	948.543	0.50	0.871	0.751
7	**-363.146**	**944.292**	**0.50**	**0.888**	**0.785**
8	-353.071	954.143	0.50	0.888	0.785
9	-340.886	963.773	0.47	0.896	0.800
10	-331.626	983.253	0.43	0.909	0.820
11	-319.771	1001.543	0.49	0.906	0.814
12	-307.766	1023.532	0.48	0.907	0.815
13	-300.657	1059.315	0.48	0.919	0.839
14	-297.540	1107.081	0.46	0.918	0.838
15	-288.615	1147.231	0.49	0.922	0.844

TABLE 4.2: Log-likelihoods, AIC values, $\sqrt{r_p r_n}$ and MCC in the visual cortex

the visual cortex) is also mostly unknown. Based on the connection density of the visual cortex (assuming unknown connections to be nonexistent), the probability of the existence of a connection classified as unknown was set to 0.385. The search for the optimal configuration started from a random initial position, first improved by the EM method, then followed by MCMC sampling after reaching the first local maximum. The sampling process was terminated when its running time exceeded a predefined time limit. The final solution was the one with the best likelihood among all samples.

Based on the benchmarks described in Section 2.6.2, the optimal number of groups was determined by the Akaike information criterion of the obtained partitions at various group numbers from 2 to 15. Partitions having more than 15 groups do not seem feasible, since in these cases, at least one of the groups will contain only one vertex. I note that the singular values of the adjacency matrix suggested only two groups (which is congruent with the anatomical fact that the visual cortex is composed of two major pathways, namely the dorsal and the ventral stream), but the minimal AIC value was achieved using 7 groups (see Table 4.2). Thresholds were selected in a way that maximises the geometric mean of correctly predicted existing and nonexisting edges ($\sqrt{r_p r_n}$) in the known part of the network. τ fluctuated around 0.5 in all cases. As expected based on the reasoning outlined in Section 2.4.5, the success rate increased steadily as I increased the number of groups, but the divergence of τ from 0.5 after having more than 7 groups is

	Preference model		Costa et al. [25]
	$\tau = 0.5$	$\tau = 0.654$	
r_p	0.940	0.926	0.701
r_n	0.839	0.734	0.697
$\sqrt{r_p r_n}$	0.888	0.825	0.699
MCC	0.785	0.669	0.397

TABLE 4.3: Comparison of the reconstruction of the known parts of the cortical network based on the preference model and the results of Costa et al. [25]. The main diagonal of the adjacency matrix was excluded from comparison. Note that Costa et al. used a slightly different set of areas. The results of Jouve et al. [59] were unsuitable for comparison, since they provided predictions only on the unknown part of the network.

likely to be a precursor of overfitting.

The fitted model with 7 groups provided probabilities for the 225 unknown connections, 137 of them were above the optimal threshold $\tau = 0.5$. The ratio of predicted edges approximately matched the density of the visual cortex when I excluded the unknown connections from density calculation (see Table 4.1). However, if I wanted the ratio of predicted connections match the density of known connections in the visual cortex, I would have had to increase τ to 0.654, predicting only 81 connections. This ratio matches the one reported in [25]. The predicted adjacency matrix with τ=0.654 is shown on Fig. 4.2. Results are summarised in Table 4.3.

I compared these results to earlier studies [25, 59]. Comparisons were based on the percentage of matching predictions. Since both studies took a slightly different set of areas into consideration, I did not take into account those areas that were not present in any of the matrices. The results are summarised in Table 4.4.

The predictions of Jouve et al. [59] are based solely on the connections in the network model of the visual cortex, similarly to the method presented here. The agreement between the two predicted matrices is moderate: 61.6% of the predictions match for $\tau = 0.5$ and only 47% for $\tau = 0.654$. Most of the disagreements involved areas V4t (28), VOT (22), FEF (17) and DP (16). Area MSTd was joined together with MSTp in this study (resulting in the vertex denoted by MSTd/p), therefore neither MSTd nor MSTd/p was taken into account. I note that the matrix used in the present paper incorporated the results of anatomical experiments that could not have been included in the matrix in [59], therefore the moderate match between the two matrices can be explained by the differences in the initial dataset. Since the prediction method of Jouve et al. [59] did not attempt to reconstruct the entire network

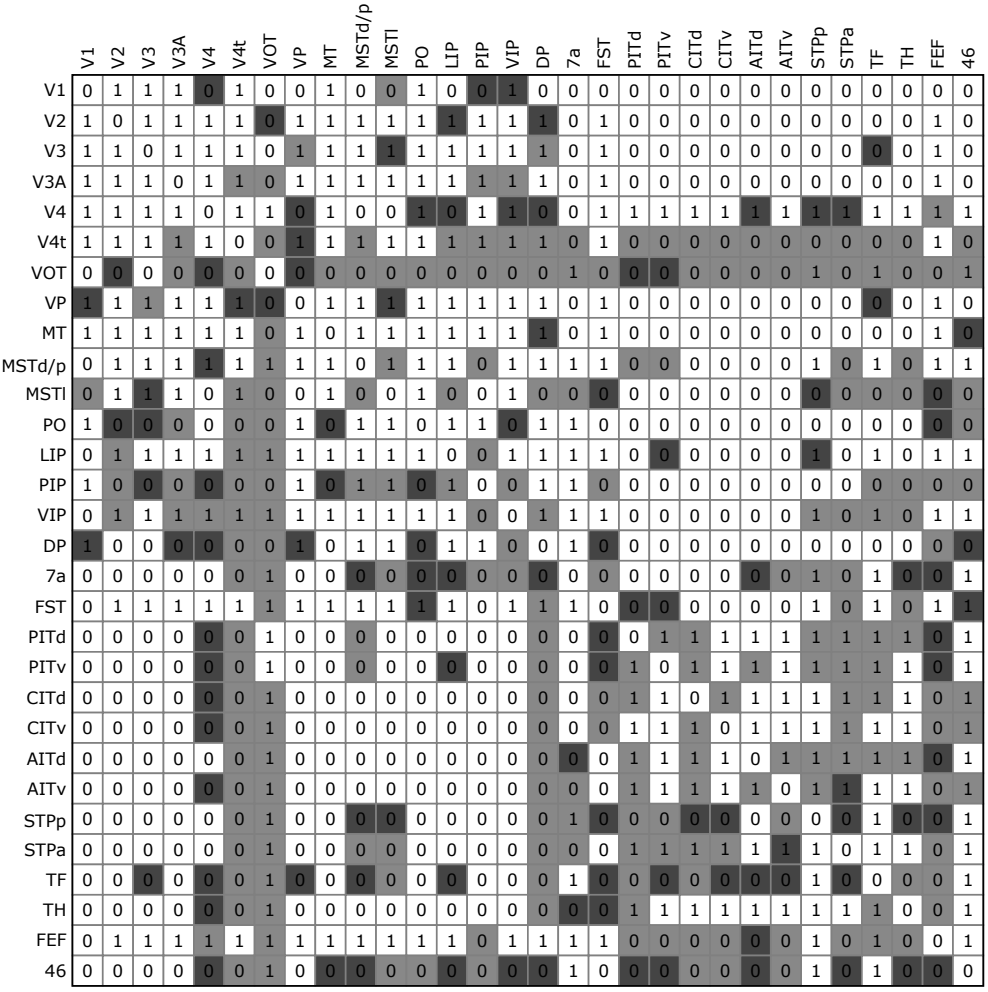


FIGURE 4.2: The predicted adjacency matrix of the visual cortex with 7 vertex groups. White cells denote confirmed existing and absent connections. Dark gray cells denote mismatches between the known and the predicted connectivity. Light gray cells denote predictions for unknown connections.

		Preference model	
		$\tau = 0.5$	$\tau = 0.654$
Jouve et al. [59]	Matching 1's	114	75
	Matching 0's	16	24
	Mismatches	81	112
	Match percentage	61.6%	46.9%
Costa et al. [25]	Matching 1's	76	58
	Matching 0's	59	88
	Mismatches	74	63
	Match percentage	64.6%	69.8%

TABLE 4.4: Comparison of the predictions of Costa et al. and Jouve et al. with the preference model regarding the unknown parts of the cortical network dataset. Areas not present in all datasets considered were excluded from comparison.

(predictions were made only on unknown connections), no comparison could be made based on the success rates of the two methods.

The predictions published by Costa et al. [25] are based on several topological (e.g., node degree, clustering coefficient) and spatial features (e.g., area sizes, local density of the areas in the 3D space, based on their known positions in the cortex). In this sense, the reconstruction method based on the preference model is simpler, since it depends solely on the connection matrix. I also note that Costa et al. inferred the topological features from a symmetrised connectivity matrix, thus their predicted matrix is also completely symmetric, while fitting the preference model produced a matrix where only 67.4% of the predicted, previously unknown connections were reciprocal. The ratios of correctly predicted 1's and 0's in the visual cortex reported by Costa et al. were slightly worse ($r_n = 244/350 = 0.697$, $r_p = 207/295 = 0.701$, $\sqrt{r_n r_p} = 0.699$, loop connections excluded). Note that the comparison can not be fully accurate because of the slightly different set of areas used in the analysis (MIP and MDP were present only in [25], whereas MSTd/p and VP were present only in the matrix used in this study). 69.8% of the predictions presented here matched the predictions of [25], and all predicted edges with a probability larger than 0.8 were predicted in [25] as well. The possible biological implications of the predicted connections are discussed in [91].

Visuo-tactile cortex

The network model of the visuo-tactile cortex is an extension of the visual cortex, obtained by adding the 15 areas of the sensorimotor cortex and their

TABLE 4.5: Likelihoods, AIC values, $\sqrt{r_p r_n}$ and MCC in the visuo-tactile cortex

K	Log-likelihood	AIC	τ	$\sqrt{r_p r_n}$	MCC
5	-814.956	1859.913	0.42	0.810	0.636
6	-783.935	1819.871	0.43	0.840	0.715
7	-756.352	1790.705	0.46	0.848	0.694
8	-736.163	1780.327	0.37	0.859	0.721
9	-718.422	1778.844	0.43	0.870	0.747
10	**-697.078**	**1774.156**	**0.49**	**0.887**	**0.761**
11	-683.335	1788.671	0.46	0.888	0.765
12	-684.105	1836.210	0.46	0.890	0.776
13	-665.337	1848.674	0.47	0.896	0.791
14	-653.755	1879.510	0.48	0.903	0.803
15	-652.173	1934.347	0.40	0.906	0.825

respective connections. Connections going between a visual and a sensorimotor area are called heteromodal connections. The density of the sensorimotor cortex is slightly higher than the visual cortex. Based on the connection densities, the probability of the existence of an unknown connection was assumed to be 0.385 inside the visual cortex and 0.404 inside the sensorimotor cortex. Unknown heteromodal connections were assumed to exist with probability 0.1 based on a prior anatomical consideration that roughly 10% of possible heteromodal connections should exist.

Note that the vast majority of heteromodal connections is unknown. In fact, there was no confirmed nonexisting sensorimotor connection indicated in the data set (see Figure 4.1). The optimal configuration was found by the combination of the EM and the MCMC method, similarly as above.

The number of groups in the preference model was determined again by the Akaike information criterion. The eigenvalues of the Laplacian and the singular values of the adjacency matrix suggested 5 groups, which is again in concordance with anatomical considerations, but as shown above, 5 groups was insufficient to reproduce even only the visual cortex. Log-likelihoods, AIC values and success rates are shown on Table 4.5, from 5 to 15 groups. The optimal number of groups with the lowest AIC was 10.

The fitted model with 10 groups predicted 225 connections with $\tau = 0.49$ out of the 1157 unknown ones ($r_n = 0.883$, $r_p = 0.892$, $\sqrt{r_p r_n} = 0.887$). This is 19.4% of the unknown connections and it roughly matches the overall density of the visuo-tactile cortex (23.3%). However, only 5 heteromodal connections (all originating from LIP) were predicted apart from the known existing ones. This is due to the fact that very little is known about the

heteromodal connections, and the algorithm cannot generalise beyond them with higher confidence. I also note that the posterior probability of many heteromodal connections in this case stayed at 0.1, the same as their prior probability. Taking into account that even a significant difference between the prior and the posterior probabilities of the heteromodal connections may not reach the threshold of 0.49, I decided to use different thresholds for non-heteromodal and heteromodal connections (τ_1 and τ_2, respectively). τ_1 was left at 0.49, while τ_2 was lowered to 0.137, the average *a posterori* probabilities of the unknown heteromodal connections. This new configuration yielded $R_0 = 0.831$, $R_1 = 0.927$, $\sqrt{R_0 R_1} = 0.877$ and 132 predicted heteromodal connections, related mainly to areas LIP, VIP, DP, 7a, FST, TF, FEF and 46 in the visual cortex. It is noteworthy that four of these areas (46, 7a, LIP and VIP) were classified as structural overlaps between the two subnetworks in the fuzzy community analysis of Nepusz et al. [92] (also presented later in Section 4.2). Anatomical considerations also support the bridge-like role of these areas between the cortices. It was previously suggested in the literature that area VIP should be split into two areas (VIPm and VIPp), establishing stronger connections with visual or sensorimotor areas, respectively [73]. VIP and LIP are involved with hand and eye coordination, requiring combined input of visual and tactile signals. Area 46 is a part of the dorsolateral prefrontal cortex, and it does not have functions related to low-level sensory information processing. Being a higher level (supramodal) area, it integrates visual, tactile and other informations. Area 7a integrates visual, tactile and proprioceptive signals. Finally, areas TF and FEF are also high level structures integrating widespread cortical information (e.g., [39]).

The full predicted connectivity matrix can be found in [91]. However, in order to demonstrate the subtle differences between predicted connections, the exact probabilities are shown on Figure 4.3, encoded in the background colour of the matrix cells (white belonging to zero probability and black belonging to 1). This figure shows the prediction in its full detail, especially in the sensorimotor cortex where the predicted clique-like subgraph reveals its internal structure more precisely.

Out of the 225 unknown connections in the visual cortex, 46 were predicted differently when I took into account the sensorimotor cortex. The most discrepancies involved the outgoing edges of VOT (10 mismatches), PIP (6 mismatches) and TF (6 mismatches). These can be caused by the additional information present in the system in the form of heteromodal connections. At the same time, predictions errors related to the known visual connections of visual areas having heteromodal connections decreased (e.g., area TF: 13 to 6, area 46: 15 to 4), due to the same additional information. Other notable improvements were at V4 (21 to 13) and DP (12 to 7).

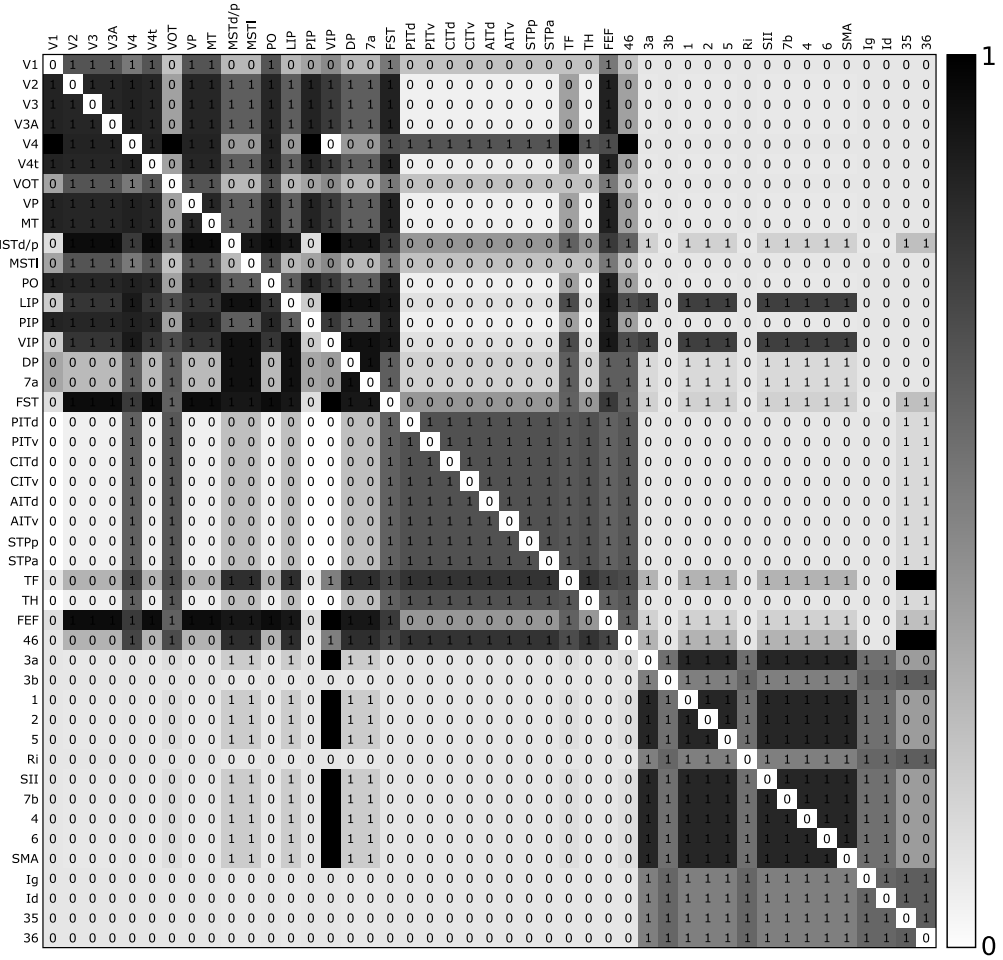


FIGURE 4.3: Probability of connections in the visuo-tactile cortex with 10 vertex groups, $\tau_1 = 0.49$ and $\tau_2 = 0.137$. Probabilities are denoted by colours, with white corresponding to 0 and black corresponding to 1. The predicted adjacency matrix is shown in the matrix cells.

The reconstruction quality of the visual cortex was improved by adding the information about the heteromodal connections and the sensorimotor cortex. This was not a simple consequence of increasing the number of clusters from 7 to 10, but the corollary of the additional information about the connections that visual areas form with the sensorimotor cortex. It is an example of a phenomenon how the inclusion of even some imprecise information about connections a substructure forms with its surrounding may improve our understanding of connectional structure within the substructure itself. This contextual information may also give guidelines for understanding the mechanisms of heteromodal interactions.

4.1.3 Other prediction approaches

Sections 2.1.1 and 2.1.2 presented a wide variety of approaches to link prediction. These methods are useful to extrapolate into the future of the time evolution of a given network from the present (and possibly the past) state, but they assume that the present state is known and accurate. This assumption does not hold for cortical networks. To justify my markedly different approach to the problem, I tested several similarity and path ensemble based methods on the visual and the visuo-tactile network. Unknown edges were treated as nonexistent. The tested methods were: cocitation and bibliographic coupling [63, 116], Jaccard and Adamic–Adar similarity indices [1, 54], SimRank [55] and the recent hierarchical random graph (HRG) model of Clauset et al. [22]. The predictive power of these measures was assessed visually by ROC curves and quantitatively via the area under the ROC curve. As shown on Figure 4.4, the fitted preference model outperformed all alternative approaches in the case of the visuo-tactile cortex. Results were similar for the visual part of the visuo-tactile cortex. The superior performance of the present method can be explained by the fact that it readily made use of the distinction between unknown and confirmed nonexistent connections, while none of the other measures were able to do so. The HRG model has the potential to improve its performance, since the present implementation[2] is not yet able to take edge directions into account.

[2]Available at `http://www.santafe.edu/~aaronc/randomgraphs/`.

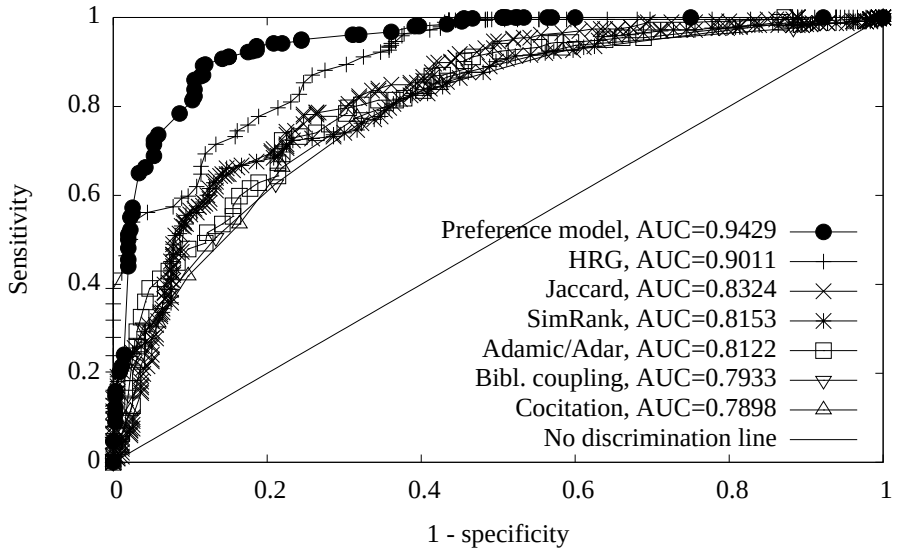


FIGURE 4.4: Comparison of the present prediction method to alternative approaches by ROC curves obtained on the visuo-tactile cortex dataset. Data points were calculated at all possible threshold levels, but only every tenth point is displayed for SimRank, the Adamic–Adar measure and the hierarchical random graph model (HRG) to keep the figure comprehensible

4.2 Higher level brain areas in the visuo-tactile cortex

Let us examine the visuo-tactile cortical network presented in the previous section from a different viewpoint: I will demonstrate the presence of fuzzy communities and bridge vertices in this network. Earlier analyses confirmed that the network has a distinct community structure [86], which is no surprise, since it consists of the union of visual and sensorimotor areas, and only a few connections are heteromodal. It was more interesting that besides this simple bisection of the network, community analysis was able to distingush the dorsal and ventral stream of the visual cortex (both are known and anatomically meaningful substructures), split the sensorimotor cortex into two parts (unfortunately the biological implications are beyond the scope of this dissertation) and isolated area 46 as an area that forms a single community alone. This is in concordance with the presumed role of area 46: it sustains attention and working memory, integrating visual, tactile and other information from various parts of the cortex that are necessary for the above mentioned cognitive functions. Fuzzy community detection was then applied to detect the presence of other bridge-like areas in the network. The approach

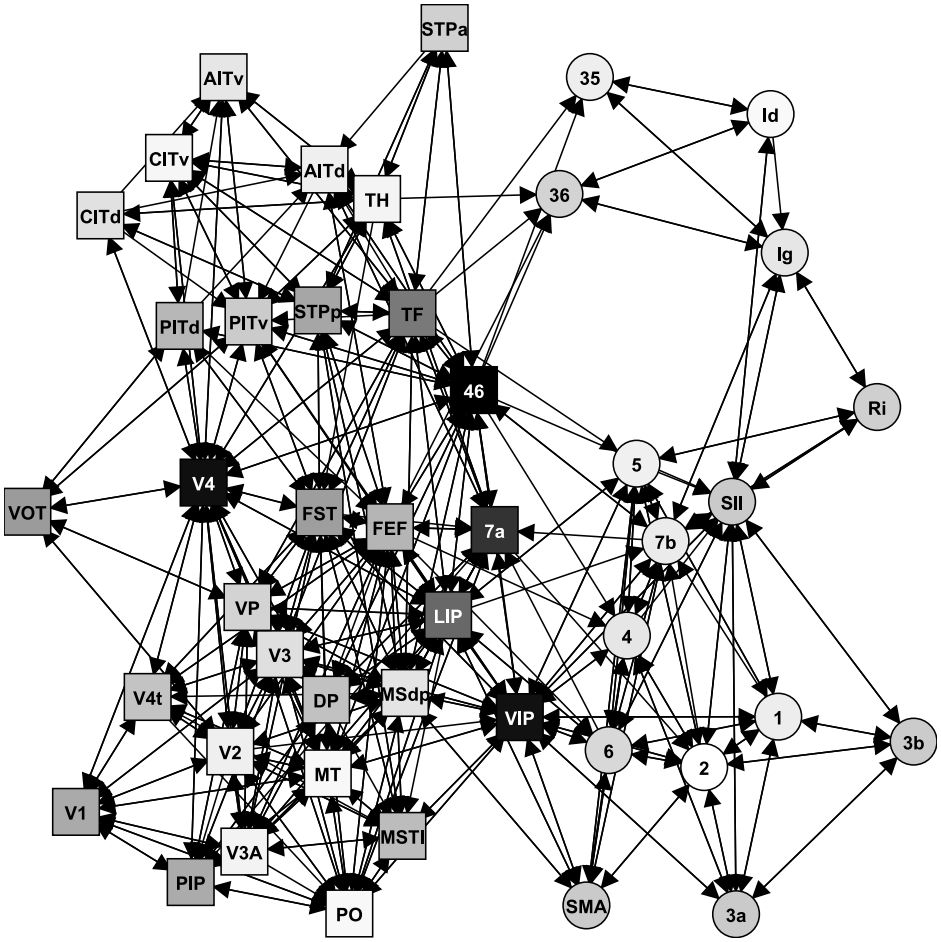


FIGURE 4.5: Bridge areas in the visuo-tactile cortex of the macaque monkey. Vertices are coloured by their degree-weighted bridgeness values.

I used was discussed in Chapter 3 and in [92].

4.2.1 Results

Since the network is directed, it had to be symmetrised before applying the method presented in Chapter 3. I removed all non-reciprocal connections and kept only mutual connections between areas. Edges were assigned uniform weights ($\mathbf{W} = [1]$). The best fuzzy modularity ($Q_f = 0.2766$) was obtained with four clusters, but I also studied the fuzzy bisection of the network into two groups.

The case of $c = 2$ classified the nodes correctly: all the somatosensory areas were unambiguously associated with the sensorimotor cortex, and most

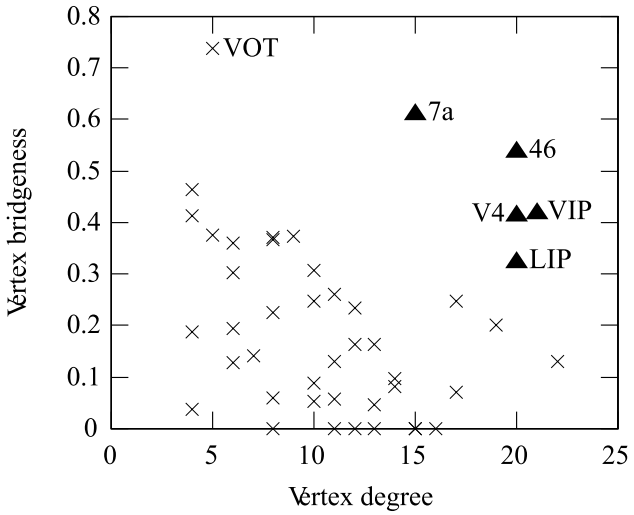


FIGURE 4.6: Bridgeness scores versus vertex degrees in the cortical network dataset. Vertices with high bridgeness and high degree are considered bridges – these are highlighted. Area VOT is considered an outlier due to its high bridgeness and low degree.

of the visual areas were associated with the visual cortex, except a few areas with a surprisingly high bridgeness (over 0.85). The vertex with the highest bridgeness (0.99) was area 46 as expected, confirming its bridge-like role and presenting an explanation of it being in an isolated cluster in the earlier crisp community analysis. Other relevant bridges found with $c = 4$ were area VIP (where the literature has already suggested that it should be split into two areas VIPm and VIPp, which establish stronger connections with visual or sensorimotor areas, respectively [73]), LIP, V4 and 7a. VIP and LIP are involved with hand and eye coordination, respectively, and both of these functions require combined information from visual and tactile signals as well. Area 7a integrates visual, tactile and proprioceptive signals. Area V4 was defined originally as the human color center [75, 79], while it was also suggested that a separated ensemble of V4 neurons successfully encode complex shapes based on the curvature of the shape boundaries [104]. The functional heterogenity is in accordance to the subdivision of V4 into different regions as suggested by Bartels and Zeki [8]. I concluded that the bridges I found are in concordance with the assumed higher level roles of these areas. Fuzzy community detection for $c = 4$ was also able to separate the dorsal and the ventral stream of the visual cortex, only area 7a and VIP were misclassified, but they retained their bridge-like properties as well as area 46. The degree-weighted bridgeness values for $c = 4$ are shown on Figure 4.5.

Palla et al. [102]	$k = 5$	V4, PITv, TF
	$k = 6$	LIP, VIP
	$k = 7$	VIP
Zhang et al. [135]	$m = 2$	40% of the vertices
	$m = 1.3$	LIP, 7a, Ri
Capocci et al. [17]	$k = 4$	V4, 46
Present method	$c = 2$	46
	$c = 4$	46, VIP, LIP, V4, 7a

TABLE 4.6: Identified bridge vertices in the cortical network dataset by various overlapping community detection algorithms. Note that every bridge vertex identified by the present method is also confirmed by at least one alternative approach.

Plotting the unweighted bridgeness values versus a chosen centrality measure (in our case, the vertex degree), shown on Figure 4.6 was found to be a useful visual cue for separating bridge vertices and outliers.

4.2.2 Comparison with other approaches

In order to compare the present method with earlier attempts on tackling the problem of overlapping communities, I examined the CPM algorithm of Palla et al. [102], the spectral method of Capocci et al. [17] and the fuzzy method of Zhang et al. [135] on the cortical network dataset. For the CPM algorithm, I used the original implementation published by the authors at `http://www.cfinder.org`. The algorithm of Zhang et al. [135] had a weight exponent m controlling the degree of fuzzification, but since the authors provided no clue about the suggested value of the parameter, I used $m = 2$, which is the most typical choice of this parameter in other known applications of the fuzzy c-means algorithm [9].

The community structure of the cortical graph seemed to be a hard problem for the algorithms. The method of Palla et al. [102] failed to discover the subdivision of the visual cortex into dorsal and ventral parts, only the visual and the somatosensory cortex was discovered when I used a clique size of 5. Larger clique sizes resulted in the discovery of the cores of the two communities, but I was still not able to recognise the subdivision of the dorsal and the ventral stream in the visual cortex. However, the algorithm identified three overlaps (V4, PITv and TF) for a clique size of 5 and two other overlaps (LIP and VIP) for a clique size of 6. Three out of these five overlaps were identified by my algorithm as well. The community closeness matrix calculated by the method of Capocci et al. [17] was harder to interpret, but vertices V4

and 46 clearly turned out to be bridges with zero community closenesses to many other vertices. The method of Zhang et al. [135] was highly sensitive on the exact value of parameter m, classifying 40% of the vertices as bridges for $m = 2$. (Since the method provides a membership matrix similar to the present method, I used the standardised bridgeness measure with a z-score threshold of 1). Lowering the weight exponent to $m = 1.3$ identified areas LIP, 7a and Ri as bridges. These results are also shown in Table 4.6.

To summarise, I found that the results of the algorithm presented here with respect to community structure discovery and bridge identification do not contradict the results of existing methods, and all the bridges found by my algorithm were classified as bridges by at least one other method. The method of Capocci et al. [17] complements this algorithm especially well, since it discovers local communities around a given vertex using the community closeness degrees while the present fuzzy method provides useful insights into the global structure of the network being analysed, also indicating the presence of bridge vertices.

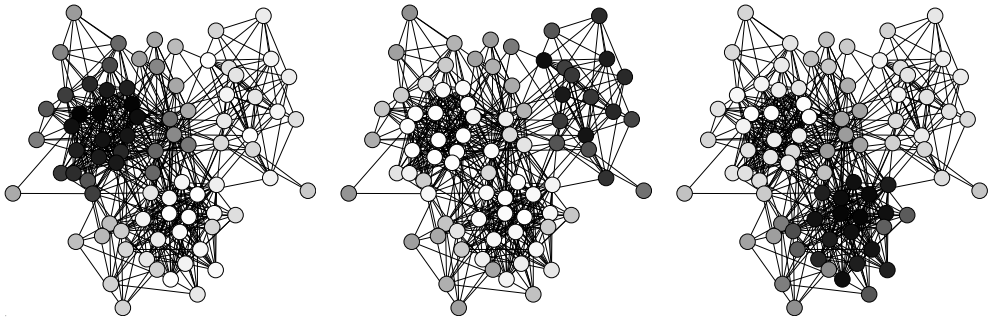
4.3 Detection of social bridges via fuzzy communities

4.3.1 The UK university faculty dataset

In this study, I used the social network of the academic staff of a given faculty of a UK university consisting of three separate schools [92][3]. This network was already shown on Figure 1.4 in Chapter 1.2.4. The network structure was constructed from social tie strength measured with a questionnaire (derived from the one published in [105] by adapting it to the academic field), where the items formed a reliable scale. Reliability was assessed by Cronbach's α [23]. Our questionnaire achieved a Cronbach's α of 0.91, suggesting high internal consistency and reliability. The questionnaire was completed by every member of the academic staff. This study used the personal friendship network, ignoring the directionality and the weight of the edges. An unweighted fuzzy community detection for three communities was performed on the graph. To show the results in greyscale, I decided to draw three individual figures (all shown on Figure 4.7), showing the values of the membership functions for community 1, 2 and 3, respectively, using different shades of grey as fill colors for the vertices.

This dataset also contained explicit information regarding the expected

[3]The dataset is available at `http://www.mit.bme.hu/~nepusz`



(a) Community 1 (b) Community 2 (c) Community 3

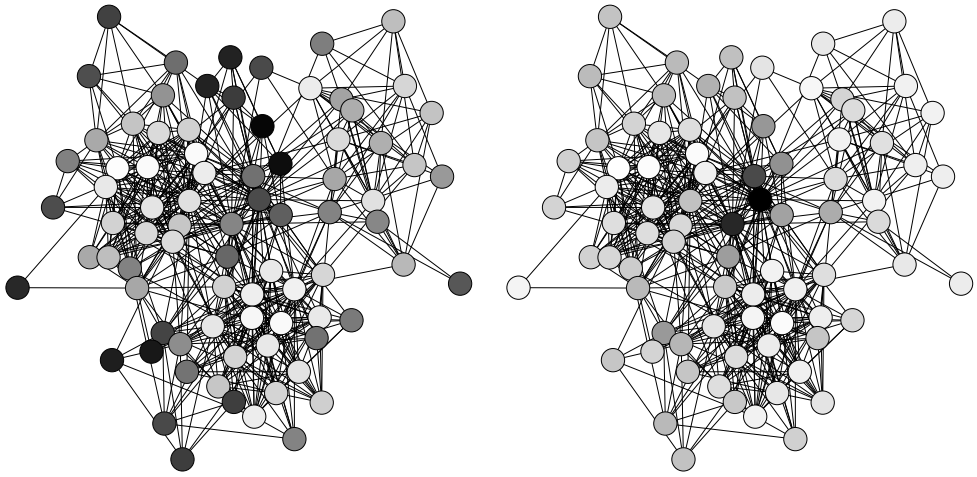
FIGURE 4.7: Fuzzy communities of the UK university faculty dataset. Vertices are coloured according to the membership functions of community 1, 2 and 3, respectively. Darker shades represent larger membership values.

community structure, since the school affiliations of staff members were attached to the vertices as textual attributes. I defuzzified the results using the dominant communities for every vertex. The defuzzification revealed that all crisp communities consisted of almost exclusively the members of a single school inside the Faculty. 75 out of 81 vertices were classified correctly, 4 were misclassified (and all of them had a bridgeness value greater than 0.7), and there were 2 vertices for which no expectation was given because of lack of information in the questionnaire. It is also noteworthy that the maximal fuzzy modularity ($Q_f = 0.2826$) was reached at $c = 6$, suggesting further subdivisions of the schools, although the improvement of the modularity compared to the case of $c = 3$ ($Q_f = 0.2541$) was not significant.

Degree-corrected bridgeness scores for $c = 3$ (Figure 4.8(b)) are particularly interesting. Highly scored individuals belong to all three communities at the same time to some extent, maintaining connections to all of them. On the other hand, vertices with low degree-corrected bridgeness scores can be thought as the cores of the communities. I also notice that the peripheries of the communities also belong almost equally to all of the communities (note the similar grey shades in Figure 4.7 for these vertices), but the degree-corrected bridgeness scores suppress this effect because of their low degree. The uncorrected and the degree-corrected scores are compared side-by-side on Figure 4.8.

4.3.2 The network science co-authorship graph

This section discusses the fuzzy community analysis of the co-authorship network of scientists involved in network science [96]. The network consists of



(a) (b)

FIGURE 4.8: Comparison of the unweighted (left) and degree-weighted bridgeness scores (right) in the UK university dataset. Vertices are colored according to their respective bridgeness scores. Darker shades represent higher bridgeness scores. Note how the uncorrected bridgeness score correlates with the centrality of the vertices in their respective community.

1589 scientists and 2742 weighted, undirected connections. Edge weights are derived from the number of joint publications: a paper with n authors contributes $1/n$ to the weight of all possible edges between the authors. Edges with a weight less than 0.2 (equivalent to only a single joint publication with at least 6 authors) were removed and the remaining weights were disregarded. The largest component of the cutted network consisted of 342 scientists and 765 connections. Fuzzy community analysis was performed on this component using the $\mathbf{W}$ matrix derived from the configuration model, as suggested in Section 3.3.2.

By looking at Figure 4.10, it is evident that the network is sparse, with a few hubs and many low degree vertices. Recall that the weight of vertex pairs is determined by $(A_{ij} - d_i d_j/(2m))^2$, where d_i and d_j are the degrees of the vertices involved, A_{ij} is zero if they are disconnected and one if they are connected, m is the number of edges in the whole network. Therefore, lower weight will be assigned to pairs of disconnected vertices if both have a low degree, and similarly, lower weight will be assigned to connected hubs. The important features of the network that will govern the resulting fuzzy community structure will be pairs of connected low-degree vertices and pairs of disconnected hubs. By analysing the distribution of w_{ij} values, it turns out that only approximately 10% of vertex pairs have weight larger than 10^{-3}

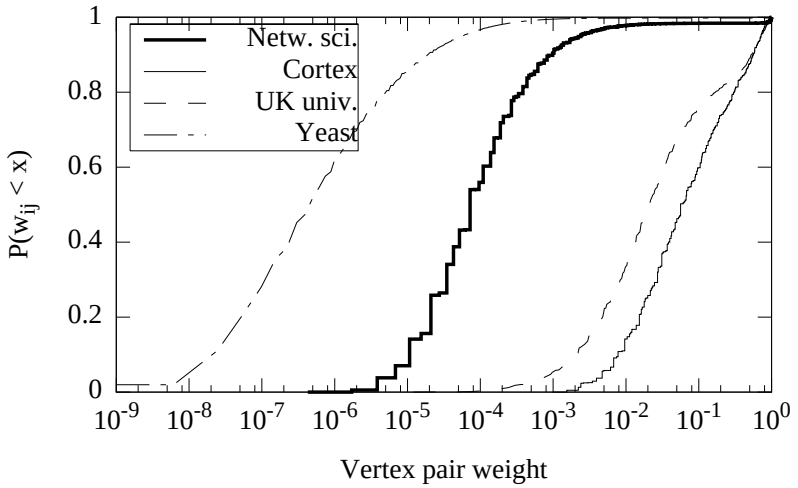


FIGURE 4.9: Cumulative vertex pair weight distribution in various datasets. Dashed lines are smoothed for the sake of better visibility. Note the logarithmic x axis and the extreme positive skewness of the distributions which is directly associated with the size of the network (cortex: 45 vertices, UK university dataset: 81 vertices, network science dataset: 342 vertices, yeast protein interaction network: 2640 vertices). In case of the network science dataset, only approximately 10% of vertex pairs have weight larger than 10^{-3}, the rest can simply be considered zero.

(see Figure 4.9). The remaining 90% can simply be considered zero, allowing efficient and fast computation of gradient vectors and the goal function. This is typical not only in this particular network, similar right-skewed distributions can be observed in the cortical network and the UK university social network dataset presented above, although the effect is less pronounced, since these networks have less vertices (45 and 81, respectively). The direct relation between the size of the network and the skewness of the distribution is also reinforced by the protein interaction network of the yeast [58], which has 2640 vertices.

The optimal number of communities was found to be 21, with a fuzzy modularity of $Q_f = 0.7221$. This shows a remarkable difference from prior unweighted analysis [92] where slightly lower fuzzy modularity (0.7082) was achieved with a higher number of communities. 245 out of the 342 vertices had a bridgeness score less than 0.01, and only 40 vertices had a standardized bridgeness score larger than or equal to 1. 34 vertices were classified as bridges based on the number of their significant communities (χ_i values). These vertices are marked on the visualisation of the network (see Fig. 4.10). No vertex had more than two significant communities (χ_i ¡ 1.998 for all

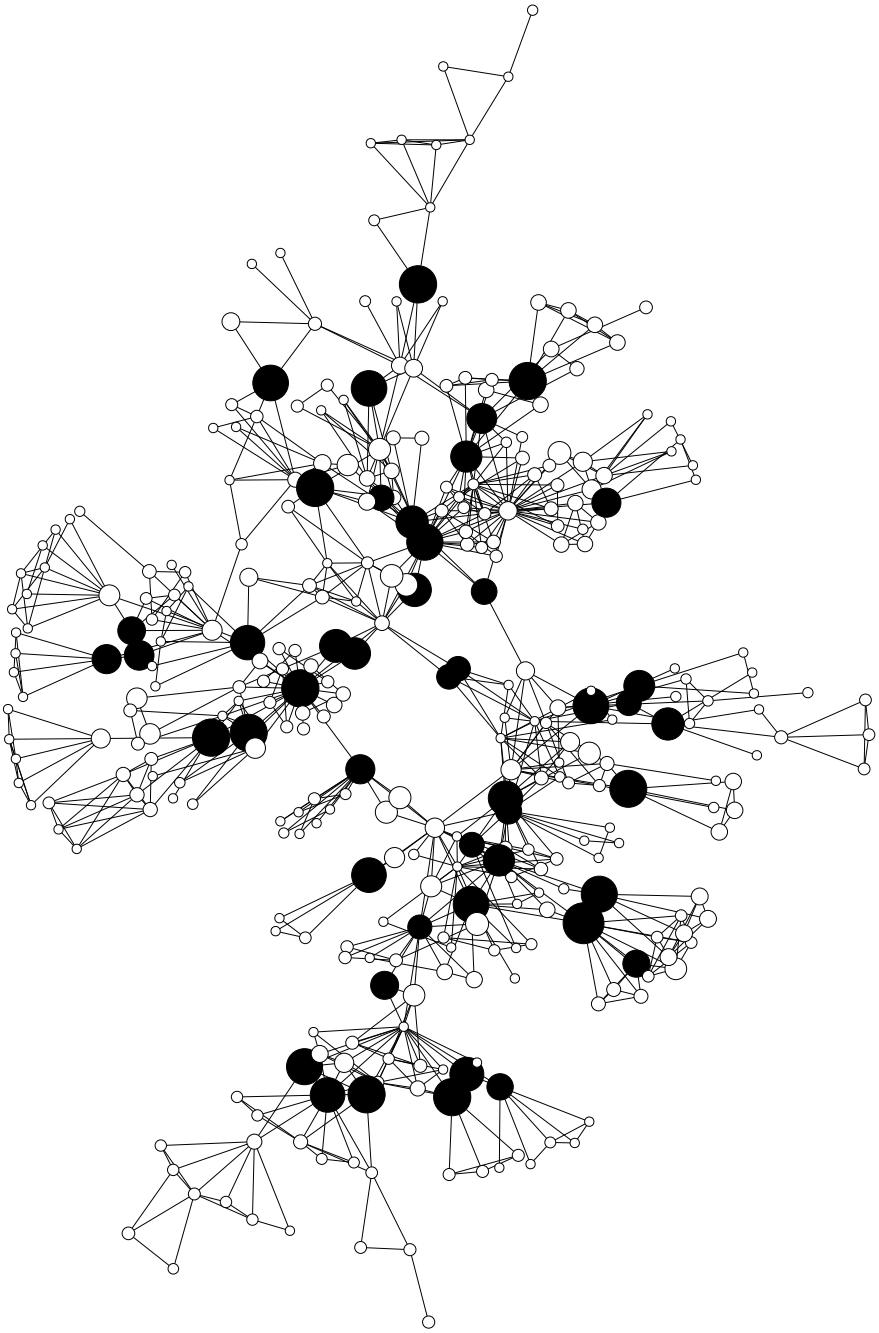


FIGURE 4.10: Visualisation of the network science co-authorship dataset. Vertex diameters are proportional to the number of significant communities χ_i. Bridge vertices ($\chi_i > 1.5$) are black, regular vertices are white.

vertices). Without mentioning names, I note that senior researchers and group leaders of the field mostly participate in only one community, with a few exceptions. This supports the community centrality measure of Newman [96]: researchers with high community centrality are not likely to be bridges between communities.

One may argue that the degree of vertices can be correlated with the number of relevant communities, for vertices with less connections are less likely to participate in multiple communities. However, the analysis of the network presented here does not seem to confirm this assumption. Although the average degree of bridge vertices is slightly higher (6.656 with standard deviation 3.579) than the mean degree of the largest component (4.473 with standard deviation 3.787), there is no trace of direct correlation between the two measures. There is only a very moderate positive correlation between the degree of a vertex and the number of its relevant communities according to a variety of correlation coefficients ($r^2 = 0.089$, Pearson's $\rho = 0.299$, Kendall's $\tau = 0.267$), showing that even vertices with a low degree can become bridges.

5

Conclusions

5.1 Link prediction in complex networks

The first part of the dissertation depicted a random graph model called the *preference model*, which can be considered as a generalisation of the well-known Erdős–Rényi random networks. The model is based on the assumption that the internal, latent traits of individual vertices can efficiently be modelled by vertex types, i.e., every vertex belongs to one of k distinct types (or two in the case of directed networks), and the connection probability of two vertices depends solely on their vertex types and a preference matrix $\mathbf{P}$ that assigns probabilities to vertex type pairs [89, 91].

I implemented two stochastic algorithms to fit the parameters of the model to a network instance being studied, one based on the expectation-maximisation (EM) algorithm scheme and the other one based on the Metropolis–Hastings algorithm. The fitted group assignments may provide insights into the inner structure of the network [88]; e.g., group assignments in a network with a distinct community structure are likely to correspond to the communities of the network and the elements of the preference matrix will contain higher values in the diagonal than outside. Similarly, a nearly bipartite network will be fitted by group assignments corresponding to the (almost) independent vertex sets, and elements of the preference matrix will be close to zero in the diagonal.

Finally, I showed how can the preference model be used to predict potentially missing links in networks whose edge set is known to be incomplete. The advantage of the method is that it is able to take into account the uncertainty of unknown connections, while other approaches (e.g., local similarity measures, see Section 2.1.1) treat all unknown connections as nonexistent.

An application of the method was presented in Section 4.1 and in [91], where missing neural connections of the visuo-tactile cortex of the macaque monkey were predicted mostly in agreement with previous analyses [25, 59], but with higher reconstruction accuracy in known parts of the network. The accuracy of the method presented here was proven to be the best so far among reconstruction attempts of the cortical network.

This part of my dissertation can be summarised as follows:

T 1/1. I showed that vertex degrees in the networks generated by the preference model are described by the weighted sum of Poisson-distributed random variables. I also proved a sufficient condition for the existence of a giant component in these networks.

T 1/2. I devised and implemented algorithms to fit the parameters of the model to a given network, taking into account the degrees of belief associated to the possible connections in the network. I tested the validity of these algorithms on computer-generated test graphs.

T 1/3. I showed that the Akaike information criterion [2] is able to choose the most appropriate number of vertex groups of the model in an unsupervised manner.

I applied the model to the problem of predicting yet uncharted connections in the visuo-tactile cortex of the macaque monkey [86] and I gave estimates on the probability of the existence of uncertain links. The fitted model also produces the most accurate reconstruction of the known part of the visuo-tactile cortex so far. These results were published in [88, 89, 91].

From the theoretical point of view, there are still a lot of open questions to be investigated. E.g., it is interesting to ask how the size of the giant community depends on the preference matrix; what is the distribution of cluster sizes in such generated networks; how does the expected diameter of the network scale with the number of vertices and so on. From the algorithmical point of view, the possibility of obtaining fuzzy group assignments by appropriately sampling from the Markov chain of model parameterisations is definitely interesting, as it may provide a remedy to the overfitting problem, and it may also be useful in increasing the accuracy of reconstruction and prediction.

5.2 Fuzzy communities in complex networks

In the second (and admittedly shorter) part of the dissertation, I presented a data mining algorithm that detects fuzzy communities in complex networks

[92]. The algorithm determines a fuzzy partition matrix of the vertices of the network into c predefined groups in a way that maximises a prescribed goal function that is assumed to yield higher values for meaningful fuzzy partitions. The goal function is grounded in the concept of vertex similarity: two vertices are considered similar if they belong to similar groups according to the fuzzy partition matrix being assessed, and the algorithm strives to maximise the similarity of connected vertex pairs and minimise the similarity of disconnected vertices.

I devised an optimisation method to maximise the goal function using either a local hill climbing algorithm with adaptive step size and occasional random mutations in order to climb out from local minima, or the iterative reassignment of vertices to communities based on the BFGS algorithm [113]. The validity of the method was then tested on computer-generated random graphs and real datasets as well.

I also introduced three new measures (bridgeness, weighted bridgeness and exponentiated entropy, see Section 3.4) that are able to quantify the sharedness of a given vertex between communities. These measures can be used to select individual vertices from a complex network that serve as bridges between separate communities. The importance of bridgeness vertices is evident: in social networks, individuals represented by bridge vertices are responsible for the flow of information between social groups; in cortical networks, bridge areas are associated with higher level cognitive functions (see Section 4.2); in protein interaction networks, bridges can be proteins with multiple functions [102] and so on. Some applications of the algorithm and the introduced bridgeness measures were presented in Sections 4.2 and 4.3. Finally, I also compared the present method to other approaches in Section 4.2.2.

The summary of this part of my dissertation is as follows:

T 2/1. I devised and implemented an algorithm to find fuzzy communities in undirected networks. The algorithm is based on the maximisation of a global goal function derived from vertex similarities. I tested the validity of the algorithm on computer-generated test graphs.

T 2/2. I extended the modularity measure of Newman [94] to account for the fuzziness of the obtained partitions. I showed how can one employ the fuzzified modularity to choose the optimal number of communities.

T 2/3. I quantified the sharedness of vertices between fuzzy communities by introducing the bridgeness, the weighted bridgeness and the exponentiated entropy measures of the membership vectors.

These results were published in [92].

The algorithm can be extended in various ways, which will serve as the basis of future research. For instance, there are many alternative definitions of vertex similarity, e.g., one can try using the cosine similarity of membership vectors instead of the dot product. The relaxation of the sum constraint imposed on membership vectors would allow outliers in the obtained fuzzy partitions; outlier vertices would then be characterised by membership vectors whose coordinates sum up to less than one. The algorithm can theoretically be extended to directed networks as well, based on an idea similar to how the direction of edges was introduced into the preference model: every vertex should have two membership vectors instead of one, the first one corresponding to the incoming connections of the vertex and the second one corresponding to the outgoing connections. The similarity of two vertices can then be defined as the dot product of the *out-membership vector* of the first one and the *in-membership vector* of the second one. Prior analyses showed that this is a promising direction [90] which merits further investigation.

A

Technical background

The algorithms and computational results presented in this dissertation were obtained using tools of the GNU Linux and Mac OS X operating systems. Algorithms were implemented in a blend of GNU C and Python [126]. I exploited the functionality of the `igraph` library [24] and its higher level interface towards Python. Timing results were calculated on an 1.83 GHz Intel Core Duo MacBook. Graphs were drawn by `igraph`, numerical plots and curve fittings were performed by GNUplot [133]. GNUplot uses the Levenberg-Marquardt algorithm [76] for least-squares curve fitting with stopping parameter $\eta = 10^{-5}$.

A.1 Generating random numbers

The algorithms presented in this dissertation make heavy use of random numbers. One should not neglect the fact that computers are not able to generate truly random sequences, unless equipped by a hardware random number generator based on some unpredictable physical process (e.g., thermal noise in Zener diodes, photoelectric effect or other quantum phenomena). Most computers implement deterministic software routines instead that generate number sequences that behave plausibly random for most purposes, although they fail rigorous tests like the Kolmogorov-Chaitin randomness test [109]. The proof is straightforward: pseudo-random number generators produce longer bit sequences than their own length in bits, hence the Kolmogorov complexity of the generated sequence does not tend to 1 as the length of the sequence approaches infinity.

Since every pseudo-random number generator possesses some built-in deterministic property, it might happen that the generated sequence interferes

with the computational problem requiring random input, thus distorting the obtained results. The algorithms presented in this dissertation are no exceptions, therefore I tested them with three different random number generators:

`drand48`. The preferred random number generator of systems complying to the System V Interface Definition (SVID) [112], which includes Linux and OS X as well. It uses an internal state of 48 bits which is updated according to the following congruential equation:

$$x_{t+1} \equiv ax_t + c \pmod{2^{48}} \tag{A.1}$$

a and c are constants, $a = 25214903917$ and $c = 11$ by default, but they can be changed by the user. It simply follows that the period of the random number generator is at most 2^{48}.

Wichmann-Hill generator [132]. It uses four linear congruential generators similar to the one used by `drand48`. The sequence is formed by the linear combination of the output of the individual generators. There are 273 predefined sets of parameters to choose from, selected in a way that ensures the independency of the generators and resulting in a period of approximately 2^{80}. The Wichmann-Hill generator was the standard random number generator of Python up to and including version 2.3.

Mersenne Twister generator [77]. The standard random number generator of Python from version 2.4. This is a twisted generalised feedback shift register generator, producing 53-bit precision floats. It has an extremely long period of $2^{19937} - 1$ (a Mersenne prime), and has been shown to be uniformly distributed in 623 dimensions.

I found that the results obtained by any of the random generators matched the others.

A.2 The `igraph` library

The `igraph` library is a handy tool for the analysis of large networks, written almost completely in C for speed and efficiency. It was developed because of the lack of network analysis software which (1) can handle large graphs efficiently, (2) can be embedded into a higher level programming language (e.g., Python, Ruby or GNU R) and (3) can be used both interactively and non-interactively. Embedding `igraph` into Python or GNU R creates a productive

research environment, since one can make use of the rapid development and prototyping features of the language in which `igraph` is embedded.

The key features of `igraph` are as follows:

Open source. `igraph` is free for non-commercial or commercial use according to the terms of the GNU General Public License. The source code of `igraph` is readily available, anyone can add new functionality and correct deficiencies.

Efficient implementation. `igraph` uses an indexed edge list representation for graphs, tailored to networks that do not change rapidly. This representation will be discussed later in detail.

Layered architecture. The internal graph representation is exposed to most of the `igraph` routines via a well-defined interface, which enables one to replace the graph representation with another one (e.g., adjacency lists), should the need arise. The new representation has to implement a small set of well-defined functions – all other parts of the source code can be left intact.

Open, embeddable system. The current `igraph` distribution contains interfaces to R, Python and Ruby, and interfaces to other languages can be added without too much effort.

`igraph` can be downloaded from `http://cneurocvs.rmki.kfki.hu/igraph`.

A.2.1 The basic graph representation in `igraph`

The key data structure in `igraph` is a C struct called `igraph_t`:

```
1:  typedef struct {
2:    igraph_integer_t n;     /* The number of vertices */
3:    igraph_bool_t directed; /* Is the graph directed? */
4:    igraph_vector_t from;   /* Edge list, 1st column */
5:    igraph_vector_t to;     /* Edge list, 2nd column */
6:    igraph_vector_t oi;     /* Index of the 1st column */
7:    igraph_vector_t ii;     /* Index of the 2nd column */
8:    igraph_vector_t os;     /* 2nd level index by vertex IDs */
9:    igraph_vector_t is;     /* 2nd level index by vertex IDs */
10:   void *attr;             /* Attribute data */
11: } igraph_t;
```

Edges and vertices in `igraph` are numbered continuously, starting from zero. The edge list is stored in the vectors `from` and `to` (`igraph_vector_t`

is a dynamic vector type containing double-precision `float`s). The source vertex of edge `i` is given by `from[i]` and the target is given by `to[i]`.

Many `igraph` routines need to iterate over the edges in the order of their source or target vertices. The ordering according to the first column of the edge list is given by `oi`, so the first edge in the ordering defined by the source vertex IDs is given by `from[oi[0]]`. Similarly, the first edge in the ordering defined by the target vertex IDs is `from[ii[0]]`. Therefore, the length of `oi` and `ii` naturally matches the number of edges in the graph.

The vectors `os` and `is` serve as second-level indices by vertex IDs, so the ID of the first edge originating from vertex `i` is given by `from[oi[os[i]]]`, the second one is given by `from[oi[os[i]+1]]` and so on, up to and not including `from[oi[os[i+1]]]`. When a vertex does not have outgoing edges, `from[oi[os[i]]]` equals `from[oi[os[i+1]]]`. The last elements of `os` and `is` are sentinels, they are always equal to the number of edges, so the length of `os` and `is` is always equal to the number of vertices. Finally, `attr` is a pointer to a data structure holding graph, vertex and edge attributes. Its type is unspecified, since attributes are handled by functions implementing a separate attribute handler interface. The attribute handler used depends on the environment in which `igraph` is embedded: there is a C attribute handler which enables numeric and strings attributes, there is an R attribute handler which can store arbitrary R objects in the attributes, there is a Python attribute handler for storing Python objects and so on. In the latter cases, the actual attribute handling is done by the host language and not `igraph` itself. The storage requirements for a graph with n vertices and m edges without attributes is $O(n + m)$.

The above data structure is efficient when the graph is static at the expense of slower edge insertion and deletion: adding or removing an edge can be done in $O(n + m)$ time due to the additional bookkeeping associated to the indices. However, note that adding or removing multiple edges can also be done in $O(n + m)$ time, where m is the number of edges after addition or before deletion. When confronted with a task involving repeated addition and deletion of many edges at once (e.g., graph rewiring while keeping the degree distribution), the graph can be converted to a more efficient unindexed adjacency list representation and converted back after completion. The conversion itself also takes $O(n + m)$ time, but this has to be done only once. The adjacency list representation is also useful when a specific task involves querying the neighbours of (almost) all vertices many times, since the neighbour vectors are created only once. `igraph` also provides a lazy adjacency list representation.

A.2.2 An example: calculating SimRank scores

In this section, I illustrate the basic syntax of the Python interface of `igraph` by prototyping an implementation of the SimRank similarity score [55]. Note that the core of the implementation does not take more than 20 lines.

```
1:  from igraph import *
2:
3:  def simrank(graph, gamma=0.8, epsilon=1e-3):
4:    n = graph.vcount()
5:    neis = [graph.predecessors(i) for i in range(n)]
6:    sim = Matrix.Identity(n)
7:    max_change = 1
8:
9:    while max_change > epsilon:
10:     max_change = 0
11:     new_sim = Matrix.Identity(n)
12:
13:     for v1 in range(n):
14:       for v2 in range(v1+1, n):
15:         s = sum([sim[a,b] for a in neis[v1] for b in neis[v2]])
16:         if s > 0: s /= len(neis[v1]) * len(neis[v2])
17:         new_sim[v1,v2] = new_sim[v2,v1] = gamma * s
18:         max_change = max(max_change, abs(new_sim[v1,v2] - sim[v1,v2]))
19:
20:     sim = new_sim
21:
22:   return sim
23:
24: ###############################################################################
25:
26: graph = load("cortical_network.graphml")
27: simrank_score = simrank(graph)
```

Bibliography

[1] L. A. Adamic and E. Adar. Friends and neighbors on the Web. *Social Networks*, 25(3):211–230, 2003.

[2] H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974.

[3] R. Albert and A.-L. Barabási. Statistical mechanics of complex networks. *Reviews in Modern Physics*, 74(1):47–94, Jan 2002. doi: 10.1103/RevModPhys.74.47.

[4] R. Albert, H. Jeong, and A.-L. Barabási. The diameter of the World Wide Web. *Nature*, 401:130–131, 1999.

[5] N. Alon, R. A. Duke, H. Lefmann, V. Rodl, and R. Yuster. The algorithmic aspects of the Regularity Lemma. *Journal of Algorithms*, 16 (1):80–109, 1994.

[6] A.-L. Barabási. *Linked: How Everything Is Connected to Everything Else and What It Means*. Plume, 2003. ISBN 978-0452284395.

[7] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.

[8] A. Bartels and S. Zeki. The architecture of the colour centre in the human visual brain: new results and a review. *European Journal of Neuroscience*, 12(1):172–193, Jan 2000.

[9] J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum, New York, USA, 1981. ISBN 978-0306406713.

[10] J. C. Bezdek and S. K. Pal. *Fuzzy Models for Pattern Recognition: Methods that Search for Structures in Data*. IEEE Press, New York, USA, 1992.

[11] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D.-U. Hwang. Complex networks: Structure and dynamics. *Physics Reports*, 424: 175–308, 2006.

[12] B. Bollobás. *Random Graphs*. Cambridge University Press, 2nd edition, 2001. ISBN 978-0521797221.

[13] V. Braitenberg and A. Schüz. *Cortex: Statistics and Geometry of Neuronal Connectivity*. Springer, 1998. ISBN 978-3540638162.

[14] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107–117, 1998.

[15] M. Buchanan. *Nexus: Small Worlds and the Groundbreaking Theory of Networks*. W. W. Norton & Co., 2003. ISBN 0-393-04153-0.

[16] R. S. Burt. *Structural Holes: the Social Structure of Competition*. Harvard University Press, Cambridge, MA, USA, 1992. ISBN 978-0674843714.

[17] A. Capocci, V. D. P. Servedio, G. Caldarelli, and F. Colaiori. Detecting communities in large networks. *Physica A*, 352:669–676, 2005.

[18] B. Cheswick, H. Burch, and S. Branigan. Mapping and visualizing the Internet. In *ATEC'00: Proceedings of the Annual Technical Conference on 2000 USENIX Annual Technical Conference*, 2000.

[19] F. R. K. Chung. Laplacians and the Cheeger inequality for directed graphs. *Annals of Combinatorics*, 9:1–19, 2005.

[20] F. R. K. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997. ISBN 978-0821803158.

[21] A. Clauset, M. E. J. Newman, and C. Moore. Finding community structure in very large networks. *Physical Review E*, 70:066111, 2004.

[22] A. Clauset, C. Moore, and M. E. J. Newman. Structural inference of hierarchies in networks. In *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, June 2006.

[23] L. J. Cronbach. Coefficient alpha and the internal structure of tests. *Psychometrika*, 16(3):297–334, 1951.

[24] G. Csárdi and T. Nepusz. The igraph software package for complex network research. *InterJournal Complex Systems*, page 1695, 2006.

[25] L. da Fontoura Costa, M. Kaiser, and C. C. Hilgetag. Predicting the connectivity of primate cortical networks from topological and spatial node properties. *BMC Systems Biology*, 1(16), 2007.

[26] L. Danon, J. Duch, A. Diaz-Guilera, and A. Arenas. Comparing community structure identification. *Journal of Statistical Mechanics*, page P09008, Oct 2005.

[27] L. Danon, A. Diaz-Guilera, and A. Arenas. The effect of size heterogeneity on community identification in complex networks. *Journal of Statistical Mechanics*, page P11010, 2006.

[28] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

[29] L. Devroye. *Non-Uniform Random Variate Generation*. Springer, New York, USA, 1986.

[30] R. Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer-Verlag, Heidelberg, 2005. ISBN 3-540-26182-6.

[31] S. N. Dorogovtsev and J. F. F. Mendes. *Evolution of Networks: From Biological Nets to the Internet and WWW*. Oxford University Press, Oxford, 2003. ISBN 0198515901.

[32] S. N. Dorogovtsev, J. F. F. Mendes, and A. N. Samukhin. Structure of growing networks: Exact solution of the Barabási–Albert model. *Physical Review Letters*, 85:4633, 2000.

[33] P. Drineas, A. Frieze, R. Kannan, S. Vempala, and V. Vinay. Clustering large graphs via the singular value decomposition. *Machine Learning*, 56:9–33, 2004.

[34] J. C. Dunn. A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, 3: 32–57, 1973.

[35] P. Erdős and A. Rényi. On random graphs. *Publicationes Mathematicae*, 6:290, 1959.

[36] P. Erdős and A. Rényi. On the evolution of random graphs. *Publications of the Mathematical Institute of the Hungarian Academy of Sciences*, 5:17–61, 1960.

[37] P. Erdős and A. Rényi. On the strength of connectedness of a random graph. *Acta Mathematica Hungarica*, 12:261–267, 1961.

[38] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the Internet topology. In *Proceedings of ACM SIGCOMM'99*, pages 251–262, Cambridge, MA, USA, Sept. 1999.

[39] D. J. Felleman and D. C. Van Essen. Distributed hierarchical processing in the primate cerebral cortex. *Cerebral Cortex*, 1(1):1–47, 1991.

[40] R. A. Fisher. On the mathematical foundations of theoretical statistics. *Philosophical Transactions of the Royal Society of London A*, 222:309–368, 1922.

[41] D. Fogaras. Where to start browsing the web? In *Proceedings of the 3rd International Workshop on Innovative Internet Community Systems (IICS)*, volume 2877 of *Lecture Notes in Computer Science*, pages 65–79, Leipzig, Germany, 2003. Springer-Verlag.

[42] A. L. N. Fred and A. K. Jain. Robust data clustering. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 128–133. IEEE, 2003. doi: 10.1109/CVPR.2003.1211462.

[43] M. Galassi, J. Theiler, and J. Davies. *GNU Scientific Library Reference Manual*. Network Theory Limited, 2nd edition edition, 2006. ISBN 978-0954161736.

[44] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin. *Bayesian Data Analysis*, chapter 11. Texts in Statistical Science. Chapman and Hall, London, 2nd edition, 2003. ISBN 978-1584883883.

[45] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

[46] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences of the United States of America*, 99:7821–7826, 2002.

[47] C. Goffman. And what is your Erdős number? *American Mathematical Monthly*, 76(7):791, 1969.

[48] G. H. Golub and K. William. Calculating the singular values and pseudo-inverse of a matrix. *Journal of the Society for Industrial and Applied Mathematics B*, 2(2):205–224, 1965.

[49] C. M. Grinstead and J. L. Snell. *Introduction to Probability*. American Mathematical Society, 2nd revised edition edition, 1997. ISBN 978-0821807491.

[50] J. Grossman. The Erdős Number Project. URL `http://www.oakland.edu/enp/`. Accessed on March 2, 2008.

[51] R. Guimerà and L. A. N. Amaral. Functional cartography of complex metabolic networks. *Nature*, 433:895–900, 2005.

[52] W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 1970. doi: 10.2307/2334940.

[53] R. A. Horn and C. R. Johnson. *Matrix analysis*. Cambridge University Press, 1990. ISBN 978-0521386326.

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

[55] G. Jeh and J. Widom. SimRank: A measure of structural-context similarity. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 538–543, New York, 2002. Association of Computing Machinery.

[56] G. Jeh and J. Widom. Scaling personalized web search. In *Proceedings of the 12th World Wide Web Conference*, pages 271–279. ACM Press, 2003. doi: 10.1145/775152.775191.

[57] H. Jeong, B. Tombor, R. Albert, Z. N. Oltvai, and A.-L. Barabási. The large-scale organization of metabolic networks. *Nature*, 407:651–654, 2000.

[58] H. Jeong, S. Mason, A.-L. Barabási, and Z. N. Oltvai. Lethality and centrality in protein networks. *Nature*, 411:41–42, 2001.

[59] B. Jouve, P. Rosenstiehl, and M. Imbert. A mathematical approach to the connectivity between the cortical visual areas of the macaque monkey. *Cerebral Cortex*, 8(1):28–39, 1998.

[60] F. Karinthy. Láncszemek. In *Minden másképpen van*. Atheneum, Budapest, Hungary, 1929.

[61] W. Karush. Minima of functions of several variables with inequalities as side constraints. Master's thesis, University of Chicago, Chicago, IL, USA, 1939.

[62] L. Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, 1953.

[63] M. M. Kessler. Bibliographic coupling between scientific papers. *American Documentation*, 14:10–25, 1963.

[64] J. Komlós, A. Shokoufandeh, M. Simonovits, and E. Szemerédi. Szemerédi's regularity lemma and its applications in graph theory. In *Theoretical Aspects of Computer Science*, volume 2 of *Bolyai Society Mathematical Studies*, pages 84–112. Springer, 2000. ISBN 3-540-43328-7.

[65] H. W. Kuhn and A. W. Tucker. Nonlinear programming. In J. Neyman, editor, *Proceedings of the 2nd Berkeley Symposium on Mathematical Statistics and Probability*, pages 481–492, Berkeley, 1951. University of California Press.

[66] S. Kullback and R. A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22:79–86, 1951.

[67] R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins, and E. Upfal. Stochastic models for the web graph. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 57–65, Redondo Beach, CA, USA, 2000. IEEE Press.

[68] M. Kurucz, L. Lukács, D. Siklósi, A. A. Benczúr, K. Csalogány, and A. Lukács. Telephone call network data mining: A survey with experiments. In B. Bollobás, R. Kozma, and D. Miklós, editors, *Handbook of Large-Scale Random Networks*, volume 18 of *Bolyai Society Mathematical Studies*. Springer, 2008. ISBN 978-3-540-69394-9.

[69] M. Latapy and P. Pons. Computing communities in large networks using random walks. *Journal of Graph Algorithms and Applications*, 10(2):191–218, 2006.

[70] E. A. Leicht and M. E. J. Newman. Community structure in directed networks. *Physical Review Letters*, 100:118703, 2008.

[71] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: Densification laws, shrinking diameters and possible explanations. In *Proceedings of the 11th ACM SIGKDD Conference on Knowledge Discovery in Data Mining*, pages 177–187, 2005.

[72] K. Levenberg. A method for the solution of certain non-linear problems in least squares. *The Quarterly of Applied Mathematics*, 2:164–168, 1994.

[73] J. W. Lewis and D. C. Van Essen. Corticocortical connections of visual, sensorimotor and multimodal processing areas in the parietal lobe of the macaque monkey. *Journal of Comparative Neurology*, 428:112–137, 2000.

[74] D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. In *Proc. 12th International Conference on Information and Knowledge Management*, 2003.

[75] C. J. Lueck, S. Zeki, K. J. Friston, M. P. Deiber, P. Cope, V. J. Cunningham, A. A. Lammertsma, C. Kennard, and R. S. Frackowiak. The colour centre in the cerebral cortex of man. *Nature*, 340(6232):386–389, Aug 1989.

[76] D. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics*, 11:431–441, 1963.

[77] M. Matsumoto and T. Nishimura. Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom number generator. *ACM Transactions on Modelling and Computer Simulations*, 1998.

[78] B. W. Matthews. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochimica et Biophysica Acta*, 405 (2):442–451, 1975.

[79] D. J. McKeefry and S. Zeki. The position and topography of the human colour centre as revealed by functional magnetic resonance imaging. *Brain*, 120(12):2229–2242, Dec 1997.

[80] S. Milgram. The small world problem. *Psychology Today*, 2(60), 1697.

[81] M. Mitzenmacher. A brief history of generative models for power law lognormal distributions. *Internet Mathematics*, 1:226–251, 2004.

[82] M. Molloy and B. Reed. A critical point for random graphs with a given degree sequence. *Random Structures and Algorithms*, 6:161–180, 1995.

[83] R. Montenegro and P. Tetali. Mathematical aspects of mixing times in Markov chains. *Foundations and Trends in Theoretical Computer Science*, 1(3):237–354, 2006. ISSN 1551-305X. doi: 10.1561/0400000003.

[84] C. R. Myers. Software systems as complex networks: structure, function, and evolvability of software collaboration graphs. *Physical Review E*, 68(046116), 2003.

[85] R. Neal and G. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 355–368. MIT Press, Cambridge, MA, 1999. ISBN 978-0-262-60032-3.

[86] L. Négyessy, T. Nepusz, L. Kocsis, and F. Bazsó. Prediction of the main cortical areas and connections involved in the tactile fuction of the visual cortex by network analysis. *European Journal of Neuroscience*, 23(2):1919–1930, 2006.

[87] D. L. Nelson, C. L. McEvoy, and T. A. Schreiber. The University of South Florida word association, rhyme and word fragment norms. *Behavior Research Methods, Instruments and Computers*, 36(3):402–407, 2004.

[88] T. Nepusz and F. Bazsó. Likelihood-based clustering of directed graphs. In *Proceedings of the IEEE 3rd International Symposium on Computational Intelligence and Intelligent Informatics*. Institute of Electrical and Electronics Engineers, 2007. ISBN 1-4244-1157-2.

[89] T. Nepusz and F. Bazsó. Maximum likelihood methods for data mining in datasets represented by graphs. In *Proceedings of the IEEE 5th International Symposium on Intelligent Systems and Informatics*, pages 161–165. Institute of Electrical and Electronics Engineers, 2007.

[90] T. Nepusz, F. Bazsó, and A. Petróczi. Bridge vertices in directed networks. Unpublished, submitted to Advances in Complex Systems, 2008.

[91] T. Nepusz, L. Négyessy, G. Tusnády, and F. Bazsó. Reconstructing cortical networks: case of directed graphs with high level of reciprocity.

In B. Bollobás, R. Kozma, and D. Miklós, editors, *Handbook of Large-Scale Random Networks*, volume 18 of *Bolyai Society Mathematical Studies*. Springer, 2008. ISBN 978-3-540-69394-9.

[92] T. Nepusz, A. Petróczi, L. Négyessy, and F. Bazsó. Fuzzy communities and the concept of bridgeness in complex networks. *Physical Review E*, 77:016107, 2008. doi: 10.1103/PhysRevE.77.016107.

[93] M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45:167–256, 2003.

[94] M. E. J. Newman. Fast algorithm for detecting community structure in networks. *Physical Review E*, 69:066133, 2004.

[95] M. E. J. Newman. Who is the best connected scientist? A study of scientific coauthorship networks. In E. Ben-Naim and Z. Toroczkai, editors, *Complex networks*, pages 337–370. Springer, Berlin, 2004. ISBN 978-3540223542.

[96] M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E*, 74:036104, 2006.

[97] M. E. J. Newman, S. H. Strogatz, and D. J. Watts. Random graphs with arbitrary degree distributions and their applications. *Physical Review E*, 64:026118, 2001.

[98] M. E. J. Newman, A.-L. Barabási, and D. J. Watts. *The Structure and Dynamics of Networks*. Princeton University Press, 2006. ISBN 978-0-691-11357-9.

[99] Y. Nourani and B. Andresen. A comparison of simulated annealing cooling strategies. *Journal of Physics A*, 31:8373–8385, 1998.

[100] J.-P. Onnela, K. Kaski, and J. Kertész. Clustering and information in correlation based financial networks. *European Physical Journal B*, 38: 353–362, 2004.

[101] L. Pachter, S. Batzoglou, V. I. Spitkovsky, E. Banks, E. S. Lander, D. J. Kleitman, and B. Berger. A dictionary-based approach for gene annotation. *Journal of Computational Biology*, 6(3-4):419–430, 1999. ISSN 1066-5277 (Print). doi: 10.1089/106652799318364.

[102] G. Palla, I. Derényi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, 2005.

[103] R. Pastor-Satorras, E. Smith, and R. V. Solé. Evolving protein interaction networks through gene duplication. *Journal of Theoretical Biology*, 222:199–210, 2003.

[104] A. Pasupathy. Neural basis of shape representation in the primate brain. *Progress in Brain Research*, 154:293–313, 2006.

[105] A. Petróczi, T. Nepusz, and F. Bazsó. Measuring tie-strength in virtual social networks. *Connections*, 27(2):49–57, 2006.

[106] D. J. d. S. Price. Networks of scientific papers. *Science*, 149:510–515, 1965.

[107] D. J. d. S. Price. A general theory of bibliometric and other cumulative advantage processes. *Journal of the American Society of Information Science and Technology*, 27:292–306, 1976.

[108] J. Reichardt and S. Bornholdt. Detecting fuzzy community structures in complex networks with a Potts model. *Physical Review Letters*, 93: 218701, 2004.

[109] L. Rónyai, G. Ivanyos, and R. Szabó. *Algoritmusok*. Typotex, 1999. ISBN 978-963-9132-16-0. (In Hungarian).

[110] H. H. Rosenbrock. An automatic method for finding the greatest or least value of a function. *Computer Journal*, 3:175–184, 1960.

[111] E. H. Ruspini. Numerical methods for fuzzy clustering. *Information Sciences*, 2:319–350, 1970.

[112] *System V Interface Definition*. SCO, 4th edition, 1995.

[113] D. F. Shanno. Conditioning of quasi-Newton methods for function minimization. *Mathematics of Computation*, 24:647–656, 1970.

[114] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[115] H. A. Simon. On a class of skew distribution functions. *Biometrika*, 42 (3/4):425–440, 1955.

[116] H. Small. Co-citation in the scientific literature: A new measurement of the relation between two documents. *Journal of the American Society of Information Science and Technology*, 24(4):265–269, 1973.

[117] R. Solomonoff and A. Rapoport. Connectivity of random nets. *Bulletin of Mathematical Biophysics*, 13:107–117, 1951.

[118] O. Sporns, D. R. Chialvo, M. Kaiser, and C. C. Hilgetag. Organization, development and function of complex brain networks. *Trends in Cognitive Sciences*, 8(9):418–425, 2004.

[119] M. Steyvers and J. Tenenbaum. The large scale structure of semantic networks: Statistical analyses and a model of semantic growth. *Cognitive Science*, 29(1):41–78, 2005.

[120] E. Szemerédi. Regular partitions of graphs. In *Problémes combinatoires et théorie des graphes*, pages 399–401. Centre National de la Recherche Scientifique, 1978.

[121] T. Tao. Szemerédi's regularity lemma revisited. *Contributions to Discrete Mathematics*, 1:8–28, 2006.

[122] N. Tishby, F. C. Pereira, and W. Bialek. The information bottleneck method. In *The 37th Annual Allerton Conference on Communication, Control and Computing*, pages 368–377, 1999.

[123] G. Tusnády. Personal communication., 2006.

[124] S. van Dongen. A stochastic uncoupling process for graphs. Technical Report INS-R0011, National Research Institute for Mathematics and Computer Science in the Netherlands, 2000.

[125] D. C. Van Essen and D. L. Dierker. Surface-based and probabilistic atlases of primate cerebral cortex. *Neuron*, 56:209–25, 2007.

[126] G. van Rossum. *Python Library Reference, release 2.5.2*. Python Software Foundation, February 2008.

[127] K. Wakita and T. Tsurumi. Finding community structure in a megascale social networking service. In *Proceedings of the IADIS International Conference on WWW/Internet*, pages 153–162, 2007.

[128] S. Wasserman and K. Faust. *Social network analysis*. Cambridge University Press, 1994. ISBN 978-0521387071.

[129] D. J. Watts. *Six Degrees: The Science of a Connected Age*. W. W. Norton & Co., 2003. ISBN 0-393-04142-5.

[130] D. J. Watts and S. H. Strogatz. Collective dynamics of small world networks. *Nature*, 393:440–442, 1998.

[131] J. G. White, E. Southgate, J. N. Thomson, and S. Brenner. The structure of the nervous system of the nematode Caenorhabditis elegans. *Philosophical Transactions of the Royal Society of London B*, 314:1–340, 1986.

[132] B. A. Wichmann and I. D. Hill. Algorithm AS 183: An efficient and portable pseudo-random number generator. *Applied Statistics*, (31): 188–190, 1982.

[133] T. Williams and C. Kelley. *The GNUplot Reference Manual*. URL `http://www.gnuplot.info`.

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

[135] S. Zhang, R.-S. Wang, and X.-S. Zhang. Identification of overlapping community structure in complex networks using fuzzy c-means clustering. *Physica A*, 374(1):483–490, 2007.

[136] C. Zhu, R. H. Byrd, and J. Nocedal. Algorithm 778: L-BFGS-B, FORTRAN routines for large scale bound constrained optimization. *ACM Transactions on Mathematical Software*, 23(4):550–560, 1997.

[137] M. Zhu and A. Ghodsi. Automatic dimensionality selection from the scree plot via the use of profile likelihood. *Computational Statistics and Data Analysis*, 51(2):918–930, 2006. doi: 10.1016/j.csda.2005.09.010.

[138] E. Ziv, M. Middendorf, and C. H. Wiggins. An information-theoretic approach to network modularity. *Physical Review E*, 71:046117, 2005.